ALGORITHMIC DEVELOPMENT OF KRIGING-BASED METHODS FOR COMPLEX PROBLEMS VIA IMPROVED KERNEL AND HYPERPARAMETER SELECTION

by

KEHINDE SIKIRULAI OYETUNDE

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Mechanical and Aerospace Engineering

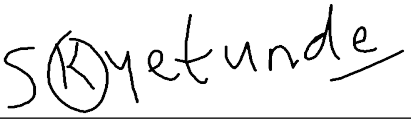
October 2022, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.



KEHINDE SIKIRULAI OYETUNDE

ALGORITHMIC DEVELOPMENT OF KRIGING-BASED METHODS FOR COMPLEX PROBLEMS VIA IMPROVED KERNEL AND HYPERPARAMETER SELECTION

by

KEHINDE SIKIRULAI OYETUNDE

This is to certify that I have examined the above Ph.D. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

Prof. Rhea. Liem, Thesis Supervisor

Prof. Qingping Sun, Head of Department

Department of Mechanical and Aerospace Engineering

28 October 2022

ACKNOWLEDGMENTS

Foremost, I would like to express my deep and sincere appreciation to my meritorious research supervisor, Professor Rhea Liem, for her substantial guidance throughout this research work, as well as her unwavering patience over the years. Such a great pleasure and honour to work under your supervision.

I would like to express my sincere appreciation to the Hong Kong PhD University Grant Council of the Hong Kong Special Administration Region for providing financial support through the Hong Kong PhD Fellowship Scheme (HKPFS) over the course of my research work. Additionally, I would like to thank and acknowledge my committee members for agreeing to serve on my thesis examination committee (TEC).

I cannot but acknowledge the support I received from my colleagues in HKUST: Jefry, Dajung, Arjit, Ikeoluwa, Harry, Bella, Yuan and many others. Thank you all for all times we worked together to meet up with deadlines and for all the fun we had in the last four years.

I thank my Nigerian families in Hong Kong, many of whom I now regard to as brothers and friends: Shem, Maradesa and his family, Ikeoluwa, Baptiste, Tomilayo, Dr. Mujib. I am greatly indebted to Miracle Adegun and Oladapo Esan for being the best brothers anyone could ask for. Thank you all for making the journey full of memorable experience and always going out of your way to offer your support.

Special thanks to all the members of the Association of Nigeria Scholars in Hong Kong (ANSHK) for giving me the opportunity to serve the association as the treasurer (2019 – 2020), then as the secretary (2020 – 2021) and as the president (2021 – 2022). It has really helped me to grow and develop my leadership skills. It is my pleasure to work with you all.

I would like to sincerely thank the Oyetunde family, particularly my parents and siblings for all the love, support, patience, understanding, prayers, encouragement, and unwavering belief in me. Thank you for being the absolute best.

Above all, I cannot thank my wife, Olamide, enough for her love, support and understanding throughout this research work. Thank you for being my editor, for always

proofreading and being a strong backbone. But most of all, thank you for being my best buddy.

May God bless you all for being a huge part of this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ALGORITHMIC DEVELOPMENT OF KRIGING-BASED METHODS FOR COMPLEX PROBLEMS VIA IMPROVED KERNEL AND HYPERPARAMETER SELECTION

by

KEHINDE SIKIRULAI OYETUNDE

Department of Mechanical and Aerospace Engineering

The Hong Kong University of Science and Technology

ABSTRACT

The rapid technological growth, whilst desirable, comes with increased complexity in real-world problems that engineers need to solve. Computational methods have been commonly employed in the design and analyses of such complex problems. Reducing computational costs of their solution methods becomes key in efficiently performing such tasks. Kriging has been a familiar solution to engineers to inexpensively represent function evaluations of complex physical systems. To be effective, ensuring good computational efficiency and predictive ability is imperative before using kriging as a surrogate in engineering applications. This requires prudent selections of model structure (including kernels) and samples. Kernel selections—which are important to establish the input-output relation of the system—often require domain expertise, which in most cases are not transferable to other applications. Sample selection has to be done such that the sample points can adequately span the design space, and yet not too many that the computational cost becomes prohibitive. The distribution of samples also affects the well-posedness of

kriging model structure. In addition, kriging construction and usage suffer from the *curse of dimensionality* with high-dimensional problems, thereby limiting its widespread applications. To address this issue, researchers have sought to integrate various dimensionality reduction strategies into the model structure of kriging. While this approach can make kriging scalable to higher-dimensional problems, the model accuracy is often sacrificed.

In this thesis, I introduce new strategies to address the three challenges mentioned above. Towards improving the predictive ability and computational efficiency of kriging-based models in high-dimensional problems, I develop the kriging with joint mutual information maximization (KJMIM). Its structure-preserving property proves superior than other dimensionality reduction techniques in ensuring high model accuracy. Multiple kernels, instead of a single one, are used to add more flexibility in fitting complex profiles using kriging-based models. To this end, I develop mixed kernel learning (MiKL) and multidimensional composite kernel learning (MCKL) that can automatically avoid non-performing kernels and select the optimum kernel combinations and hyperparameters. A systematic study on the relationship between multimodality of the function and kernel selection in high-dimensional problems is also presented in this thesis. Lastly, I develop an active learning strategy with multiple stopping criteria to improve the robustness of sample selection. In particular, the likelihood information and expected improvement are exploited to select the most informative points to be added to the sample set. In this thesis, I also demonstrate that the combination of the proposed active learning strategy and effective kernel selections can notably improve the predictive capability of kriging on complex problems. The works presented in this thesis contribute to enhancing the efficacy and practicality of kriging-based models in real-world engineering applications. The exposition on methodology, results, and observation is intended to provide more intuition and insight into the effects of kriging structure and parameters on its performance.

CHAPTER 1

INTRODUCTION

This research aims to develop efficient and scalable surrogate-based computational models for complex problems. We use the surrogate modeling approach which involves creating a representative model of models to effectively capture the characteristics of such problems. To be specific, this thesis will address the efficient modeling of high dimensional problems and selection of effective sample points for kriging-based models. Deep intuition into kernel selections will be given, and new methods to help engineers select the appropriate kernels will be proposed in this thesis. Lastly, multimodality in complex problems, which causes serious issues during modeling will be investigated and extensively discussed. Kernels that are suitable to avoid this problem will also be highlighted. This chapter presents the motivation and objectives of this research, followed by the original contributions and the organization of the thesis.

1.1 Background and Motivation

Real-world problems are often complex. This is because the problems are usually obtained from systems with many simple components which may interact with each other, from whom a collective behaviour with interacting dynamical properties emerges. They are seen in different sectors such as in the aerospace and energy industries [277]. Due to the many interacting components, the data obtained from such systems are often high-dimensional. These systems are considered to be highly dimensional if they have more than fifteen features. Modeling these systems usually come with different levels of complexity [242] and demand more computationally expensive simulations and sometimes expensive physical experiments [171]. Another major challenge to effectively modeling these systems could be the paucity of data, due to the limited recorded data [101]. As a result, methods such as the artificial neural networks (ANN) [270] become non-applicable.

This is because ANN and other deep learning-based methods would usually require large number of samples (> 1000) to be able to sufficiently train a model.

Other complexities may include function discontinuity and high non-linearities. Understanding the complex underlying relationships between the input factors of the system with as few data points as possible can be achieved using inexpensive approximating models known as surrogate models. Surrogate models [181] are developed as a solution to capture the complexity in these problems with lesser computational requirements. For instance, black-box surrogate models are used to approximate a function based on a set of training points, which are then used to estimate function values at new points. Engineering optimization relies on this benefit to perform inexpensive optimization that would have been computationally challenging without high-performance computing [195]. Surrogate models have wide applications in several disciplines and they provide means to allow engineering design exploration with high-fidelity computer simulations. Some of their common applications are (1) design optimization and exploration; modeling the relationship of the input variables, the objective and constraint functions, in which the exact relationship is difficult or even impossible to obtain, (2) in uncertainty quantification; where the goal is to quantify the uncertainty in the output as a function of the random inputs, and (3) in sensitivity analysis; with the aim of quantifying the contribution of each random variable to the output variation, taking into account the interaction between variables. Also, surrogate models have been extensively used in structural reliability analysis for the computation of the probability of failure by classifying the problem space into a feasible and infeasible region [195].

There are different types of surrogate models. The detailed classification will be provided in Chapter 2. This thesis focuses on data-based surrogate models, a major class of surrogate modeling techniques. They are also known as black-box methods. Major types of data-based surrogate models used in engineering communities include support vector machine (SVM) [110, 194], radial basis function (RBF) [115, 238], and kriging [51, 52]. An important machine learning technique developed by Vapnik and his colleagues in the late 1980s is support vector machines (SVM) [181]. This is a method rooted in statistical learning theory. It is an algorithm that exploits the Vapnik-Chervonenkis (VC) theory's dimension and its ability to learn from data [49]. The building block of SVM is the idea

of structural risk minimization. The essence of this idea is the control of the complexity of the metamodel with the overall goal of preventing overfitting [250, 251]. Regardless of the fact that the method was originally developed to tackle binary classification problems, it has since evolved to handle multi-class and regression tasks [232]. Domingos opined that SVMs are the most commonly adopted kernel-based frameworks for classification problems [59].

Radial basis functions (RBFs) [36, 99] consist of linear combinations of basis functions. Each basis function relies on the distance from the assigned prediction point to the associated training point. We solve a dense system of linear equations to obtain the respective coefficients of the functions that make up the linear combination. To capture broad patterns, RBFs are frequently supplemented with polynomial functions [115]. RBFs have demonstrated a high level of accuracy in the characterization of many problems. In addition, they take less training time for small datasets. Although RBFs have numerous advantages, studies from literature show them to be be inadequate interpolants, especially with respect to energy minimization. This is particularly pronounced in cases with widely distributed data points [115].

Kriging is another commonly used surrogate modeling method. The method was initially created for *geostatistical* modeling purposes [44]. The first formal mathematical formulation was presented by Matheron [166] and it has since gained widespread use in computational modeling and design applications [213]. When dealing with blackbox functions, kriging gives an approximate relation between the variables concerned, namely input, $\mathbf{x}$, and output, $\mathbf{y}$ [51]. The idea behind kriging surrogates is to consider the output $\mathbf{y}(\mathbf{x})$ as a realization of a stochastic process $\mathbf{Y}(\mathbf{x})$ [218]. When kriging is used to model a function, the mean and variance of prediction can be obtained on a given dataset. The provision of the output variance which can be used to estimate uncertainty has made kriging one of the go-to methods in Bayesian optimization and uncertainty quantification applications [224, 197].

The incorporation of kernels within kriging formulations makes it a flexible predictor [208]. The kernel functions may be designed for various data representations [275]. In addition, The kernel flexibility makes kriging to be easily adapted to solve more problems [45]. For instance, homoscedastic noise in some real-world data can be captured by

adding a noise variance parameter to the correlation matrix [75]. It is also possible to incorporate expert knowledge, for instance via the cokriging formulation [76], or trend functions [16]. Incorporating the gradient information into kriging can help improve model accuracy and reduce overall uncertainty in prediction [153]. The hyperparameter values in kriging's covariance function could be use to identify important variables. Hence, kriging can be used for automatic relevance determination (ARD) [265].

It is worthy to mention that the use of kernels has always been related to prior knowledge of the correlations in the input information. In most cases, such as in *engineering design optimization*, where it could be difficult to have prior knowledge, certain kernels are usually used. In engineering applications, the Gaussian kernel function is selected primarily. The Gaussian kernel assumes that the response function is smooth, however, this assumption is not always true, especially for real-world processes [241]. Instead, the use of Matérn class covariance functions has been recommended. More discussions on kernels will presented in Section 2.3.1.

The application of kriging is majorly plagued by $O(N_s^3)$ computational complexity incurred during model training. This is due to the inversion of an $N_s \times N_s$ correlation matrix, where N_s is the number of data points. This inversion is carried out multiple times during hyperparameter estimation. Therefore, a large data set may become costly and difficult to train. Huang *et al.* [114] reported that sometimes the kriging metamodel could be more expensive than running the original simulation model, especially when more samples are added. However, sparse kriging [233], which is also known as cluster kriging [257], may also be used to address this issue. This method entails using a carefully selected subset of all data points to train data points. In general, kriging modeling suffers from the *curse of dimensionality*, where it performs poorly for high dimensional decision spaces. This may be attributed to large quantities of data being the prerequisite to developing a representative model. For this method, distances also fall short of the appropriate representation of the proximity of data points in higher dimensions [5]. High dimensional decision spaces may necessitate some dimensionality reduction method [247]. With the availability of sufficient data, there is also a possibility of numerical problems with regards to matrix inversions or decomposition during model training. The covariance matrix may become ill-conditioned, resulting in inaccurate parameter estimation or predic-

tion. This might occur when the data points are very dense in certain areas of the function space [45]. With large data, there are fairly likely two design points that are spatially close to each other, and thus the two corresponding columns in the covariance matrix are close to being linearly dependent. Another major issue is with kernel selections. While the kernels introduce a great level of flexibility into the kriging model, making a definitive decision on the most effective kernel function to use needs domain expertise. Unsurprisingly, designing a robust framework that cuts across different problem domains poses a significant challenge. Furthermore, discontinuities encountered along the profile of the objective function profile continues to be a difficult problem for kriging-based models. This drawback of kriging models results from the fact that standard stationary kernels may not approximate the objective function to the degree that it becomes useful for applications such as efficient global optimization (EGO) [234]. Regardless, this class of kernels commonly provide satisfactory results when dealing with smooth profiles, provided there is sufficient training data. In the end, while kriging hyperparameter values could be used for ARD, it has been difficult to estimate the kriging hyperparameter directly from the relationship between the input and output variables. If this can be done, the computational time it takes to train a model can be greatly reduced.

From literature, different methods have been used to improve kriging models. However, the efforts made to improve the performance of kriging models will be futile if the kernel is not selected properly. This is because the kernel functions remains the heart of kriging models. Selecting the appropriate kernel for a particular problem is a non-trivial task as using an unadapted kernel might harm model performance [91]. Hence, it is desirable to be able select the kernel for kriging model with little efforts. This thesis focuses on providing more intuition to help non-experts understand and select effective kernels for their various applications. Other issues associated with kriging will also be addressed in this thesis. The aim and contributions from this thesis are highlighted in Section 1.2.

1.2 Original Contributions

In this thesis, the key objective is to provide more guidelines towards effective kernel selections and improve the overall performance of kriging models. In particular, we aim

to gain a fundamental understanding on how the assumptions, model structure selection, and hyperparameter derivation in kriging construction affect the effectiveness of kriging model. For this purpose, scalable and efficient kriging-based models will be developed to overcome some of the limitations of kriging mentioned in Section 1.1. To be specific, significant improvements are made in the areas of kernel selection, model reduction and adaptive sample selection. The study presented in this thesis will help guide users, mostly engineers, to choose the right surrogate model for the specific problem at hand, with less dependence on trials and errors or *ad hoc* solutions. The original contributions presented in this thesis are briefly described below.

Show the impact of using multiple kernels on model performance

Selecting the appropriate kernel for a particular application is non-trivial. The use of multiple kernels has been proposed to address this issue. From this research, it was identified that Gaussian kernel performs worse for pre-training applications because performance of the model developed with the kernel is largely dependent on the initial values during hyperparameter optimization. With multiple kernel models, this issue can be solved. The limitations of different kernels can be corrected by simultaneously using multiple kernels. Lastly, assigning different kernels to different variables of the data during model training was found to improve model performance, albeit at the expense of the computational training time. Hence, it is recommended that multiple kernel strategies be used only for low-dimensional problems.

Show the effect of multimodality on model performance

While multimodality has been highlighted in several literature as a hindrance to effective modeling of some complex problems, there has been no clear explanation as to why multimodality affects model performance. In this research, the reason behind this phenomenon is exposed. From various experiments, it becomes plausible that multimodality becomes an issue if it reflects in the likelihood profile. The investigation presented in this thesis shows that a multimodal likelihood profile tends to adversely affect model performance.

Improve modeling accuracy with Gaussian kernels

The Gaussian kernel is the most used kernel in surrogate modeling techniques. Some

advanced methods such as kriging with partial least squares (KPLS) only use the Gaussian kernel within their formulations. However, multimodality is a major issue with the Gaussian kernel, as will be shown in Chapter 4. Because of this, the model performance becomes dependent on the choice of starting point during hyperparameter estimation. Different ways to circumvent this limitation are given in this thesis. Having multiple starting points can help avoid this issue. This can be achieved by running model training in parallel. Another solution proposed to solve this issue is to use the general Matérn kernel and simultaneously optimize the ν parameter with the hyperparameter. The ν upper bound is limited to a small value of about 3.0. After optimization, the optimal hyperparameter value is used to initialize another model's training process, with the Gaussian kernel used. Significant improvement in model performance can be achieved with either of the proposed methods.

Provide clear explanation on kernel selection in both low and high-dimensional problems

While proven strategies can be deployed to improve the modeling accuracy of the Gaussian kernel when used to model complex problems, it is only effective for low-dimensional problems. In Chapter 5, I will show that the Gaussian kernel is not necessarily better than other kernels in terms of modeling accuracy, especially when sufficient training samples are used. Hence, it becomes important to prudently select the right kernel for a specific application, which might not be the "default" Gaussian kernel. This is because of the great effect of multimodality on the Gaussian kernels. Traces of multimodality are also seen in the likelihood profiles of models trained with the Matérn 5/2. For this reason, the use of the exponential or the Matérn 3/2 kernels is recommended for modeling complex problems.

Derive effective hyperparameter approximation with information theory

Model training for high-dimensional problems can be difficult due to the number of parameters to be trained simultaneously. There is also the issue of longer training time. This limits the application of such model to real-time applications where "instant" training and prediction are required. To alleviate the anticipated problems, several methods have been proposed in literature to attempt to approximate hyperparameter of surrogate modeling techniques such as kriging. However, there has

been significant loss in model performance with the recorded reduction in model training time. In this thesis, the reason for such losses are explained. For instance, the KPLS method which uses partial least squares (PLS) to attempt to reduce the hyperparameter search space could lose as much as 50% model accuracy when compared with the baseline model. The loss is attributed to a significant change in the model structure. In this thesis, attempts are made to preserve the model structure of kriging by using knowledge gained from information theory. The joint mutual information maximization technique is used to approximate the kriging trend before hyperparameter optimization. By using this, the kriging model structure is preserved and hence, the model accuracy is also preserved while the computational training time is greatly reduced.

Develop efficient stopping criteria for active learning strategies

Insufficient sample points can lead to a poor predictive capability of the model. Intuitively, a user might want to add more sample points to improve the model performance. But in doing so, the question of "how many more points will be sufficient?" needs to be answered. In practice, some researchers and engineers keep increasing the number of points and keep trying it out on a set of defined test points to ensure that they get the model quality desired. In this thesis, it is shown that increasing the number sample points does not always improve the model accuracy, and might worsen model performance in some cases. Hence, it is important to add new points *adaptively*, that is, where needed. In modest cases, users set the maximum number of data points as the stopping criterion. Once this maximum number of points is obtained, the process is terminated and the data points are used for model training. In other cases, researchers use some defined criterion to help stop the learning process of the adaptive sampling algorithm. This criterion is usually case-specific and requires expert knowledge to obtain the desired results. When the number of points is less than required, it might affect model performance. In the case where more than necessary points are added, it might cause some numerical issues. In this thesis, the multimodality characteristics of the Gaussian kernel are exploited and used to help define a clear and robust stopping criterion for active learning strategies. To be specific, it is observed that it is sufficient to stop the learning process when the

likelihood profile of the problem is “multimodal-free”.

1.3 Organization of the thesis

Figure 1.1 illustrates the structure of this thesis, showing how the chapters are related to each other. Chapter 2 provides a detailed explanation on complex problems and the characteristics of such problems. After the explanation on complex problems, the chapter provides detailed explanation on surrogate modeling and its use in solving complex problems. Detailed background on Gaussian processes is also provided. Chapter 3 presents the basic issues which could arise when a single kernel is used for modeling a problem. The two common multiple kernel methods proposed in literature to tackle the misspecification tendencies of single kernels are introduced. We then explore the potentials of using different kernels and kernel combinations across the problem dimensions. The results from the benchmark of models trained with single kernels, methods from literature and the proposed methods are discussed. The assessment of the behaviour of the proposed methods in different cases is provided. In Chapter 4, a study on the sensitivity of kernels to hyperparameter values is presented. From the study, two methods proposed to overcome the limitations identified in the kernel study are explained. The benchmark results are also discussed. While the strategies introduced in Chapter 4 are best suited for low-dimensional problems, Chapter 5 gives detailed explanation on kernel selections for high-dimensional problems. An information theory-based and kriging-based proposed model to reduce model computational training cost without altering the model structure is discussed. In Chapter 6, an effective stopping criteria for active learning strategies is proposed. The benchmark results are also discussed. This thesis ends with summary and conclusion of this work, and is consolidated with some proposed future work in Chapter 7.

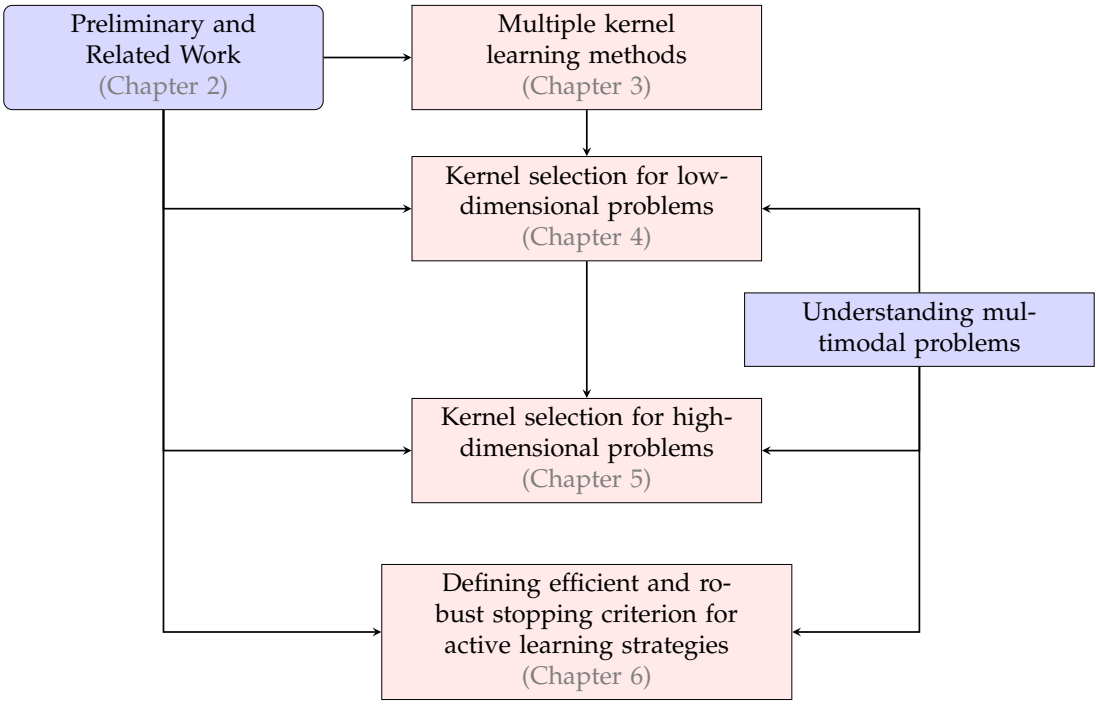


Figure 1.1: Flowchart showing the interconnection of the chapters in this thesis.

CHAPTER 2

PRELIMINARIES AND RELATED WORK

This chapter begins with explanation on definition and characteristics of complex problems. Different methods used to model such complex problems are explained. A general overview of surrogate models and kriging as a Gaussian process is given. On Gaussian processes, the fundamental formulations are presented along with a detailed discussion on the various types of covariance functions. Next, hyperparameter estimation via optimization is explained. Lastly, a comprehensive review on the different ways that have been proposed to improve the modeling capability of kriging is presented.

2.1 Complex problems

The applicability of conventional techniques such as deterministic simulations become limited when they are used with complex problems. Solving such problems sometimes require the development of novel methods. This is because they usually require more computationally expensive simulations and sometimes expensive physical experiments [171]. Many real-world problems fall within this category, including those in aerospace engineering. In fact, the complexity that comes with aircraft aerodynamic designs has been on the increase in recent times [272].

2.1.1 Characteristics of complex problems

Complex problems have certain defining characteristics. It is important to note that some problems classified as complex problems may not have all of these characteristics. Some of the major characteristics of complex problems are described below.

Non-linearity

One of the prominent attributes of complex problems seen in many real-world problem is non-linearity [187]. Non-linearity property defines the indirect relationship

between independent variables and dependent variables. This property is often considered to be essential for complexity. Non-linearity makes modeling challenging, and it is commonly encountered in the context of aerospace engineering, such as when we wish to optimize aerodynamic systems. The non-linearity in this case is observed in the objective functions and constraints [272]. Therefore, should the outcomes of data analysis consist exclusively of linear methods, it is highly likely that predictions may not be accurate [201].

With the establishment of the criticality of non-linear methods, it is also essential to point out that there are cases where linear methods will be sufficient. The main reason for the preference of linear techniques in these cases is their explicit dependence on the tested and trusted efficiency associated with the simplicity of matrix theory and linear algebra. We must however reiterate the fact that the operation of a system over an extensive range of conditions, as is common with a significant number of real-world problems, requires kernel methods. This is because they provide the only adequate means of handling the non-linear dynamic behavior that results from their physical and mathematical representation. Given a dataset, a scatter plot can be simply used to identify non-linearity. Another approach that can be used is the application and validation of a simple linear regression [266, 175] on the dataset. If the model performs poorly on that dataset, it is rather likely that such data are non-linear.

High-dimensionality

With the tremendous growth in technology, it has been easier to capture different types of data. Hence, the data capable of completely defining many contemporary real-world systems typically have many features, i.e., multidimensional. These features usually interact with each other and with the response of the system. When a particular dataset has so many features, it is considered to be high-dimensional. High-dimensionality is a major challenge towards effective modeling of systems [123]. This is because such problems tend to require more samples before they can be adequately modeled. With a large number of sample points, it becomes more computationally expensive to model such data. In fact, the *curse of dimensionality* is considered as one of the biggest current challenges in surrogate modeling development.

Multimodality

Multimodality is the presence of several modes or peaks in a particular data. This situation is very common in many real-world problems. It is not uncommon for high-dimensional data to be multimodal. Multimodality brings about uncertainty in modeling [221]. Using a modeling technique which is not able to reflect this multimodality would lead to a sub-optimal model. Hence, a technique capable of capturing the inherent multimodality in data would be of great value. It could also be very difficult to know before hand the presence of multimodal points in a particular dataset.

Discontinuity

When a data has several distinct profiles, it is said to be discontinuous. Discontinuities in the function space may impose difficulties in modeling [45]. A set of connected discontinuity points indicates a boundary [13]. This makes the behaviour of the parts at the different sides of the boundary to have distinct behaviour. As such, using conventional modeling techniques to model a dataset with discontinuity could lead to poor models. Several methods have been introduced to handle discontinuity and generally improve modeling performance. Most of these methods focus on identifying the boundaries, assign different models to the distinct profiles and efficiently combine the predictions made from different parts.

Data Paucity

Real-world problems such as those seen in the complete design of aircraft systems require computational processes that are computationally expensive. For instance, the use of computational fluid dynamics (CFD) or finite element analysis (FEA) in evaluating the aerodynamic coefficients and wing stresses for a high-fidelity aircraft model might take days or even weeks. As such, it becomes difficult to readily obtain much data from such experiments. With the paucity in data, some techniques are usually employed to augment the data. However, there is a potential issue of bias transfer. The bias in the original data is easily transferred to the artificial data. Hence, some techniques like deep learning cannot be used to accurately model such problems. With black-box surrogate models, a dataset with few points might still be used to build a model with reasonable predictive ability.

Noisy data

Most data, especially those from real-world measurements, are subjected to a degree of randomness considered as *noise*, which is often meaningless. The noise in data can adversely affect modeling capacity if not handled properly. There are several common sources of noise. The noise in measured data could be considered as a result of defect in the measuring device. Mistakes during data collection could also lead to noise in data. In dealing with this type of noise, the variance is usually known. When the noise variance is not known, it is important to estimate its magnitude and make necessary allowance in the modeling process. Unknown noise variance can either be homoscedastic [200] or heteroscedastic [19]. Homoscedastic noise have constant variance across the dataset. They can be estimated during the hyperparameter optimization process. Heteroscedastic noise occur when the noise variance differs from one point to another. Because of this nature, they are more difficult to handle.

2.1.2 Methods used to solve complex problems

In the early days, simple mathematical formulations were sufficient for solving engineering problems derived from relatively simple geometries and their corresponding loading conditions. A good example is the design of a compound pulley system for construction works. In some cases, physical experiments were designed and performed in the laboratory to support the designs of complex problems. The situation changed with advances in engineering. The problems from the real-world now involve cases with complex shapes, boundary conditions and material behaviors. Hence, it is insufficient to rely only on classical analytical closed-form methods to solve such problems. With physical experiments, there is the issue of costs. It could take several months to years to set up a fully functional laboratory. Even after setting up a fully functional aerospace laboratory, for instance, the parts must be tested thoroughly before being released for distribution. When this is not adequately done, several lives could be at risk. There is also the increased hazard tendencies for people who work in such laboratories. Because of these limitations, methods such as finite element analysis (FEA) [222, 244], computational fluid dynamics (CFD) [139, 248] and molecular dynamics (MD) [206, 111] were developed. These numerical-based meth-

ods use computer approximations to solve problems which often have no analytical solution. With finite element analysis, it is possible to predict problems or failures in products before they even exist. For instance, FEA can be used for aircraft steel analysis and stress testing designs, thereby eliminating the need for random assumptions. It also becomes easy to spot issues that engineers might fail to anticipate. With the use of FEA in design process, companies are able to fast-track the release of products as well as have a significant saving on material and labour costs.

With the various advantages of using versatile computation-based numerical methods such as FEA for solving the complexities that are prominent in real-world problems, there are two major issues which should be addressed. These include the associated margin of error and the computational expense of running heavy simulations. These significantly limit the application of methods such as FEA. The errors could be modeling, discretization or numerical in nature.

Modeling errors are usually due to the over-simplification of the real-problem. This could lead to significant loss in model accuracy. The errors from discretization occur due to inappropriate meshing of the geometry. This error is usually difficult to estimate [212]. Numerical errors arise due to numerical instabilities from the approach used. Regardless of the possibility of errors, finite difference techniques such as FEA are not suitable with noisy or discontinuous models [167].

The computational cost of using these methods with high-fidelity solution can be exhaustive. Sometimes, it might take several days to run analysis on a particular model. With the issues with errors and computational cost highlighted, it is important to mention that errors in computational models can be adequately managed with experience from an expert. With more practice, engineers learn to avoid any form of errors in the modeling process. Likewise, the time it takes to run highly-detailed analysis can be greatly reduced by running the process in parallel with cluster computing such as high performance computing (HPC). Setting up and managing HPC resources has relatively high cost implications. As such, it is important to be able to use cheaper modeling techniques such as surrogate models [161, 121] for solving complex problems.

2.2 Surrogate modeling

Surrogate modeling entails the use of mathematical models to derive simpler approximations of physical systems. The resulting approximations thus require less computational cost for subsequent analysis and optimization [235, 229]. Conceptually, the outcome of the surrogate modeling process (also known as a surrogate model) may be adopted as low-cost substitutes of typically costly evaluations of physics-based models. This is especially suitable for computationally intensive problems [90]. Surrogate models may also be called metamodels [230, 181] or models of models [122, 169]. Surrogate models can either be non-physics or physics-based. In physics-based models, the governing laws of nature are embedded into the learning process. These methods are best-suited in learning from data about large-scale complex systems [263]. The physics-based techniques are further divided into multifidelity and reduced-order modeling techniques [7].

Multifidelity models

The multifidelity models are also known as variable-fidelity or hierarchical models [210, 211, 92]. With multifidelity models, a physics-based low fidelity model is used as the surrogate in place of the high-fidelity model [66]. The derivation of the low-fidelity models depends on the problem. The low-fidelity model can be obtained by using simpler engineering models with simplified physics [10, 246] or even using the same high-fidelity models with considerable modification. The necessary modifications can be achieved either by significantly increasing the residual tolerance [75] or using a model with coarser grid [33, 140].

Reduced-order models (ROM)

A typical reduced order model is a simplified representation of a complex, high-fidelity model. ROMs makes it possible to analyse the behavior of the original model, and thus be able to quickly extract useful information pertaining to its dominant effects, with as little computational resources as possible. ROMs use numerical methods to approximate the relationships between inputs and outputs of a system. They do this primarily by reducing the order of the system under consideration. These approximations are obtained by having the original model projected onto a basis domiciled in a lower dimension space [145]. The key objective of developing

these ROMs is twofold. The first is the provision of system dynamics representations that are quantitatively accurate with an associated computational expense that is significantly less than what may be required to process the original numerical model. The second is the provision of a mechanism by which a complex system can be quickly interpreted. [157]. Antoulas [20] gives further insights on ROMs, their development and adoption. The computation of the basis of the reduced space can be done with different methods, some of which are proper orthogonal decomposition [118], Krylov-subspace methods [102], and approximate balanced truncation [239, 103].

Non-physics based techniques are known as *data fit* surrogate modeling methods. Data fit techniques are also known as *black-box* models. With black-box models, the derivations depend only on the input and output data of a system. They are more suited for applications where the physics of the system is not known. They are preferred in modeling of very complex systems due to their non-intrusive nature [145]. For this reason, only data fit surrogate models are considered in this thesis.

2.2.1 Data fit surrogate models

This kind of surrogate models are further categorized based on the related parameters, the number of data points and the derivation method adopted. Figure 2.1 shows the different types of surrogates models based on their classifications.

Adopting the number of data points used for model development, we can classify black-box surrogate models into local, multipoint, and global models [128]. Local models approximate functions, using a single data point as the focus. A classic example of this is the Taylor approximation with its intervening variables. The validity of these approximations is limited to the vicinity of the point that defines the function's locality. Multipoint models on the other hand use multiple points (typically two) in the model development process. Examples of this include two-point exponential approximation (TPEA) [69] and two-point adaptive nonlinear approximation (TANA) [258]. Ultimately, global surrogate models build their approximation models, using the entire input space. As a result, they are more accurate than the foregoing types of surrogate. Their higher accuracy makes

them more popular with the modeling community. Polynomial regression [243], multivariate adaptive regression splines (MARS) [77], support vector machines (SVM) [37], artificial neural networks (ANN) [113, 267], kriging [213, 51, 132], and RBF [168, 162] are a few examples of models that fall within this category.

Data fit models can be divided into interpolation- or regression-based categories depending on the derivation method [66]. In interpolation-based methods, the approximated function should pass through all the points in the sample set. Interpolation involves finding the line of best fit such that that line goes through *all* the training data [259]. It is assumed that the points in the dataset represent the values of a function accurately. Hence, interpolation-based methods are not suited for noisy data. These model types are best suited for deterministic computer experiments. A computer experiment is said to be deterministic if the outcome of multiple runs of the same input settings are exactly the same [156]. Regression-based models, on the other hand, attempts to fit a function without it having to go through the previously known points. Regression models turn out to be more suitable for function approximations where random error is inherent. A typical case of this is experimental data [145].

Depending on how the approximation functions represent the input-output relations, a surrogate model may be parametric or nonparametric [128, 145]. Once the model parameters or unknown coefficients are derived, a parametric model no longer utilizes the training data. A polynomial regression model is a good example of a parametric model. Although the models in this category are quite straightforward, they are not very useful when the actual input-output relationship is complex. By definition, a nonparametric model continues to make predictions at new points using the data from the training sample. However, nonparametric models can still have tunable parameters, often known as *hyperparameters*.

The different classification of data fit surrogate models are summarized by Liem [145]. Figure 2.1 shows the taxonomy of data fit surrogate models as well as some examples of the different categories.

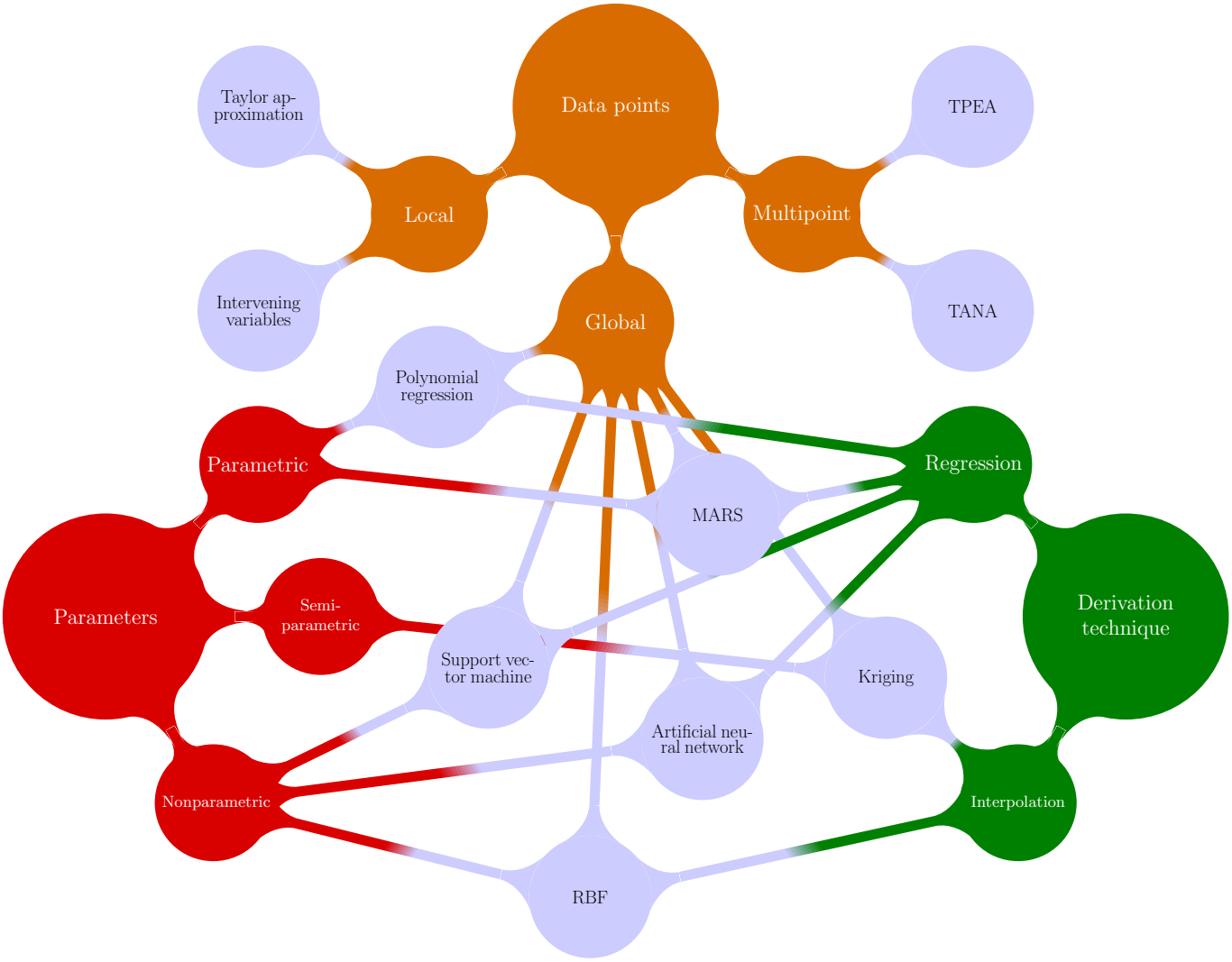


Figure 2.1: Taxonomy of data fit surrogate models [145].

2.2.2 Steps involved in surrogate-aided engineering design

Demands for better designs across different industries such as the aerospace industry has brought about increasing complexity. An example is seen in current aircraft designs. Aircraft design is known to be highly multidisciplinary because they involve many interrelated disciplines. Multi-objectivity as well as non-linearity of the objective functions and constraints have also posed a major challenge in design. There is also the issue of uncertainty quantification in aerodynamic problems. These complexities pose a great challenge to designers and most times require well-detailed models to capture their intricacies in the design. Evaluating these ever-growing models could be expensive given the restrained

computational budgets [272]. As a result of these challenges, fast and cheaper methods known as surrogate-based approaches were introduced and widely adopted to reduce the computational cost of expensive models, while preserving the model accuracy. The process of utilizing surrogate models in engineering design is known as *surrogate-aided engineering design*. The target of this process is to use surrogate models in place of the expensive computations in a complex engineering system. This practice is capable of saving computational cost and time. However, it is important that the surrogate model is trusted to handle the process before it is used as a reliable surrogate. Hence, it is essential to ensure that the surrogate model is carefully chosen and properly trained. Many researchers have drawn up approaches for using surrogate models in engineering design. Most of the approaches focus on the use of surrogate model within optimization framework. As such, when the prescribed criteria are not met, more points are added *adaptively* and the model is re-trained. This approach could be limiting. It assumes that the surrogate model used and the corresponding parameters (and hyperparameters, if applicable) are well selected. However, this is not always the case. Sometimes, replacing the kernel might be the change needed to obtain a better surrogate model. In Figure 2.2, we present a flowchart showing the extensive process of using surrogate models in engineering design. The selection of sample points for model training and the validation of the trained model are key steps in surrogate-aided design. Sample selection and model validation techniques are explained in Sections2.2.3 and 2.2.4 respectively.

2.2.3 Selecting sample points for model training

Without any prior information, experiments start as a *trial and error* method, where attempts are made to find the best guesses or sample points locations. These methods do not guarantee achieving the goal of locating the best points. Due to this limitation, the one factor/variable at a time (OFAT [54] or OVAT) approach is commonly adopted. However, these methods do not take into account the interaction between the variables (also known as factors). The substantial number of runs needed is also disadvantageous as it can be too expensive. Fisher *et al.* [73] introduced the design of experiment (DoE) in physical experiments to control noise and bias [42, 174] in addition to overcoming the limitations highlighted above. The DoE method was initially used to gain insights into the probabilis-

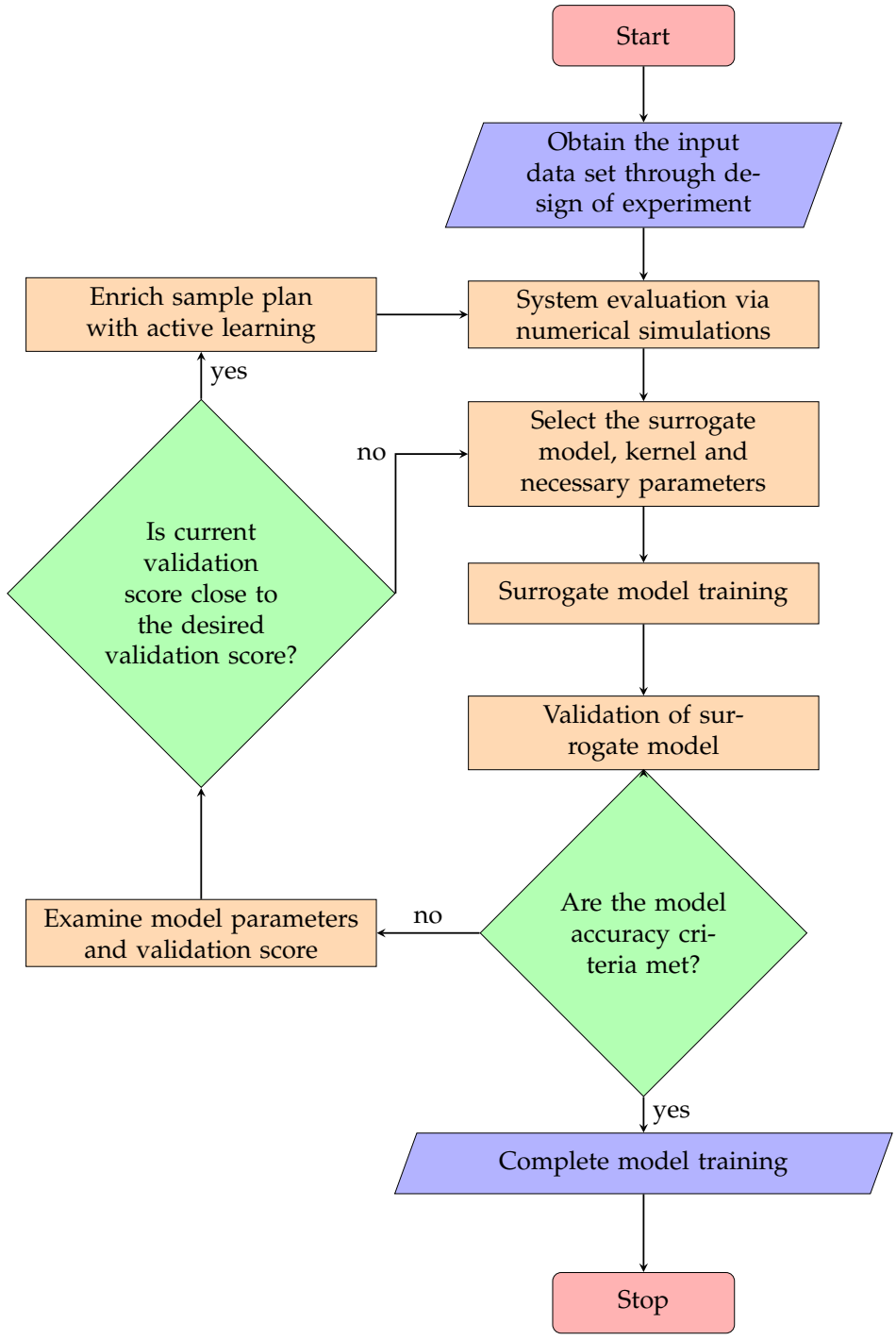


Figure 2.2: Flowchart showing the different stages involved in effective use of surrogate modeling techniques in engineering design.

tic behaviour of agricultural crops systems. The advent of computational facilities and numerical approaches led to the birth of computer-based simulations. These simulations have gradually been used to replace physical experiments, whenever possible. One major difference in the two experimental methods is that computer-based experiments are deterministic, hence, easily reproducible, unlike the physical experiments that are known to be stochastic in nature [174]. Interested readers are referred to reference [131] for more information on the comparisons between the two methods. Computer-based DoE methods are sometimes referred to as *sampling techniques*. These techniques are used to optimally select a set of data points with the aim of maximizing the information gained from a limited sample points. These sample points are key components of the data fit surrogate modeling methods. The selection of these points can greatly affect the performance of surrogate models. The techniques used in selecting sample points are categorized into two major groups, which are classical [15, 155, 130, 174] and modern methods [213, 132, 42, 71, 39]. For more information on the adoption of design of experiment methods in applications such as in aerospace industries, readers are referred to the references [93, 84].

The modern sampling techniques are known as design and analysis of computer experiments (DACE). Also, sampling techniques can either be domain- or response-based. *Domain-based sampling techniques* are strategies used to draw sampling points without information about the model. They are also known as one-shot, offline or single-stage strategies. These techniques are known as one-shot methods because the sample points are generated just once [276]. One-shot design of experiment (DoE) [73, 272, 203, 231, 86] techniques such as Latin Hypercube Sampling [47], Halton sequence [105, 262] and Sobol' [237, 236] were devised to draw sample points such that they are evenly distributed in the design space. The methods attempt to allocate sample points throughout the design space with minimal holes and bias. This makes them space-filling techniques [218, 71, 203]. With these techniques, improved model performance is expected to be achieved by increasing the number of samples drawn. However, this is not always the case. This approach towards sampling assumes that *one-size-fits-all* for different problems. However the model prediction quality is highly dependent on both the size and distribution of the given training points. The needed training point distribution is dependent on the function profile. As a way to ensure that the appropriate samples with the required distribution is used for model training, response-based methods were developed.

In *response-based methods*, the selections are made with the model in active consideration. These methods are also known as online, model-based or sequential techniques. The information supplied to the techniques aim at improving their efficiencies. In several complex applications such as in aerospace applications, it is impossible to achieve a given model accuracy through the selection of sample points *a priori*. Generally, space-filling approaches are used to generate the initial sample points. More sample points are then added adaptively at promising locations within the specified domains. This approach is known as adaptive sampling. *Adaptive sampling*, also known as *active learning approaches*, were introduced to identify informative locations for additional observations [159]. In general, adaptive sampling techniques can help reduce the number of samples as much as possible, while generating a proficient surrogate model [79]. The general procedure for one-shot and adaptive sampling is shown in Figure 2.3.

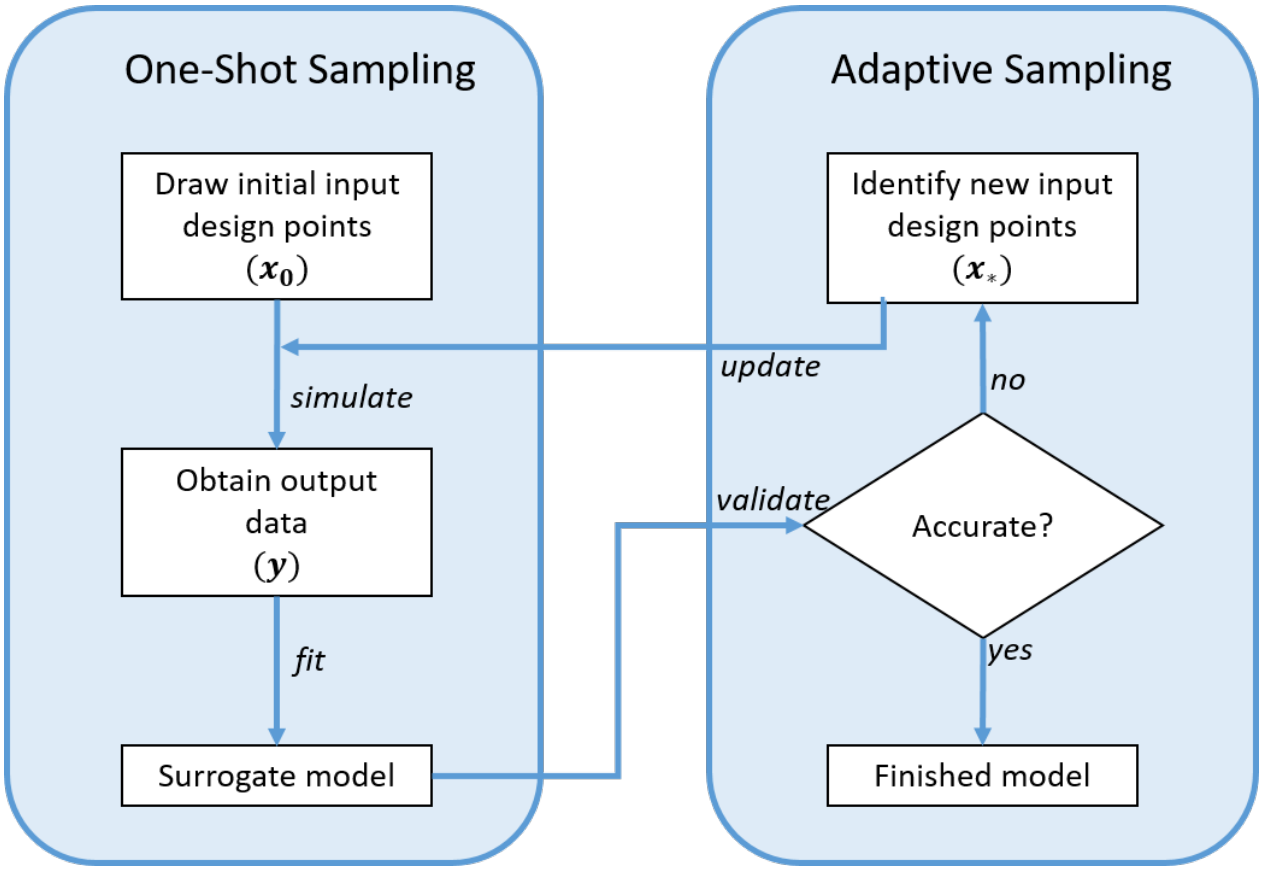


Figure 2.3: General procedure for the one-shot and adaptive sampling techniques (adapted from McBride and Sundmacher [167]).

2.2.4 Assessing model performance

The process of quantifying the fidelity of a model is known as *model validation*. This is usually carried out after the development of surrogate models. It is important to quantify the fidelity of models for two major reasons. First, model validation helps to ensure that the developed model is suitable to be used as surrogates in computationally-intensive analyses and optimizations. Second, the model validation metric values obtained can be used as the basis for comparison with other models.

Model validation methods are broadly categorized into two, namely (i) methods that utilize existing data, and (ii) those that require additional data [169]. For methods that require additional data, a test set—which is different from the training set to ensure the unbiasedness of the validation results—is usually needed. This process can be expensive and, as such, might not be practically suitable for many surrogate modeling applications [171]. The two major categories of methods used for model validation can either be global or local estimation techniques [95]. The performance of a surrogate model considered across the whole problem space is deduced using global measures. On the other hand, point-wise and local measures give the accuracy of the surrogate model in different parts of the problem space. These two methods are briefly described below.

Model independent global measures Popular approaches to quantifying model independent global error measures include cross-validation, likelihood estimate and Akaike's information criterion (AIC) method [204].

A technique that is widely used to estimate the error associated with a surrogate model, without extra system evaluation is cross-validation. When adopting a q-fold cross-validation, the data set is randomly divided into q (approximately) equal data subsets. The surrogate model is subsequently constructed q times, while leaving out one of the data subsets from the training each time. The data subset that is left out during each iteration, is used in the evaluation of the cross-validation error [252]. The k-fold cross-validation technique is a variation of the q-fold technique described above. In this case, all the possible data subsets, each of size k are used in the evaluation of the cross-validation error. On the other hand, we may also consider the leave-one-out cross-validation approach ($k = 1$). For this approach, the training

set is generated at each iteration, using all sample points except one. The left-out point is then used to evaluate the error incurred by the surrogate model prediction vis-à-vis the actual value.

The likelihood estimation method is commonly used in statistical communities to assess and compare model performance. A major downside of using this method is the bias in the lack of fit. This is corrected by the AIC method. In the AIC method, the surrogate model's performance is obtained using a technique that capitalizes on penalizing the likelihood function. The AIC is derived from the sum of a penalty term and the negative log likelihood. The equation is presented in Table 2.1. In the AIC equation, parameter $L(\theta)$ represents the maximized likelihood function, and N_f captures the number of free parameters associated with the model. N_f is designated as a measure of capturing system complexity. It is also a compensation for the bias in the lack of fit when the maximum likelihood estimators are utilized [32].

Model independent local measure The linear reference model (LRM) [188] is a good example of a local validation estimation method [171]. LRM is a model-independent method for measuring the local performance of a surrogate model. In LRM, regions with many fluctuations are labelled as high-error areas. In this method, the deviations of surrogate model from the local linear interpolant is used to estimate the errors of the surrogate model in the design domain [188]. This method is applicable for surrogate models such as kriging [171] because in kriging, the correlation between two points is related to the distance between them. Hence, the errors at the different points in the design domain are not independent. When the distance between the two points is smaller, the correlation tends to approach one, and when the distance increases, the correlation tends to approach zero. Using this correlation strategy, there is a higher confidence for prediction on test points closer to the sample points. The local error measuring approach for the kriging predictor at the particular point x^* reflects this idea. At the sample points, LRM error equals zero and is equal to the variance of approximation error in the stochastic process.

Model dependent measures While it is relatively cheaper to use model-independent measures to assess the performance of surrogates, there are two major limiting issues, namely bias and model overfitting. The model-independent methods tend to be

more biased toward over-represented regions. This is because only the training data are used for the assessment of model performance. There is also the issue of over-fitting. It becomes impossible to know whether the model being assessed is being overfitted. Hence, a model validation technique that has proven to be more reliable uses additional test points in the computation of approximation errors [256].

The root mean square error (RMSE) is widely used as an evaluation metric in dealing with regression problems. It follows an assumption that errors are unbiased and follow a normal distribution. The *squared* nature of this metric helps deliver more robust results which prevent positive and negative error values cancelling each other off. Also, the RMSE metric tends to heavily penalize deviations because of the power of *square root*. With large samples, reconstructing the error distribution using RMSE is considered to be more reliable. However, RMSE is highly affected by outliers. Therefore, it is advised that the users remove the outliers from the test data before using the metric. This, however, is not always possible. There is also the issue of lack of intuition. The RMSE value gives little to no intuition on the performance of a model. It could also be difficult to use the metric for comparison with other models.

In order to have more intuition from the validation metric, the normalized root mean square error (NRMSE) is used. The NRMSE is also used when the function value varies widely in the input space of interest. In the normalized RMSE, each error component, $(y_i - \hat{y}_i)$, is normalized with respect to its actual value, y_i, before the RMSE is computed, to give the relative approximation error. RMSE and NRMSE provide good estimates of the global error over the region of interest. Simpson *et al.* [231] recommended to choose a model with a low RMSE, as it reflected a good overall model accuracy. However, it is impossible to provide simple guidelines to specify a maximum allowable RMSE value that applies to all problems.

When there is significant presence of outliers, the coefficient of determination, R^2 is used as the model metric. The R^2 is a statistical measure of how close the data are to the fitted regression line. The R^2 score for a model is intuitive and easily understood. For example, the surrogate model is more accurate if R^2 is closer to one. This metric makes it easier to compare model performance and define acceptable model performance values.

Figure 2.4 shows the classification of model validation methods. The formulations of some commonly used metrics used to validate the performance of surrogate models are given in Table 2.1.

Table 2.1: Various metrics used to evaluate the performance of surrogate models

Validation metric	Notation	Expression
Akaike's information criterion	AIC	$AIC = -2\ln(L(\theta)) + 2N_f$
Root mean square error	RMSE	$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$
Normalized root mean square error	NRMSE	$100 * \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2}$
Coefficient of determination	R^2	$1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$

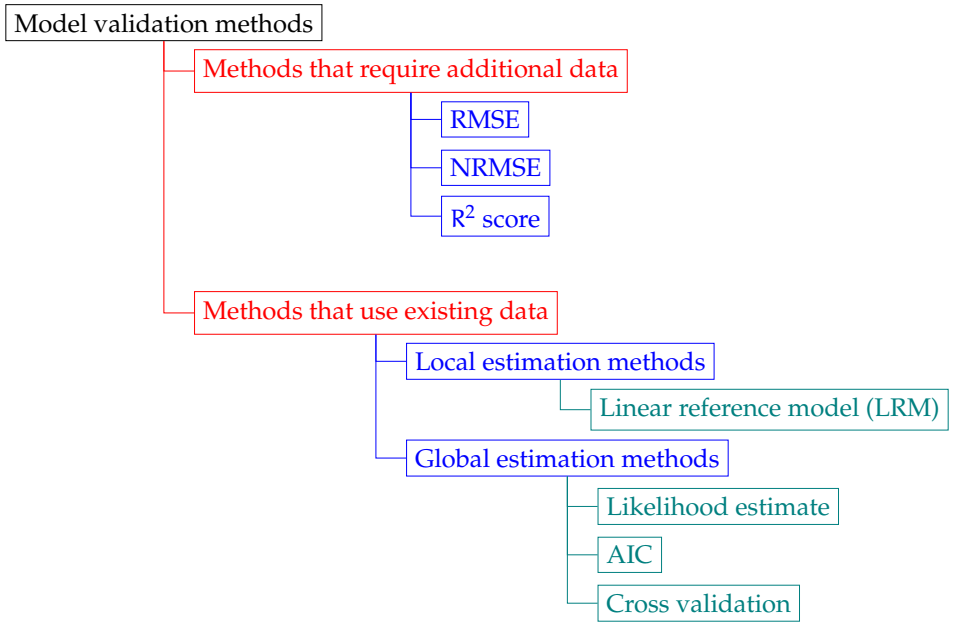


Figure 2.4: Chart showing the classification of model validation methods.

2.3 Gaussian Processes

Gaussian processes (GPs) are more flexible surrogate models that are highly suitable for capturing complex nonlinear responses [191]. They are nonparametric estimation methods which give simple probabilistic representation of random processes. GP models can be subdivided into two, namely regression and classification models [208]. For our research, we are only interested in *kriging*, which is a special case of Gaussian process regression (GPR). Kriging is an interpolation technique, hence, the errors are zero at the sample locations [75]. The use of kriging extends many disciplines which includes engineering design optimization, uncertainty quantification [64, 192, 182, 55] and global sensitivity analysis [185, 23]. The use of kriging for engineering optimizations include mainly both aerospace [119, 149, 284, 144, 137] and structural applications [214]. In this thesis, we use kriging and Gaussian process interchangeably to refer to interpolation method case of GPR.

The function input, x, of a blackbox function and the corresponding output, y, in kriging methods are related by the expression

$$Y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}), \tag{2.1}$$

where the first term $\mu(\mathbf{x})$ is the global model. The global model is the deterministic function approximating the mean trend of the output. The second term of Equation 2.1, $(Z(\mathbf{x}))$ is the stochastic term, which has a zero mean and the covariance:

$$\mathrm{Cov}\left[Z\left(\mathbf{x}\right), Z\left(\mathbf{x}'\right)\right] = \sigma^2 \mathbf{R}\left(\mathbf{x}, \mathbf{x}'\right), \tag{2.2}$$

where σ^2 is the process variance and $\mathbf{R}\left(\cdot,\cdot\right)$ is the correlation matrix.

The global models in kriging are similar to the global polynomials used for RBFs [115]. When a known mean is used, the model is known as *simple kriging* model. Ordinary and universal kriging are other variants of kriging based on the global model. When the global model has a constant but unknown mean, it is known as *ordinary kriging*. On the other hand, kriging with general polynomial mean, as its global model, is known as *universal kriging* [280]. Universal kriging is also known as "kriging with a trend" [143, 81]. The trend is defined by regression functions. The universal kriging model can be difficult to build when the relevant trend is not known. When there is *a priori* knowledge about the

evolution of the output variable, elaborated regression functions are used [189]. In some cases, physical laws can provide some assumption about the output variable [129]. When this *a priori* knowledge is not known, it is sufficient to use a constant mean and build an ordinary kriging model [213, 228].

The stochastic part is considered the fundamental part of a kriging model [191]. It is modeled as a Gaussian process where a specified covariance function also known as kernel is used for the spatial correlation matrix. Kernel selection for the correlation matrix is an important factor to be considered in the construction of kriging models. This is because the kernel captures the assumptions about the dependence structure of the function of interest. When dealing with higher-dimensional problems, the correlation matrix is usually constructed by employing the product correlation rule. Here, the correlation function is expressed as a product of stationary, one-dimensional correlations. This correlation between samples i and j in the dataset can be subsequently expressed as

$$R_{ij}\left(\mathbf{h}, \theta\right) = \prod_{k=1}^{N_d} \kappa\left(h_{ij}^{(d)}, \theta^{(d)}\right), \tag{2.3}$$

where the value $h_{ij}^{(d)}$ is the distance between two points in the d^{th} dimension, $\left|x_i^{(d)} - x_j^{(d)}\right|$. The vector of correlation parameters is denoted as $\theta = \left\{\theta^{(d)}\right\}$ for $d = 1, \ldots N_d$. This value shows the strength of correlation in each direction [143]. Details into the estimation of this value are given in Section 2.3.2.

2.3.1 Covariance functions

Key to the model structures of kriging are the kernels [101]. Kernels can either be stationary or non-stationary [209]. Our focus is on stationary kernels which can be expressed as a function of the difference between their inputs (distance-based). This type of kernels has two components, namely the distance and the hyperparameter θ. The distance is a measure of the separation of points $|x - x'|$. In most cases, kriging-based models adopt predefined kernels in their construction. In literature, there are large number of possible kernels, with their respective free hyperparameters, which need to be evaluated to define the model structure. The Gaussian kernel [208] is the most widely used because of

its simplicity and flexibility [191]. It is also infinitely differentiable which facilitates its adoption in applications such as gradient-enhanced kriging (GEK) [283]. In spite of the wide adoption of the Gaussian kernel, it is also known to assume the smoothness of the function, which limits its usage in many problems, even for some computer-based experiments [191]. As a result, the Matérn kernels were introduced for modeling real-world processes [241]. The Matérn 3/2 and Matérn 5/2 kernels are the commonly used in the Matérn family. The exponential kernel is a special case of the Matérn class known to be mean square continuous but not differentiable. The Gaussian, Matérn 5/2 and Matérn 3/2 are infinitely, twice- and one-time differentiable respectively [208]. This thesis considers the following four kernels—the Gaussian, Exponential, Matérn 3/2 and Matérn 5/2 kernels. The kernels are used for developing the various models. The covariance functions, $\kappa(h, \theta)$ of the four kernels are given below;

- Gaussian

$$\kappa(h, \theta) = \exp -\frac{1}{2}\left(\frac{h}{\theta}\right)^2 \tag{2.4}$$

- Exponential

$$\kappa(h, \theta) = \exp -\left(\frac{h}{\theta}\right) \tag{2.5}$$

- Matérn 3/2

$$\kappa(h, \theta) = \left(1 + \frac{\sqrt{3}|h|}{\theta}\right) exp\left(-\frac{\sqrt{3}|h|}{\theta}\right) \tag{2.6}$$

- Matérn 5/2

$$\kappa(h, \theta) = \left(1 + \frac{\sqrt{5}|h|}{\theta} + \frac{5h^2}{3\theta^2}\right) exp\left(-\frac{\sqrt{5}|h|}{\theta}\right) \tag{2.7}$$

where, h is $|x - x'|$ and θ is the length scale.

2.3.2 Hyperparameter Estimation

Hyperparameters are essential parts of the kernel formulation. They refer to a number of parameters of the model structures. These hyperparameters are also called *length-scales*. They are a measure of decay rate of correlation [101]. If the length-scale has a

very large value, the covariance will become almost independent of that input variable. A low value of θ is a reflection of low distance weight. Intuitively, this indicates a strong correlation between samples. When θ increases, the correlation between samples weakens, but when the value gets really large, the correlation matrix (R) becomes singular. In the study of kriging-based models, correlation functions may be either anisotropic or isotropic. Isotropic correlation functions are those in which a single θ value is adopted across multiple dimensions [208]. This culminates in the simplification of the estimate of the the maximum likelihood. Thus, it is treated as a function of a single lengthscale. On the other hand, anisotropic correlation functions are preferred in cases where each variable is associated with a specifically defined physical quantity [108]. In these models, the variable dimensions have unique θ values. Furthermore, this approach provides more modeling flexibility, albeit at an added cost, namely, a more complex maximum likelihood estimate [53, 154].

The subject of selecting an appropriate range of values for the lengthscale in kriging models is still a challenge in the surrogate modeling community. Readers are invited to consult the paper by Gramacy [100] for more information.

In the estimation of hyperparameters, cross-validation and maximum likelihood estimation (MLE) are the methods of choice for most researchers [164, 208, 22, 193]. However, the MLE method is preferred due to its computational efficiency [83]. On the other hand, the CVE method is inclined to give less biased results, hence its inevitability in certain applications [61]. A brief overview of the MLE method is given below;

Maximum Likelihood Estimation

The maximum likelihood estimation approach is based on the likelihood function which can be expressed as;

$$L(\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}|R(\theta)|^{1/2}} \exp\left[-\frac{(y-1\mu)^T R(\theta)^{-1}(y-1\mu)}{2\sigma^2}\right] \tag{2.8}$$

The values for the maximum likelihood estimates of the coefficient of regression (μ), and variance ($\hat{\sigma}^2$) are obtained from the derivative of the natural logarithm of the likelihood

function and setting the result to zero.

$$\mu = (\mathbf{1}^T R^{-1} \mathbf{1})^{-1} (\mathbf{1}^T R^{-1} \mathbf{y}) \tag{2.9}$$

and,

$$\hat{\sigma}^2 = \frac{1}{N_s} (\mathbf{y} - \mathbf{1}\mu)^T R(\theta)^{-1} (\mathbf{y} - \mathbf{1}\mu) \tag{2.10}$$

Substituting Equations 2.9 and 2.10 into the natural logarithm of the likelihood function of Equation 2.8, we obtain the concentrated log-likelihood function as:

$$LL(\theta) = -\frac{N_s}{2} \ln(2\pi) - \frac{N_s}{2} \ln \hat{\sigma} - \frac{1}{2} \ln |\mathbf{R}|, \tag{2.11}$$

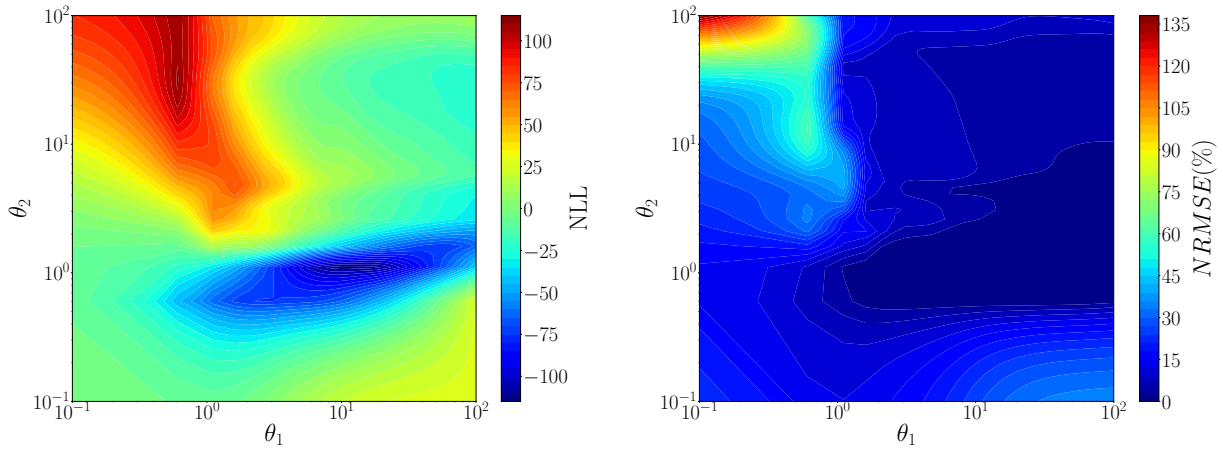
as a function of the set of hyperparameters θ. θ is optimized by maximizing the function. We can obtain the negative likelihood estimate and minimize the resulting equation instead. In essence, the optimization task becomes obtaining the set of hyperparameter values that minimizes the negative likelihood estimate (NLL). It is important that this process is done as thoroughly as possible. This is because the likelihood estimate affects the accuracy of a model. To demonstrate this, the Branin function is used.

The Branin or Branin-Hoo [75] function is an analytical function in two dimensions. It is non-convex and continuous within its domain of definition. It is usually evaluated in the domain: $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. Analytically, it is given by Equation 2.12.

$$f(x_1, x_2) = \left(x_2 - \frac{5.1 x_1^2}{4\pi^2} + \frac{5 x_1}{\pi} - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos(x_1) + 10 \tag{2.12}$$

With a Gaussian covariance function, a range [10^{-1},10^2] on both hyperparameter axes, a training point of 40 and a 200 $\times$ 200 test grid, we show the contour plots of the negative likelihood estimate and the NRMSE in Figure 2.5.

From Figure 2.5a, we observe the various regions within the likelihood plot. It is also evident that the region having the least negative likelihood coincides with the region having the least model error in Figure 2.5 b. From this, it is clear that the likelihood has a positive correlation with model accuracy. Hence, when training kriging models, it will be desirable to obtain better model likelihood. We can also observe from Figure 2.5b that to get a model with significantly high accuracy, we only need to ensure that we have the hyperparameter values within the region surrounding the maximum achievable likelihood.



(a) Likelihood contour plot

(b) Model accuracy contour plot

Figure 2.5: Plots showing the relationship between likelihood estimate and model accuracy.

2.3.3 Hyperparameter optimization

Hyperparameter optimization [273] is the process undertaken in obtaining the best hyperparameters that effectively captures a search space of interest. While dealing with optimization, it is essential to choose the optimization technique that suits the problem under consideration. For an in-depth description of optimization techniques, readers are encouraged to check the book on engineering design optimization by Martins and Ning [165]. Within the context of kriging-based models, hyperparameters values are chosen after the optimization of a defined loss function, e.g., cross validation error or maximum likelihood estimate. In the selection of these hyperparameters for kriging and other surrogate modeling techniques, any optimization algorithm can be used. The non-negativity constraint is enforced such that the hyperparameter values are positive everywhere.

Pattern search algorithms such as Nelder-Mead algorithm [158, 28] are widely used for the optimization of hyperparameter values of surrogate models. Nelder-Mead method (also known as downhill simplex method) is a local direct search deterministic algorithm. It was first proposed by John Nelder and Roger Mead for non-linear function optimization applications [186]. It is an algorithm that is capable of adjusting its structure to the objective function's landscape. The algorithm reduces the function search space and facilitates easy convergence. However, it is very sensitive to the initial guess. An undesirable feature of this technique is its tendency to converge to a local optimum. Besides, the algorithm does not guarantee convergence to a stationary point [173]. For the definite Nelder-Mead

algorithm adopted in most parts of this thesis, we use the `constNMPy` package [28] because it has features that allow us specify nonlinear inequality constraints and variable bounds.

2.4 Methods to improve the performance of kriging models

Over the years, many contributions have been made towards improving the performance of kriging models. The improvements will be summarized based on the two parts of the kriging formulation; the global and stochastic terms. In this section, we give an extensive review of those methods to the best of our knowledge.

2.4.1 Global term

As mentioned in Section 2.3, there are three main types of kriging based on the global model. Because of its flexibility, major contributions have been made in universal kriging. Joseph *et al.* attempted to estimate the polynomial function of the global model by using data analytic procedures [126]. Kersaudy *et al.* [129] and Schobi *et al.* [219] proposed the use of polynomial chaos expansions in the approximation of the global term of universal kriging model. Yang *et al.* attempted to integrate physical knowledge into kriging [269]. The authors used Monte Carlo (MC) in construction of a kriging model. To be precise, multilevel Monte Carlo (MLMC) was used to approximate both mean and covariance of a kriging model.

2.4.2 Stochastic term

The computational complexity of the kriging model is more pronounced in the stochastic term. This is because the inversion of the covariance matrix in the kriging formulation is of a cubic order [104], which is computationally expensive. Because of this, traditional kriging models are not scalable with a large number of sample points. This limits kriging usage to low dimensional problems, since high dimensional problems tend to require a large number of sample points, which could be computationally challenging. Some of

the major areas in which improvement has been made to the stochastic term of kriging formulation are:

High-dimensional modeling

Several works have been done to make kriging scalable at high dimensions. Sakata *et al.* reported a 20% reduction in the computational cost when the Sherman-Morrison-Woodbury formula is used to estimate the inverse of the kriging coefficient matrix [215]. Bouhlel *et al.* combined kriging with partial least squares (PLS) to build kriging models capable of giving promising results on problems with up to 100 dimensions. Also, the joint model known as KPLS was reported to be 450 times faster than conventional kriging [29]. Hensman *et al.* attempted to extend the use of Gaussian processes for big data by introducing the stochastic variational inference into Gaussian process models [116]. Other works on high-dimensional kriging are carried out by Durrande *et al.* [184].

Adaptive sampling

Liu *et al.* proposed the bias-variance decomposition framework to select new points that maximizes the expected prediction error [147]. Eason and Cremaschi showed that combining adaptive and space-filling techniques could reduce the needed sample points by up to 40% when compared to using purely space-filling methods [65]. However, this method may fail to capture the behaviour of the underlying function if there are strong nonlinearities at the bounds of the search space. Also, the approach adds considerable computational cost to the sample selection algorithm. Other works on active learning/adaptive sampling can be found at the references [87, 245, 125, 1].

Incorporation of gradient information

Gradient information has also been used to improve the performance of kriging models [46]. This is known as gradient-enhanced kriging (GEK). When the gradient information are used to argument the correlation matrix, the model becomes a form of cokriging. The gradient information can be obtained by various methods such as adjoint or finite difference methods [56, 75]. GEK is reported to improve the accuracy of kriging models. Towards improving this method, Zuhal *et al.* incorporated

sparse polynomial chaos expansion (PCE) as the trend function within the formulations of gradient-enhanced kriging [284]. The authors reported an improvement in the performance of the kriging models. Even though the method outperforms conventional GEK, it has more hyperparameters to train, hence disadvantageous with increase in dimensions. In the same vein, Bouhlel and Martins combined partial least squares method to gradient-enhanced kriging to control the size of the correlation matrix while maintaining accuracy of the model [31]. This made GEK useable with problems up to 100 dimensions. Common to all gradient-based kriging methods is that they do not scale well with increase in sampling points because the size of the correlation matrix increases from $N_s \times N_s$ to $N_s(N_d + 1) \times N_s(N_d + 1)$. More contributions can be accessed in the following references [153, 249, 107, 138].

Provision for stochastic simulations

Poor handling of uncertainties that arise in stochastic simulations might lead to the development of poor kriging models [18]. As such, stochastic kriging (SK) [17] was proposed to combine generalized least squares regression with kriging. The method models the response surface as a realization of a Gaussian random field (GRF). Staum [240] and Ankenman *et al.* [18] reported that the method enables better prediction especially in instances where uncertainty is non-negligible, as seen in stochastic simulations. However, the experiments were mainly performed on one-dimensional problems. As such, more experiments are needed to ascertain the claims by the authors. The authors also reported that the GRF model developed with the Gaussian kernel did not work well for approximating both the prediction mean and variance. Perhaps, using other kernels might be worth exploring.

Towards improving the performance of SK, Chen *et al.* proposed incorporating gradient information into SK models [43]. The authors reported that using gradients obtained from infinitesimal perturbation analysis (IPA) is best suited for combination with SK to avoid degradation in prediction performance. This past research also suffered from limited experiments as only two experiments were carried out. More information on stochastic kriging can be accessed in the paper by Zou and Zhang [281].

Mixture of experts

The presence of distinct function profiles can hurt the accuracy of kriging models. As such, mixture of experts (MoE) method was proposed to model problems with heterogeneous function profiles [117]. Liem *et al.* proposed the use of different gradient-enhanced kriging models for the distinct function profiles [144]. The authors used Gaussian mixture model (GMM) to cluster the problem space. Their method was limited in that users need to specify the numbers of clusters, which is not a trivial task. Rasmussen and Ghahramani introduced infinite mixture of Gaussian processes to eliminate the need to specify the number of experts *a priori* [207]. Meeds and Osindero proposed an alternative approach to the infinite mixture of Gaussian processes approach, to allow dealing with incomplete data [170]. Bettebghor *et al.* used an MoE approach as a surrogate for a discontinuous problem domain in a structural optimization problem [25]. More details about mixture of experts, see the comprehensive survey by Yuksel *et al.* [274]

Kernel selection

All the methods discussed above are mainly focused on improving the performances of kriging by proposing different methods leaving out the kernel selections. In most cases, the Gaussian kernel was used in the kriging formulation. The dangers of choosing a kernel blindly is explained in Section 4.3. Different works such as that by Abdessalem *et al.* have been proposed in literature, where they proposed the use of approximate Bayesian computation (ABC) algorithm to help with kernel selection and parameter estimation for machine learning applications [2].

Greedy search for kernel selection

In spite of the extensive literature on automatic selection of kernels, it is important to note that the computational efficiency of such a process is also crucial for practical implementations. As such, different multiple kernel methods have also been proposed in literature. The methods aims at providing less rigid and more accurate solutions to real-world problems by attempting to find optimal combination of multiple kernels [68, 21, 97]. Combining models in some way has been shown to improve the approximation performance of the surrogates [26, 274]. Kronberger and Kommenda proposed the use of genetic programming to select the best ker-

nel combination [82]. The authors generated all possible combinations using valid kernel operations such as multiplication and addition before using genetic programming to select the best combination. This approach incurs so much computational cost and as such, is not easily implementable. Duvenaud *et al.* proposed the use of greedy search to locate the best kernel combinations [63]. In the same efforts, Lloyd *et al.* also used greedy search to build new kernels and applied to time-series data [152]. In the three contributions mentioned above, the search of the best combination using greedy search or genetic programming is inefficient and costly. Also, there are potentials for serious issues especially when multiple models need to be constructed.

Multiple kernel models

In literature works on automatic selection and the three greedy search methods discussed above, common to them all is their approach of constructing multiple models based on the same training set, evaluating their performances and selecting the best while discarding the rest. These methods are considered wasteful; as it does not take full advantage of the resources. Also, these methods are based on the assumption that changes in the training set will not impair the performance of the selected model [3].

Goel *et al.* introduced the combination of surrogate models known as an *ensemble*. The authors combined the different models with weights approximated from the error measurements [96]. Acar and Rais-Rohani introduced the combination of prediction of metamodels using optimized weights [3]. Ginsbourger *et al.* in their attempt to reduce the risk of model misspecification, combined the predictions from two kernels using Akaike weights [91]. Palar *et al.* proposed the use of ensemble method in improving the performance of kriging in EGO applications [196]. The authors also explored the use of different methods in computing the weights of the ensembles [197].

Common to all ensemble method discussed above are four stages: (1) training of the individual models with same data; (2) approximating the response of each individual models for the unknown points; (3) estimating the ensemble weight; (4) combining the individual model predictions with their corresponding ensemble weights.

This could be time consuming and hence, limits the use of the ensemble method for applications which require large number of points.

Palar *et al.* proposed the composite kernel learning (CKL) to combine different kernels efficiently; the method combines different kernels using a weight obtained simultaneously with the hyperparameters by maximizing the MLE [193, 191]. The method comes with the assumption that using a single compound kernel across all dimensions is sufficient. In certain cases, this could lead to poor models especially in highly dimensional data for two reasons. First, the negative impact of non-performing kernels becomes more prominent under this circumstance. Second, there is a possibility of misspecification from over-generalization when the variables exhibit different trends.

Liem *et al.* [146] introduced a brute force approach to select the best kernel combinations across variable dimensions. In the method, unique kernel combinations are generated and used within kriging formulations. This method still lacks efficiency and could be time consuming for problems greater than two dimensions. Even though the search for the best kernel-variable combination proved to be exhaustive, the results showed consistent improvement in the accuracy of the kriging model.

2.5 Benchmarking functions

In this thesis, several functions is used for benchmarking purposes. These functions, mostly, algebraic functions are formulated to simulate real-world problems and their complexities. For example, complex real-world data can be highly-dimensional, non-linear and multimodal, the Dixon-Price [279] functions also has the same characteristics. In fact, some algebraic functions such as the robot arm and cantilever beam [67] functions, used in this thesis, directly model common engineering problems. The robot arm function models the position of a robot arm with multiple segments. By using this function, more intuition can be obtained, which can help engineers achieve target goals when designing robots.

There is also the cantilever beam function, which is common in structural applications. The function models a simple uniform cantilever beam with horizontal and vertical loads.

By understanding the use of surrogate models with this function, engineers can learn to improve the structural integrity of their designs.

The benchmarking functions are discussed in details in Appendix A.

2.6 Summary

In this chapter, we defined what could make a problem complex and highlighted the characteristics of such problems. After, we introduced surrogate models and exposed the various categories of surrogate models. Due to the extensive application of surrogate models in engineering designs, we explained the steps involved in integrating surrogate models in design applications, followed by the different ways of drawing samples used for model training during a surrogate-assisted design process. We then explained in details and exposed our intuitions about assessing the performance of surrogate models. After giving sufficient introduction into surrogate models, we shifted the focus to kriging, which is essentially an interpolative Gaussian process. We presented the formulations and explained the processes behind estimating and optimizing their hyperparameters. We also presented the review of the different ways to improve the performance of kriging models as proposed in literature. Lastly, we presented some benchmarking functions that are commonly used in assessing surrogate modeling performance. In particular, we select those that can represent the characteristics of complex real-world problems. These benchmark problems will be used in the subsequent chapters of this thesis.

CHAPTER 3

IMPROVED MODEL PERFORMANCE WITH MULTIPLE KERNELS

In this chapter, two new multiple kernel learning (MKL) strategies are proposed and presented. Two existing state-of-art MKL strategies as well as the newly proposed methods are extensively studied and discussed under different scenarios. The results from the various computational experiments are presented and summary is described.

3.1 Overview of multiple kernel learning methods

Choosing a suitable kernel for a specific application is a non-trivial task [208]. The selection of the kernel in nonparametric models such as kriging is often referred to as a *black art* [63]. The use of unadapted kernels or models can cause model misspecification that sacrifices the result accuracy [91]. As discussed in Section 2.3.1, the adoption of the Matérn kernel in real world cases does not guarantee that it will give better results when compared with the Gaussian kernel in different problems. For instance, a well-known smooth function such as the Branin function might be best modeled with the Gaussian kernel. However, generalizing the performance of the Gaussian kernel based on its good fit with a problem might lead to the development of models with poor predictive capability. Researchers in recent times recommend using multiple-kernel models to improve model performance consistency. In this type of practice, the kernels are usually combined using weights. The assignment of kernel weights is an important research focus in the recent development. Commonly used MKL methods will be further discussed in Section 3.2.

3.2 Existing state-of-the-art methods

The weight combination of several kernels can be done either *intrinsically* of *extrinsically*. In *ensembles* or extrinsically-weighted models, available kernels are used to build separate models, which are then combined according to their assigned weightings to make predictions. On the other hand, *composite kernel models* obtain a "new" single kernel via weighted combinations of existing kernels. The ensemble and composite kernel methods are explained in details in Sections 3.2.1 and 3.2.2, respectively.

3.2.1 Ensemble method

The ensemble method is a well-developed multiple-kernel method used in the EGO community [196]. The method combines the prediction of individual models in an attempt to improve prediction quality. Here, we use a global ensemble to combine the individual models and use the corrected-AIC (AICc) to compute the ensemble weights. Assuming that we have m different kriging models, the general form of ensemble of the models can be expressed with Equation 3.1,

$$\hat{y}_{ens}(x) = \sum_{i=1}^{m} w_i(x)\hat{y}_i(x), \tag{3.1}$$

where $\hat{y}_i(x)$ and $w_i(x)$ are the prediction values and weight of an i-th kriging model. The weights define the contribution of each kriging model to the ensemble function. Before computing the weights, it is important to calculate the number of free parameters, N_f. This can be calculated using Equation 3.2,

$$N_f = \text{len}(\theta) + 2, \tag{3.2}$$

where len(θ) is the number of hyperparameters in the model. After obtaining the value of N_f, the AIC can be estimated using Equation 3.3,

$$AIC = -2\ln(L) + 2N_f, \tag{3.3}$$

where L is the likelihood estimate of the model. Once the AIC for the different models is obtained, the AICc can be computed using Equation 3.4,

$$AIC_c = AIC + \frac{2N_f^2 + 2N_f}{N_s - N_f - 1}. \tag{3.4}$$

The relative AICc difference can then be computed as;

$$\Delta AICc_i = AICc_i - AICc_{min}, \tag{3.5}$$

where $AICc_{min}$ is the value of the model with least $AICc$.

The weights to be used for combining the different models are computed using equation 3.6,

$$w_i = \frac{exp(-0.5\Delta AICc_i)}{\sum_{j=1}^{m} exp(-0.5\Delta AICc_j)}, \tag{3.6}$$

The steps involved when ensemble method is used for modeling the problem are shown in Figure 3.1. For more information about the ensemble method, readers are advised to read the papers by Palar *et al.* [196, 197]

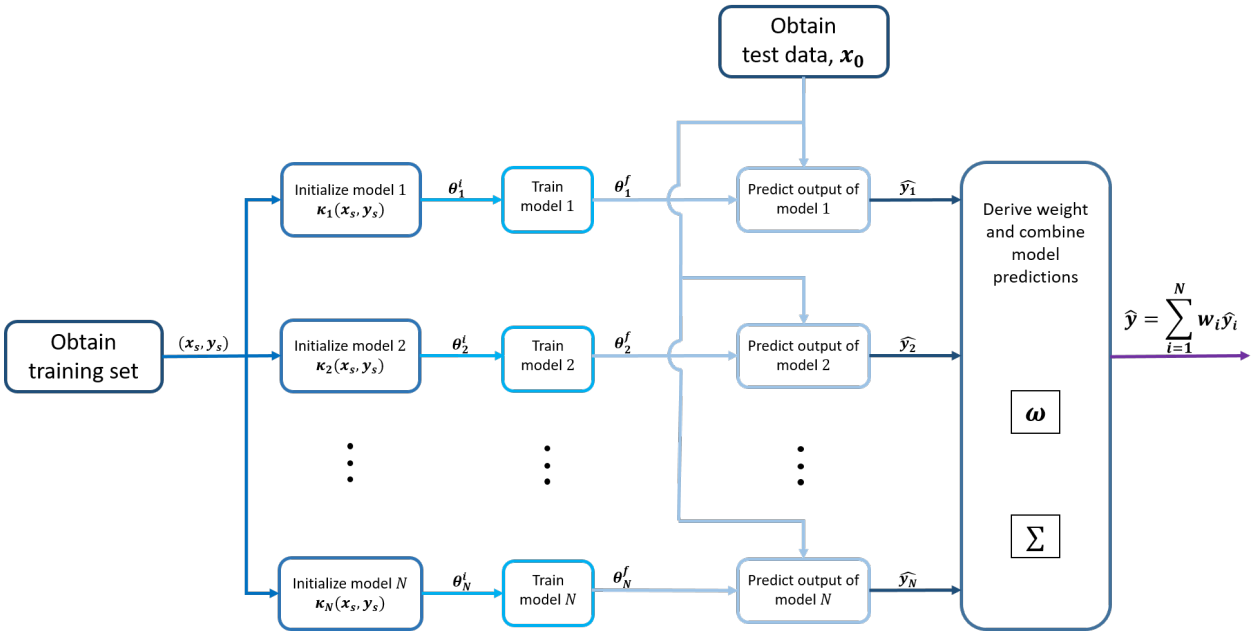


Figure 3.1: Steps involved in ensemble modeling.

3.2.2 Composite kernel learning

The composite kernel learning method was proposed by Palar *et al.* [193] to improve the performance of kriging models. The method attempts to enhance the approximation quality of the kriging model. This is achieved through construction of new kernels by weighted combination of existing kernels. The weights are obtained alongside the kriging hyperparamters by maximizing the likelihood estimate. The weighted sum formulation of combined kernels as follows:

$$R_{CKL} = \sum_{i=1}^{m} w_i(x) R_i, \tag{3.7}$$

where w_i is the non-negative weight vector that should satisfies the equality constraint,

$$\sum_{i=1}^{m} w_i = 1, \tag{3.8}$$

and R_i is the covariance function of the i-th model. Individual kernels' characteristics in the mixture are intuitively regulated by their matching weights. For more information on the method, readers are advised to read the papers by Palar *et al.* [193, 191].

3.3 Proposed Methods

Our goal is to develop methods which will introduce more flexibility to kriging models, automate the kernel selection across variables and most importantly, improve the performance of kriging models for complex systems. In this section, we discuss in details the proposed methods. In our attempt to improve the performance of kriging models, we propose the mixed kernel learning method. Also, we propose the multidimensional composite kernel learning by making a modification to the existing CKL method. It is important to mention that the CKL method simply creates a new kernel for each problem. As oppose to that, our methods are tailored to treat each variable differently. We will also compare the performances of the two methods.

3.3.1 Mixed Kernel Learning

We aim at developing a kriging-based method capable of selecting the best kernel for each variable in a multidimensional problem space. The question becomes how do we do the selection? And what next after selection? Towards solving this problem, we propose a two-stage method summarized in Algorithms 1 and 2. In the first stage, the optimal kernel for each variable is selected. We do this by introducing an $N_d \times m$ weight matrix into the correlation functions where m and N_d are as previously defined. For the weight matrix as shown in Figure 3.2, we introduce a constraint such that the sum of weights on each row is one. In simple terms, the contributions from all the available kernels for each variable must sum up to one. To make obtaining the optimal weight matrix as efficient as possible, we add the constraint to our objective function in Equation 2.11 and also optimize the weights alongside the hyperparameters. The modification is made to Equation 2.11 to give Equation 3.9. In Equation 3.9, the weight matrix is also optimized and this is only done in the initial stage of the MiKL model. After kernel discoveries, Equation 2.11 is then used in the second stage of MiKL model. It is worthy of mention that Equation 3.9 has the negative log likelihood as its objective function. In simple terms, we will still be maximizing the likelihood estimate by minimizing the negative log likelihood. After the optimization, we choose the best kernels for each variable by selecting the kernel with the highest weight for each variable dimension (matrix row). We also store the optimal hyperparameter at this stage. In the second stage, we use the optimal hyperparameters from stage 1 to initialize. This is to help the model converge faster with pre-training information. We then assemble the correlation matrix by using the different kernels obtained in stage 1 for the different dimensions. Afterwards, we train our model again by maximizing the likelihood estimate (Equation 2.11). After training our model, we then use the model for approximations. The second stage is summarized in Algorithm 2.

$$\begin{aligned} \min_{w,\theta} \quad & \frac{N_s}{2}\ln(2\pi) + \frac{N_s}{2}\ln\hat{\sigma}(\boldsymbol{w},\theta) + \frac{1}{2}\ln|R(\boldsymbol{w},\theta)| \\ \text{s.t.} \quad & \sum_{i=1}^{m} w_{di} - 1 \\ & \theta^{(d)} \geqslant 0 \end{aligned} \tag{3.9}$$

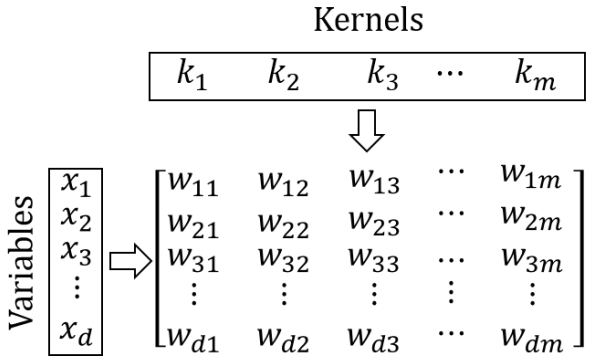


Figure 3.2: Kernel-Variable weight matrix.

Algorithm 1 MiKL: Obtain the optimal kernel-variable combinations

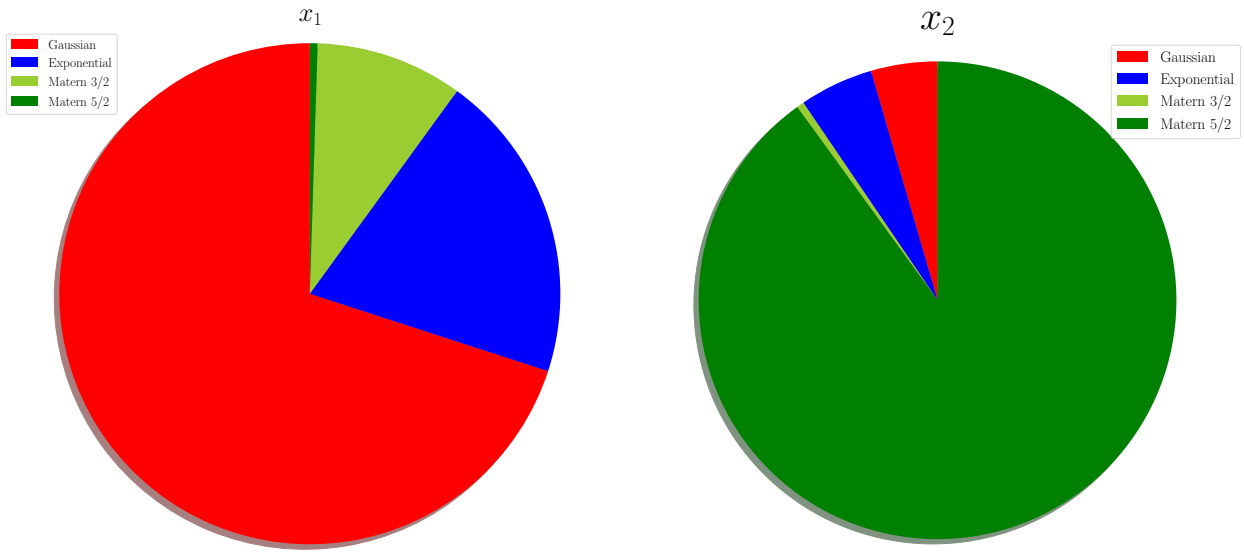
Input: $\mathbf{x}$, y, kernel list $\mathbf{K} = [k_1, k_2, ...k_m]$
Output: θ_1^f, K_{best}
1: Initialize θ_1
2: Initialize $w = \frac{1}{m}$
3: Assemble $\mathbf{R}(\theta_1) = \prod_{d=1}^{N_d} \sum_{i=1}^{m} w_{di} R\left(\theta_1^{(d)}\right)$
4: Obtain $\boldsymbol{w}^f$, ${\theta_1}^f$ by maximizing the likelihood estimate (Equation 3.9)
5: Set K_{best} = []
6: **for all** variable dimension (d) **do**
7: $\quad K_{best,d} = K[\text{argmax}(w^f_{d,:})]$
8: **end for**=0

We provide an example of a hypothetical two-dimensional problem for illustration purposes. After optimizing the weights for four kernels in stage 1, we obtain weights for the first and second variable as shown in Figures 3.3a and 3.3b respectively. The Gaussian kernel will be selected for the first variable because it has the highest weight compared to the other three kernels. In the same way, Matérn 5/2 will be selected as the best kernel for the second variable. Hence, a Gaussian-Matérn 5/2 kernel combination will be used for stage 2.

Algorithm 2 MiKL: Train kriging model and predict

Input: x, y, $\mathbf{X}$, θ_1^f, K_{best}
Output: θ_2^f, $\hat{y}$

1: Initialize $\theta_2 = {\theta_1}^f$
2: Assemble $\mathbf{R}(\theta_2) = \prod_{d=1}^{N_d} R_{kbest}\left(\theta_2^{(d)}\right)$
3: Obtain ${\theta_2}^f$ by maximizing the likelihood estimate (Equation 2.11)
4: Compute the kriging weights $\boldsymbol{\omega} = \mathbf{R}^{-1}(y - \mu)$
5: Estimate $\hat{y} = \mu + r'(\mathbf{X}) \cdot \boldsymbol{\omega}$



(a) First variable (b) Second variable

Figure 3.3: Kernel weight distribution before selection of best combination.

3.3.2 Multidimensional Composite Kernel Learning

We propose a new method to combine the characteristics of composite kernel learning (CKL) and mixed kernel learning (MiKL). It is unlike CKL which uses a single "super kernel" to capture the similarities of the data points across all dimensions. The CKL method shares much similarity with using a single kernel as it assumes homogeneity across the data dimensions. In addition, it is unlike MiKL which uses a unique single kernel for different variables. The proposed multidimensional composite kernel learning (MCKL) method is posed to derive a new kernel for each problem dimension. To achieve this, we introduce an $N_d \times m$ weight matrix into the correlation functions. Just like in stage 1 of the MiKL method, the weight matrix (Figure 3.2) is optimized along with the kriging

hyperparameters. After the optimization, the weights in each row are used to obtain a unique combination of kernels, which is then used in kriging formulation to construct the correlation functions. The method is then used for predicting the responses of supplied inputs. The steps are summarized in Algorithm 3.

Algorithm 3 MCKL

Input: $\mathbf{x}$, y, $\mathbf{X}$, kernel list $\mathbf{K} = [k_1, k_2, ...k_m]$
Output: θ^f, $\hat{y}$

1: Initialize θ
2: Initialize $w = \frac{1}{m}$
3: Assemble $\mathbf{R}(\theta) = \prod_{d=1}^{N_d} \sum_{i=1}^{m} w_{di} R\left(\theta^{(d)}\right)$
4: Obtain $\boldsymbol{w}^f$, θ^f by maximizing the likelihood estimate (Equation 2.11)
5: Assemble $\mathbf{R}(\theta^f) = \prod_{d=1}^{N_d} \sum_{i=1}^{m} w_{di}^f R\left(\theta^{(d)}\right)$
6: Compute the kriging weights $\boldsymbol{\omega} = \mathbf{R}^{-1}(y - \mu)$
7: Estimate $\hat{y} = \mu + r'(\mathbf{X}) \cdot \boldsymbol{\omega}$

3.4 Benchmark with real world data

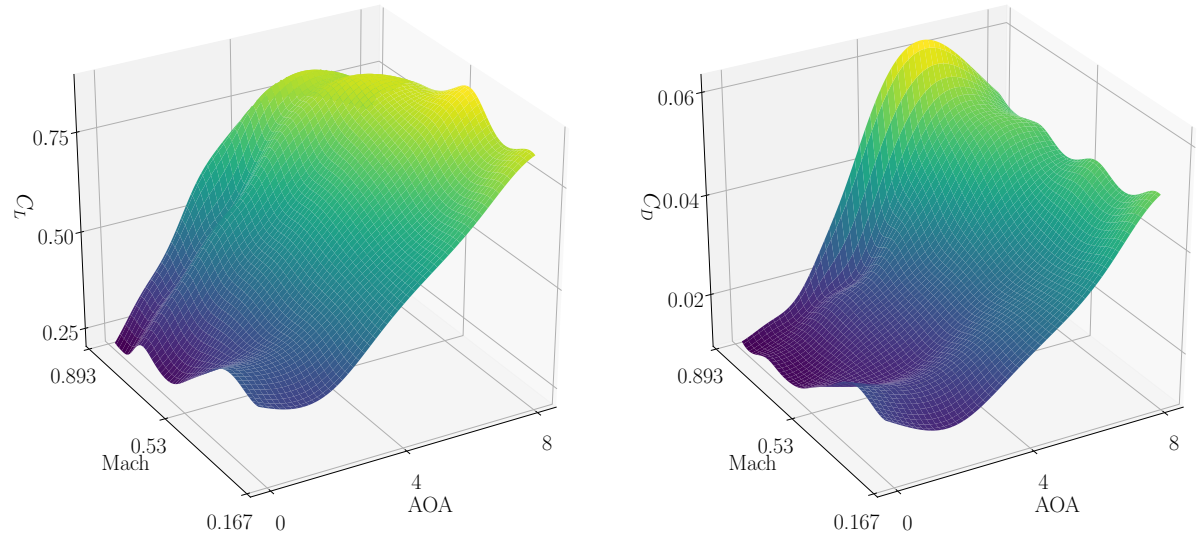
In this section, we assess the performance of the MiKL and MCKL methods on real world problems. Specifically, we use several engineering test cases to benchmark our methods against other methods. The 2D problem sets are explained in Section 3.4.1 and the higher dimensional problems are detailed in Section 3.4.2. We perform comparisons with models built on each of the Gaussian, Exponential, Matérn 3/2 and Matérn 5/2 kernels. We also perform comparisons with CKL and the ensemble methods which are also variants of the multiple kernel kriging. In building the models, the input and output domains are rescaled to $[0, 1]^{N_d}$ before fitting the kriging models. For hyperparameter optimization as well as optimization of the weights in our model, we use sequential least squares programming (SLSQP) [133, 134] implementation in the `SciPy` library. Lastly, we use the NRMSE to assess the quality of the surrogate model. The results and discussions following the benchmarking using the engineering test cases are provided in Section 3.4.3.

3.4.1 Two dimensional cases

For this case, we obtain the aerodynamic coefficients (lift and drag) generated from the Reynolds-averaged Navier–Stokes (RANS) simulations on the SU2 solver by Lyu and Liem [160]. The input data they used were drawn using Halton sequence [105]. In the RANS simulation, the NASA Common Research Model (CRM) aircraft configuration was used. For more details on how the data were generated, readers are advised to read the paper by Lyu and Liem [160]. For the purpose of clarity, we label this dataset as the 2D-CRM dataset. The 2D-CRM data have the angle of attack and Mach number as design variables and the lift and drag coefficients as outputs of interest. The value ranges for the 2D-CRM data are shown in Table 3.1. The approximated response surfaces for the lift and drag coefficients are shown in Figures 3.4 (a) and (b) respectively. In both plots, the x-axis and y-axis are the angle of attack (AOA) and Mach number (Mach) respectively.

Table 3.1: Value ranges for the 2D-CRM input variable.

	Lower limit	Upper limit
Mach number	0.2	0.9
Angle of attack (°)	0	8



(a) Lift Coefficient (b) Drag Coefficient

Figure 3.4: 2D-CRM response surface.

3.4.2 Higher dimensional cases

Here, we explore the performances of our models in 4D, 8D and 16D cases. First, we use a case study problem of a three-dimensional (3D) Common Research Model (CRM) formulated as a 4D problem. For the purpose of clarity, we label the data as 4D-CRM datasets. The problem has the angle of attack, Mach number, altitude and tail angle as design variables. The outputs of interest are the lift, drag and pitching coefficients. This benchmark case is referred to as the mission data problem or 4D-CRM problem. The dataset has 1000 input data with the design space value ranges presented in Table 3.2. For the 8D and 16D test cases, we use an airfoil drag coefficient datasets which has the airfoil shape parameter x_i as input. The drag values are approximated from a computational fluid dynamics experiment. Also, for clarity purpose, we label this dataset as the airfoil shape parameter dataset. Tables 3.3 and 3.4 show the value ranges for inputs of the 8D and 16D cases respectively.

Table 3.2: Value ranges for the mission data input variable.

	Lower limit	Upper limit
Mach number	0.0	0.9
Angle of attack (°)	-9	20
Altitude	10	50000
Tail angle (°)	4	40

Table 3.3: Value ranges for the 8D airfoil shape parameter input variable.

	Lower limit	Upper limit
x_1	3×10^{-3}	1.2×10^{-2}
x_2	-1×10^{-3}	3.5×10^{-3}
x_3	-9.4×10^{-3}	9.7×10^{-4}
x_4	5.1×10^{-4}	5.6×10^{-3}
x_5	-2.5×10^{-3}	4.2×10^{-3}
x_6	-1.3×10^{-3}	3×10^{-3}
x_7	-5.4×10^{-3}	8.5×10^{-3}
x_8	-6×10^{-3}	4.2×10^{-3}

Table 3.4: Value ranges for the 16D airfoil shape parameter input variable.

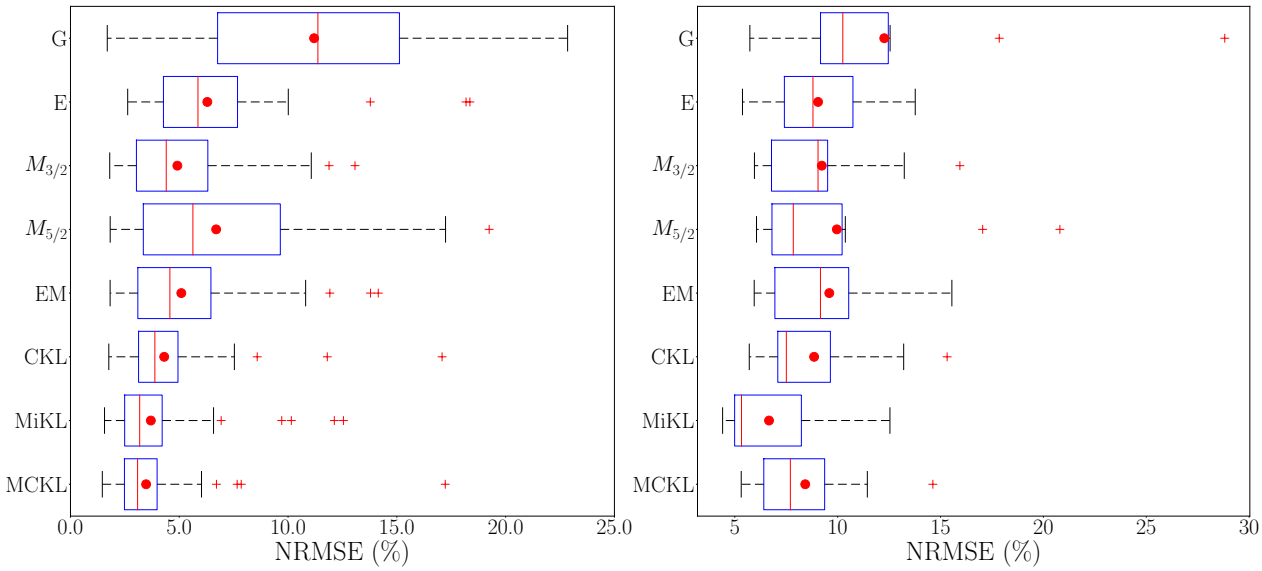
	Lower limit	Upper limit
x_1	4.6×10^{-3}	1.2×10^{-2}
x_2	-1.1×10^{-2}	1.3×10^{-2}
x_3	-1.3×10^{-3}	1.3×10^{-2}
x_4	-2.0×10^{-3}	1.3×10^{-2}
x_5	4.2×10^{-3}	1.3×10^{-2}
x_6	-4.0×10^{-4}	1.3×10^{-2}
x_7	9×10^{-4}	1.3×10^{-2}
x_8	-4.5×10^{-3}	1.3×10^{-2}
x_9	3×10^{-3}	1.2×10^{-2}
x_{10}	-1.4×10^{-3}	1.3×10^{-2}
x_{11}	1.8×10^{-3}	1.3×10^{-2}
x_{12}	-4.2×10^{-3}	1.3×10^{-2}
x_{13}	-6.6×10^{-3}	1.3×10^{-2}
x_{14}	-7.3×10^{-3}	1.3×10^{-2}
x_{15}	-1.5×10^{-3}	1.3×10^{-2}
x_{16}	-4×10^{-3}	1.3×10^{-2}

3.4.3 Results and Discussion

Here, we compare the performance of our methods with both single and multiple kernel methods using the test cases introduced in Sections 3.4.1 and 3.4.2. The kriging models with single Gaussian, exponential, Matérn 3/2 and Matérn 5/2 are represented as G, E, $M_{3/2}$ and $M_{5/2}$ respectively. The ensemble and the composite kernel learning methods are denoted as EM and CKL respectively. This translates to the use of four single and four multiple kernel methods. It is worthy of mention that each of the multiple-kernel methods uses all four single kernels in their formulations. The performance comparisons are visualized with box plots. Lines at the lower quartile (25%), median (50%), and upper quartile (75%) values serve as the boundaries of the box in box plots. At the end of the box on both sides, lines extend at a distance of 1.5 times the inter-quartile range. The outliers are shown by "+" sign beyond the lines. The mean of the data is represented by a red dot, usually inside the box.

For the 2D-CRM problem, we use the leave-one-out cross validation (LOOCV) to evaluate the performances of our models. For each set, we leave one data point out and build

our kriging models on the rest. After this, we evaluate the performance of the models in predicting the response of the left out data. The summaries of results obtained are presented in box plots as shown in Figure 3.5 for both lift and drag coefficients.



(a) Lift coefficient

(b) Drag coefficient

Figure 3.5: 2D-CRM benchmark.

For the results of lift coefficient shown in Figure 3.5a, considering the single kernel based models, the Gaussian kernel performs worst and the best performance is observed with Matérn 3/2 kernel. Considering the multiple-kernel methods, MiKL and MCKL perform best and similarly, with the MCKL slightly better. As shown in Figure 3.5b for the approximation of the drag coefficient, the single kernels here have similar performance with the error in their performance ranging from about 6.0 to 13.5%. Here, the Gaussian kernel yields the poorest accuracy. For multiple kernel methods and in general terms, MiKL performs best with upper error boundary being at 13%. Common to the results obtained from both aerodynamic coefficients is that the Gaussian kernel consistently yields the poorest performance. Hence, care must be taken when using Gaussian kernels in aerodynamic performance approximations.

However, with deeper look into the kernel weights for the 2D CRM drag coefficient case as shown in Figure 3.6, we observe that the Gaussian kernel is preferred for modeling the Mach number. The angle of attack, on the other hand, has the exponential weight

almost nearing 1.0. This shows the suitability of the exponential kernel for modeling the angle of attack in this case. While the MCKL will use all kernels in modeling both mach number and angle of attack, the MiKL model will use the Gaussian and exponential kernels to model the mach number and angle of attack, respectively. The ability of the MiKL and MCKL methods to obtain the optimum weights and kernels across problem dimension provides a plausible explanation on why they are better than the single kernels.

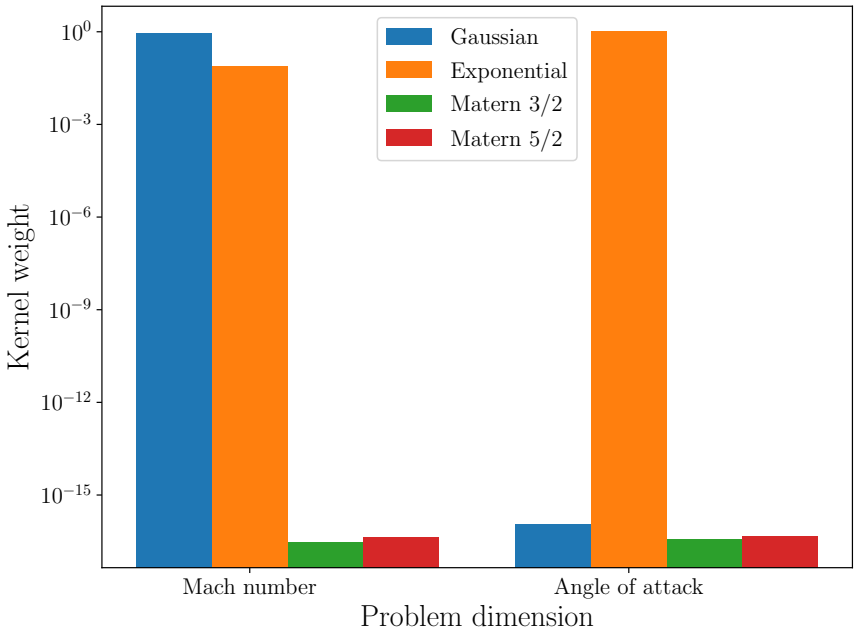
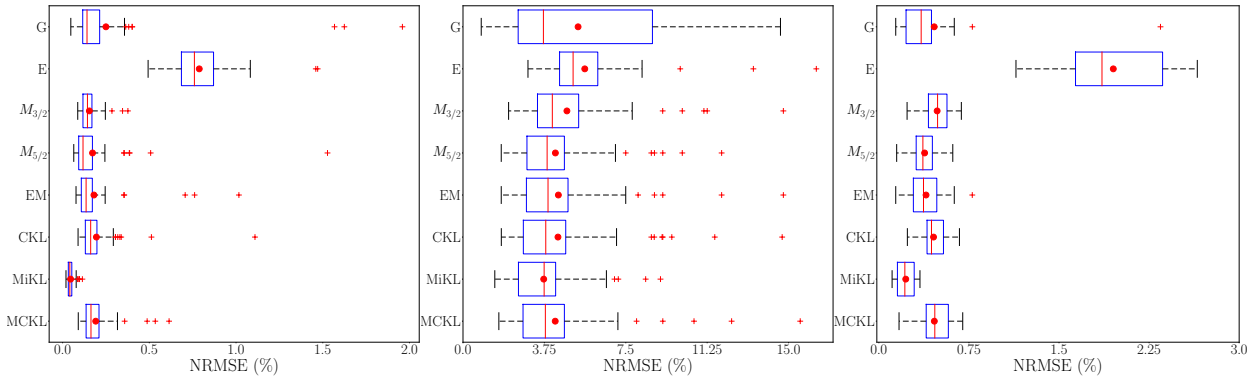


Figure 3.6: Kernel weights for both mach number and angle of attack (drag coefficient test case)

In the mission data problem, we evaluate the performance of the various models using a reverse 10-fold cross validation. The cross validation is said to be reverse here because the kriging model is built on the left-out data and validated with the rest of the data set. This is because the mission dataset has several points and it will be time consuming building kriging models for several hundred points. We present the result of the validation in Figure 3.7.

The performance comparison for the higher dimensional mission data test case is presented in Figure 3.7. From the result for lift coefficient as shown in Figure 3.7a, it is observed that the two Matérn kernels are the best performing single kernels. The performance of the Gaussian kernel here is good with the upper bound of the error bar fixed



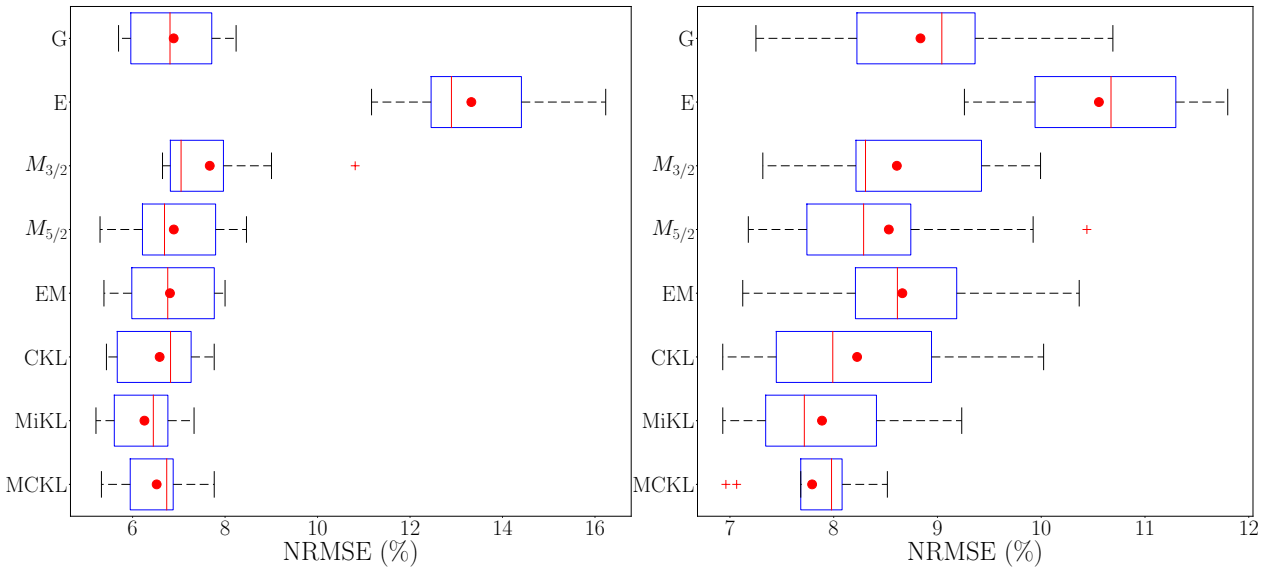
(a) Lift Coefficient (b) Drag Coefficient (c) Local pressure Coefficient

Figure 3.7: Mission data benchmark.

at about 0.42% and having similar median as the Matérn 3/2 kernel. Considering the multiple kernels, MiKL performed significantly better than other models. In this case, the performance of other multiple-kernel methods are similar. Considering the approximation accuracy of the models in the drag coefficient test case as shown in Figure 3.7b, the performance of models (single and multiple kernel models) are similar with the minimum and maximum error bars being about 1.72 and 7.34%, respectively. However, the Matérn 5/2 has the least mean error amongst other single kernel models of about 3.8% and MiKL performed slight best with a mean and median of 3.75% when compared to other multiple kernel methods. In the approximation of the local pressure coefficient as shown in Figure 3.7c, it can be observed that the Matérn 5/2 is the best performing single kernel model with a mean and median approximation error of about 0.4%. The performance of the Gaussian and the Matérn 3/2 is also remarkable. In this comparison, the exponential kernel model performs the worst with the errors spanning between 1.2 and 2.5%. In this test case, MiKL performs significantly better than both single and multiple kernel models with a median and mean error of about 0.26%.

Lastly, for both 8D and 16D airfoil shape parameter problems, we also use the reverse 10-fold cross validation to assess the performance of the models. For the results of the 8D case shown in Figure 3.8a, it is observed that the exponential kernel has the worst performance. Considering the multiple-kernel methods, MiKL yields the best performance, albeit with only a slight improvement when compared to the MCKL. This is evident in the position of the mean and median of the model benchmark. A similar performance

is seen in the 16D benchmark (Figure 3.8b) with the exponential kernel having the worst performance and MiKL performing well. The difference is that MiKL can be considered to perform the best considering the least median error. But when we consider the mean error, MCKL can be considered as the best performer.



(a) 8D (b) 16D

Figure 3.8: Airfoil shape parameter benchmark.

3.5 Assessment of complex methods in various scenarios

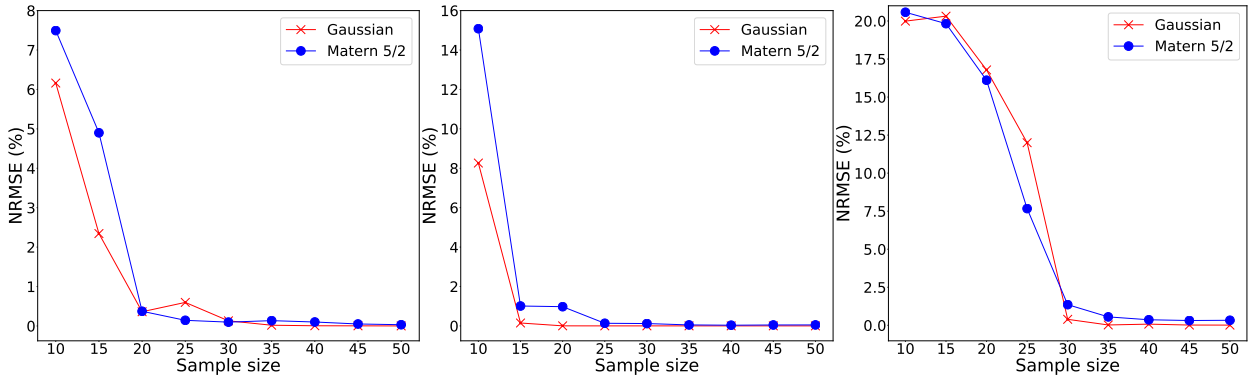
In this section, we assess the performance of MiKL and MCKL on synthetic problems. Even though our focus is on complex real world problems, the algebraic functions are used because they reflect the behaviour of complex systems in most cases. For instance, the robot arm function can be used to model the position of a robot arm which has four segments. The cantilever beam function models a simple uniform cantilever beam with horizontal and vertical loads. Hence, using algebraic functions in benchmarking studies might be sufficient in gaining insight into the behaviors and characteristics of different models. In this section, we study the effect of pre-training and choice of optimizer on the performance of our methods using several algebraic functions. The cantilever beam function is then used to study the performance of multiple kernel methods with increasing variable dimensions. For this study, the accuracy of the kriging-based models are

measured using the NRMSE.

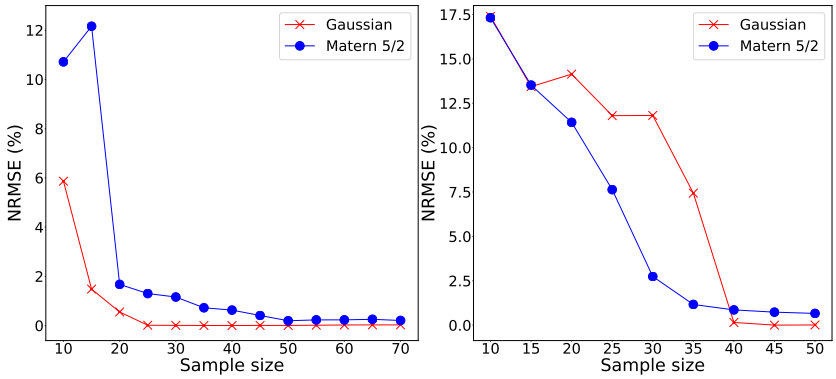
Five two-dimensional algebraic functions which include the Branin, Rosenbrock, Himmelblau, multimodal and camel back functions are used for this study. The description of the Branin, Rosenbrock, Himmelblau, multimodal and camel back functions are detailed in the Appendix A.5, A.8, A.10, A.6 and A.7 respectively. Here, we use the two-dimensional functions to better understand the behaviour of our methods to pre-training and choice of optimizer. Before the two studies, a convergence test is carried out to select optimum sample sizes for the functions. Lastly, we study the performance of the multiple kernel methods across the dimensions of a defined algebraic problem.

3.5.1 Convergence test

By most convention, ten times the problem dimension is used as the number of training data ($N_s = 10N_d$). However, this does not work every time as it does not factor other characteristics of the problem which might necessitate the use of more or less samples sizes. Hence, it is best to select the sample size after running convergence analysis. We carry out a test to study the sensitivity of selected algebraic functions to change in sample sizes. In these tests, we use the Gaussian and Matérn 5/2 kernels. We assume that the point of convergence using the two kernels will be applicable to our methods. Here, the point of convergence is defined as the point with less than 1% improvement in accuracy as measured by the NRMSE, regardless of the number of steps. For Branin, Rosenbrock, Himmelblau and camel back problems, we experiment with sample sizes between 10 and 50 with an increment of 5. For the multimodal function, the variation is from 10 to 70 also with a step size of 5. The samples points for the various sizes are generated using Halton sequence design of experiment (DoE) algorithm [105, 231]. In all the test cases carried out here, 200 validation points which comprise linearly spaced points are generated within their respective design point. We use MLE and CMA-ES algorithm for hyperparameter estimation and optimization respectively. The results of the tests are shown in Figure 3.9. From the plots, it is observed that the optimal sample sizes for the Branin, Rosenbrock, Himmelblau, multimodal 2D and camel back are 30, 25, 35, 50 and 40 respectively.



(a) Branin (b) Rosenbrock (c) Himmelblau



(d) Multimodal 2D (e) Camel back

Figure 3.9: Plots showing the convergence of sample sizes for different functions when the Gaussian and Matérn 5/2 kernels are used for model training.

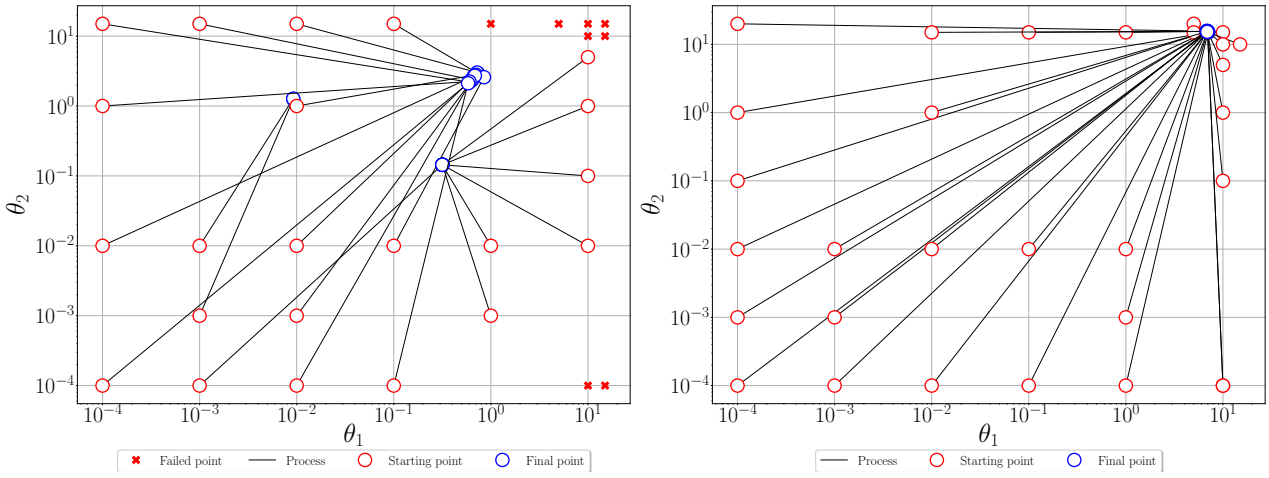
3.5.2 Effects of pre-training on model performance

When compared to the single kernel methods, our methods have extra hyperparameters to be tuned alongside the length scales hence increasing its complexity. This complexity makes the hyperparameter optimization a lot more complex which might lead to the hyperparameter converging at a local optimum. Because of this, there is a risk of the models performing poorly. Pre-training complex models with much simpler models might be beneficial.

In the context of kriging, complex methods such as the multiple kernel learning methods might be pre-trained using single kernel learning models. The next question is which kernel should be used for the pre-training to get the best results? In addition, it is important to study the hyperparameter convergence of multiple kernel methods.

When the point of convergence of a model depends on the starting point during hyperparameter optimization, it could lead to subpar model performance. This is because the hyperparameter is chosen at a local optimum. Hence, convergence of the hyperparameter at a single point or within close radius regardless of the starting point of the optimization process will be more suitable. We explore the converging abilities of single kernel as well as multiple kernel models. For the single kernel models, we use the Gaussian and Matérn 5/2 kernels and for multiple-kernel models, we use MiKL and MCKL. For this study, we particularly choose the camel back 2D function due to its complex surface profile, making it suitable for the convergence study. We then vary the initial starting points of the hyperparameter optimization and record the final hyperparameter values after optimization. The constrained Nelder-Mead [28] algorithm is used for the optimization process. The result of this exploration is shown in Figure 3.10 for the single kernel models. Figures 3.11 and 3.12 show the convergence behaviour of the MiKL and MCKL models, respectively.

In the plots, the starting and final points of the optimization processes are depicted by the circles with red and blue outlines respectively. The process is visualized with a black line which connects the starting and final points. Failure of the optimization process is defined as a situation when the final points and the starting points are almost same (less than 1% change). This is denoted with red 'X' marks on the plots.



(a) Gaussian

(b) Matern 5/2

Figure 3.10: Hyperparameter convergence behaviour of the Gaussian and Matern 5/2 kernels.

As shown in Figure 3.10a, there are several final points after hyperparameter opti-

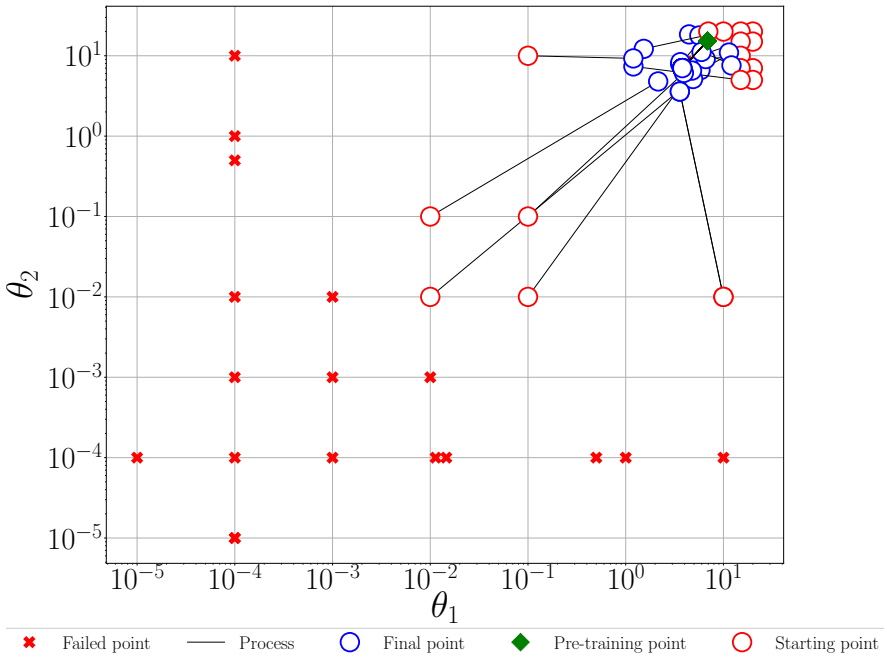


Figure 3.11: Hyperparameter convergence behaviour of the MiKL model.

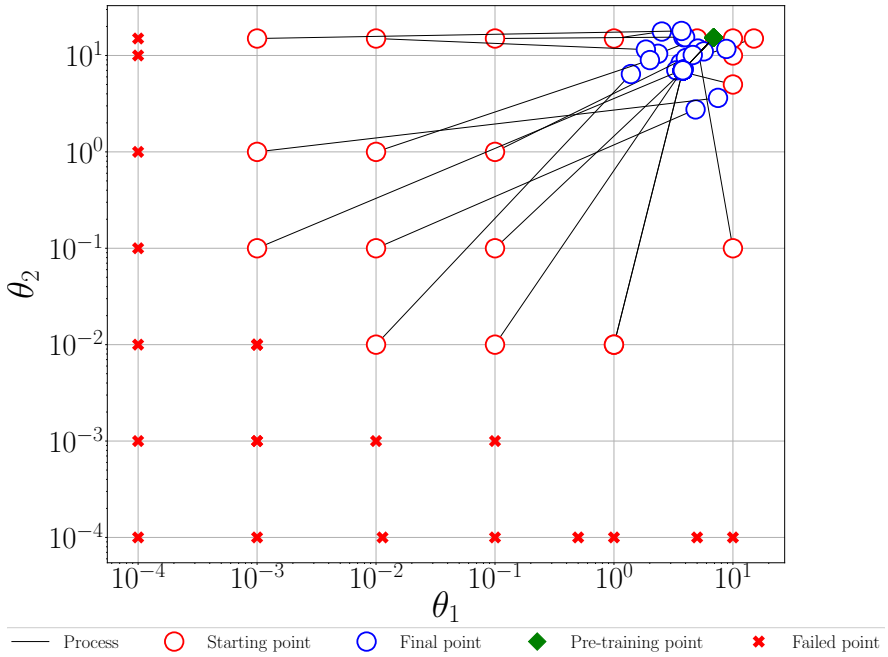


Figure 3.12: Hyperparameter convergence behaviour of MCKL model.

mization. Similar starting points tend to converge at same final points. This phenomenon could affect the performance of the Gaussian kernel when used to model difficult problems such as the camel back problem. The case is different when the Matérn 5/2 kernel

is used, as shown in Figure 3.10b. With the multiple kernel methods, the hyperparameter convergence test shows that they are sensitive to the starting points. When the starting points are far away, it could lead to a failure in the optimization process as shown in Figures 3.11 and 3.12. However, when a reasonable hyperparameter bound is defined, the multiple kernel methods have a high chance achieving high model performance. This is because they tend to converge at the global optimum easily.

Next, we explore the effects of warming up the hyperparameter optimization process by using the optimum hyperparameters of the kriging model trained with single kernel based kriging model as the initial point. Three models are used in the comparison intended in this study. They include model trained with Gaussian kernel, MiKL and the MCKL. The Gaussian kernel-trained model is used as a representative of single kernel-based methods. In this study, we compare the performance of these three models with and without pre-training. The performance is measured based on two criteria; the NRMSE (%) and the training time (s). The optimal hyperparameter from a Matérn 5/2-based kriging model is used as the initial point for hyperparameter optimization of the three models. In this study we use the five algebraic functions for benchmarking purpose. To train the models, we generate the sample points using Halton sequence and used the optimum sample sizes obtained in Section 3.5.1 for each of the functions. Similar to Section 3.5.1, we generate 200 validation points for testing purpose. We use the CMA-ES algorithm for hyperparameter optimization.

The effect of pre-training on the model accuracy measured by the NRMSE is summarized in Table 3.5. For the MiKL method, improvement in model accuracy is observed as seen in the Branin, multimodal 2D and camel back functions. The largest improvement is seen in the camel back function test case where the model error notably reduces from 2.745% to 0.28%. Slight improvements are also observed in the Branin and multimodal 2D cases. For MCKL, improvement in the model accuracy is observed in both the Himmelblau and Rosenbrock function test cases. However, there is no difference in the accuracy of the model in the case of the Branin and multimodal 2D. Figures 3.13, 3.14 and 3.15 show the plots of NRMSE values obtained from the benchmark functions with and without pre-training for the Gaussian kernel, MiKL and MCKL models respectively.

The effect of pre-training on the computational training time measured in seconds (s) is

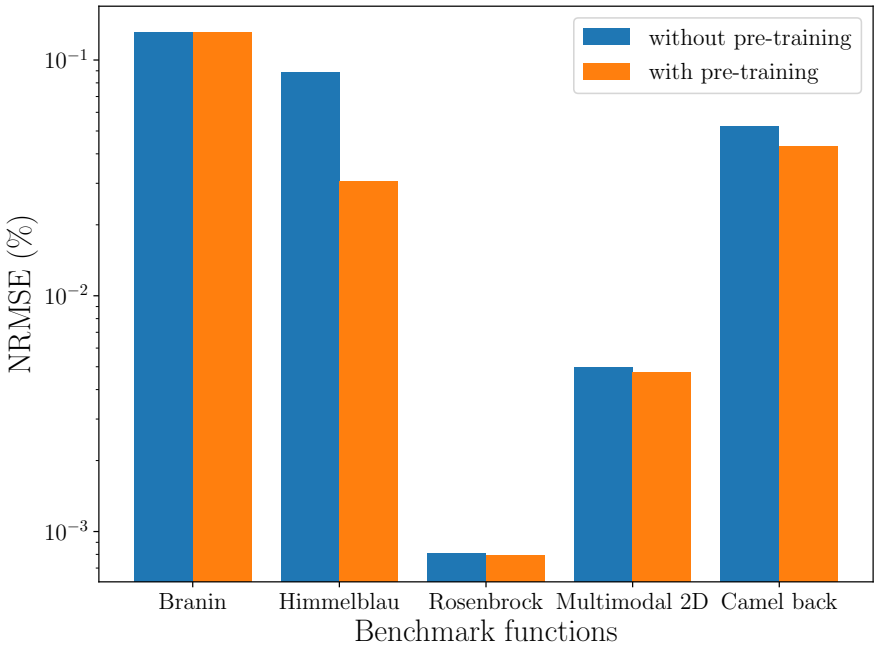


Figure 3.13: Effect of pre-training on model accuracy of the Gaussian kernel model.

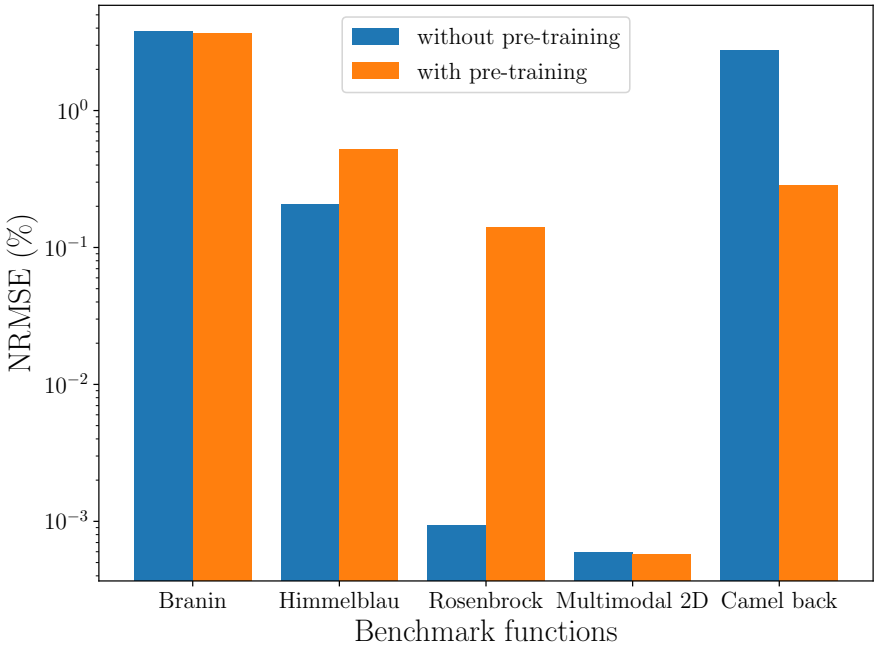


Figure 3.14: Effect of pre-training on model accuracy of the MiKL model.

summarized in Table 3.6. Figures 3.16, 3.17 and 3.18 show the plots of training time values obtained from the benchmark functions with and without pre-training for the Gaussian kernel, MiKL and MCKL models respectively.

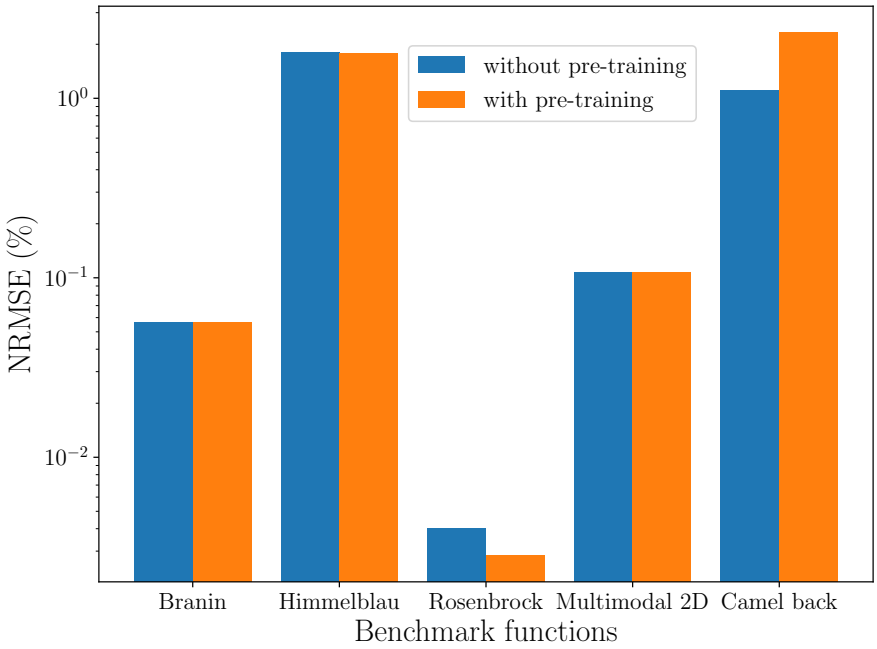


Figure 3.15: Effect of pre-training on model accuracy of the MCKL model.

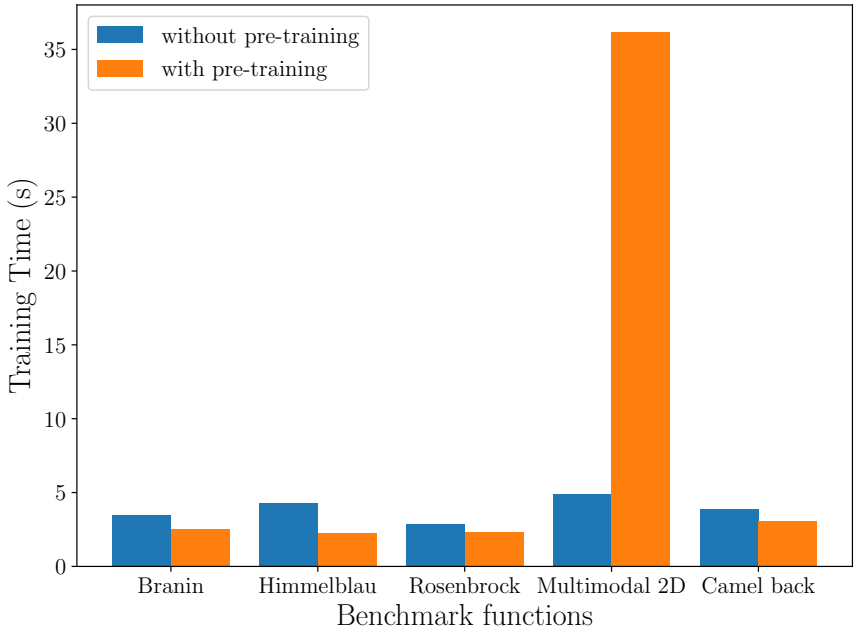


Figure 3.16: Effect of pre-training on training time of the Gaussian kernel model.

Reductions in the training time is observed with most of the benchmark cases. For instance, with MiKL, slight reduction in training time is observed when the models are pre-trained. A good explanation for this is that MiKL requires two stages. Hence, savings

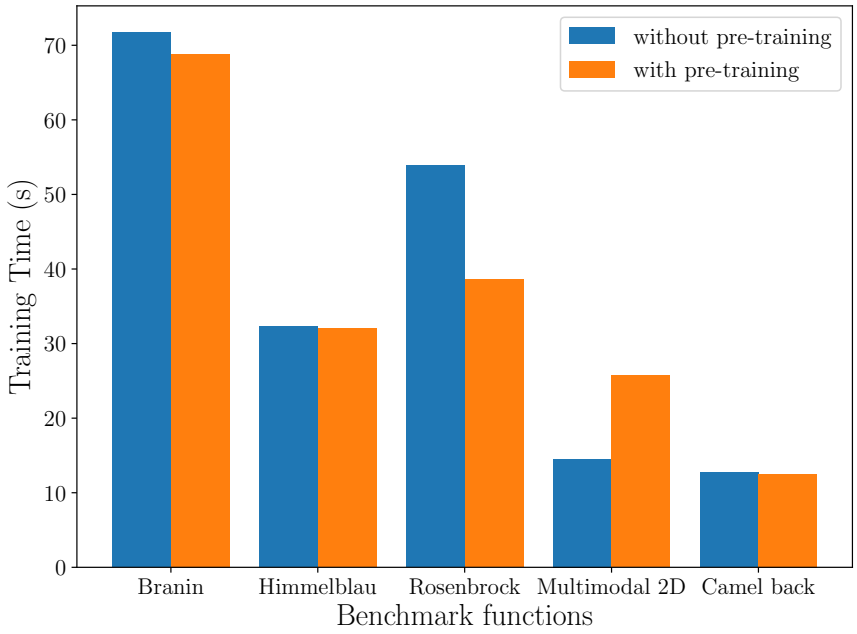


Figure 3.17: Effect of pre-training on training time of the MiKL model.

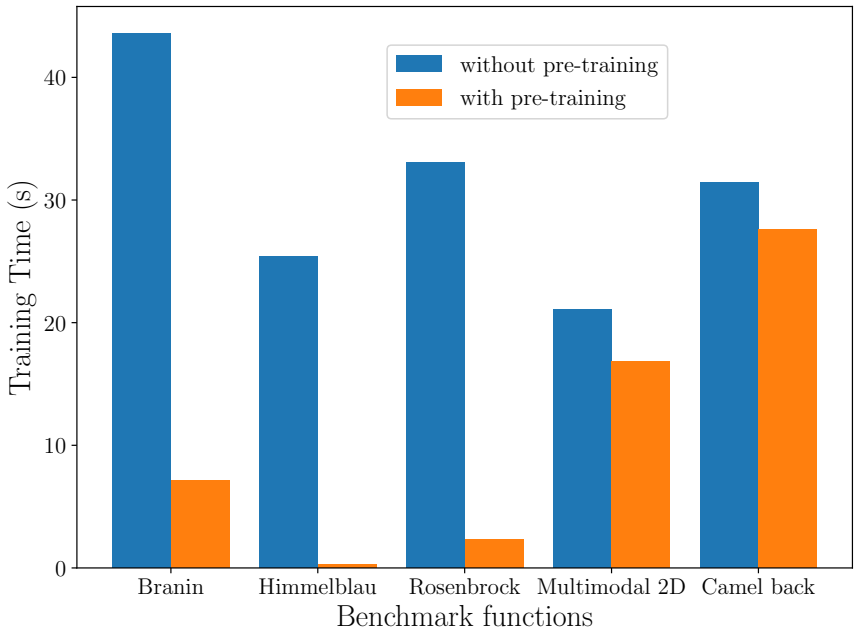


Figure 3.18: Effect of pre-training on training time of the MCKL model.

in the first stage might contribute very little to the overall time reduction.

In summary, pre-training with a single kernel-based kriging model could help save great computational time and makes it more likely to improve the accuracy of models. The

Table 3.5: Effect of pre-training with Matérn 5/2 kernel on model accuracy.

	Without pre-training - NRMSE (%)			With pre-training - NRMSE (%)		
	Gaussian	MiKL	MCKL	Gaussian	MiKL	MCKL
Branin	1.31×10^{-1}	3.78×10^{0}	5.67×10^{-2}	1.31×10^{-1}	3.63×10^{0}	5.67×10^{-2}
Himmelblau	8.90×10^{-2}	2.06×10^{-1}	1.80×10^{0}	3.07×10^{-2}	5.19×10^{-1}	1.77×10^{0}
Rosenbrock	8.09×10^{-4}	9.37×10^{-4}	4.00×10^{-3}	7.90×10^{-4}	1.41×10^{-1}	2.83×10^{-3}
Multimodal 2D	4.96×10^{-3}	5.88×10^{-4}	1.07×10^{-1}	4.73×10^{-3}	5.70×10^{-4}	1.07×10^{-1}
Camel back	5.23×10^{-2}	2.75×10^{0}	1.10×10^{0}	4.30×10^{-2}	2.85×10^{-1}	2.33×10^{0}

Table 3.6: Effect of pre-training with Matérn 5/2 kernel on training time.

	Without pre-training (seconds)			With pre-training (seconds)		
	Gaussian	MiKL	MCKL	Gaussian	MiKL	MCKL
Branin	3.49×10^{0}	7.17×10^{1}	4.36×10^{1}	2.53×10^{0}	6.88×10^{1}	7.11×10^{0}
Himmelblau	4.29×10^{0}	3.23×10^{1}	2.54×10^{1}	2.23×10^{0}	3.20×10^{1}	2.98×10^{-1}
Rosenbrock	2.86×10^{0}	5.39×10^{1}	3.31×10^{1}	2.30×10^{0}	3.86×10^{1}	2.32×10^{0}
Multimodal 2D	4.90×10^{0}	1.45×10^{1}	2.11×10^{1}	3.62×10^{1}	2.57×10^{1}	1.68×10^{1}
Camel back	3.85×10^{0}	1.27×10^{1}	3.14×10^{1}	3.05×10^{0}	1.25×10^{1}	2.76×10^{1}

savings in computational time could be attributed to the ease of convergence at the global optimum of our model's hyperparameter when it is warmed up using a pre-informed hyperparameter.

3.5.3 Effect of optimizer on model performance

Here, we study the effect of using different optimization codes in the training of both MiKL and MCKL methods. We use three optimizers which includes a direct search, evolution-based algorithm and a quadratic programming (QP) algorithms. For the direct search algorithm, we use a constrained Nelder-mead package, `constNMPy` package [28], and CMA-ES as the evolution-based algorithm. The SLSQP that is based on the sequential quadratic programming (SQP) method and dedicated to the solution of nonlinear programming problems, is also used in the study. Two models are used in the comparison intended in this study. They include models trained with our methods, MiKL and MCKL. In this study, we compare the performance of the three optimization algorithms in the training of these two models. The performance is measured based on two criteria, namely

the NRMSE (%) and the training time (seconds). In this study, we use the five algebraic functions for benchmarking purpose. To train the models, we generate the sample points using Halton sequence and use the optimum sample sizes obtained in Section 3.5.1 for each of the functions. Similar to Section 3.5.1, we generate 200 validation points for testing purpose.

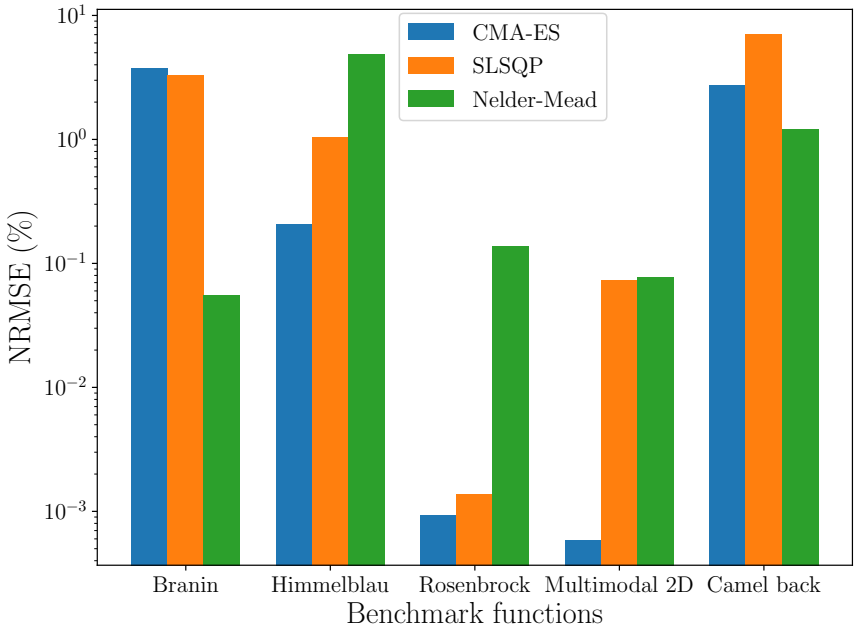


Figure 3.19: Effect of optimizer choice on model accuracy of the MiKL model.

The effect of choice of optimizer on the model accuracy measured by the NRMSE is summarized in Table 3.7. For the MiKL method, an improvement in model accuracy is observed when CMA-ES is used as the optimization algorithm. This is evident in the Himmelblau, multimodal 2D and Rosenbrock functions. The largest improvement when CMA-ES is used are seen in both the multimodal 2D and Rosenbrock functions. After the CMA-ES algorithm, SLSQP performed better with these three function. However, great improvement is seen in Branin and camel back functions when Nelder-Mead algorithm is used as the optimization algorithm. For the MCKL method, there is a general improvement in the accuracy of models trained with the CMA-ES algorithm. These show the superiority of the CMA-ES as the choice of optimizer for the MCKL method. The effect of optimizer choice on model accuracy for the MiKL and MCKL models are shown in Figures 3.19 and 3.20 respectively.

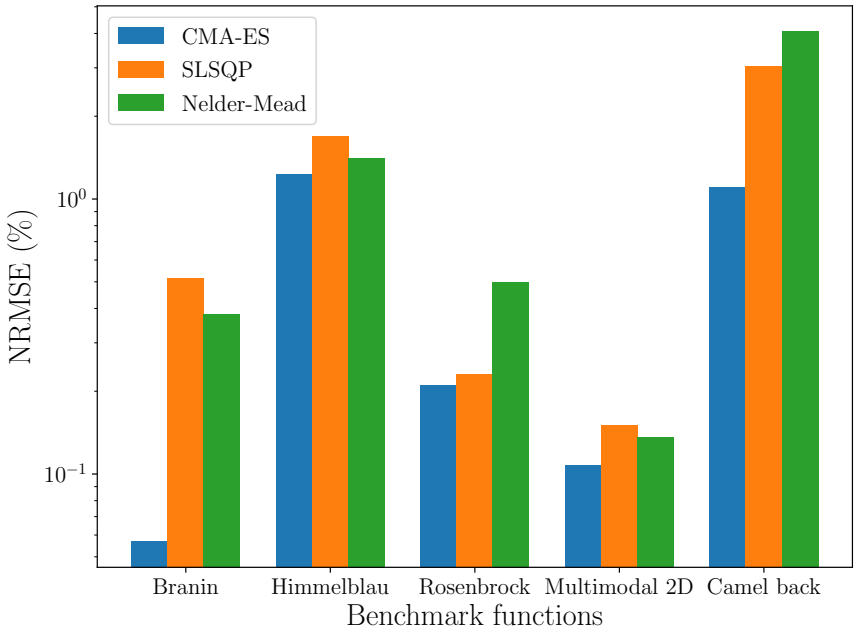


Figure 3.20: Effect of optimizer on model accuracy of the MCKL model.

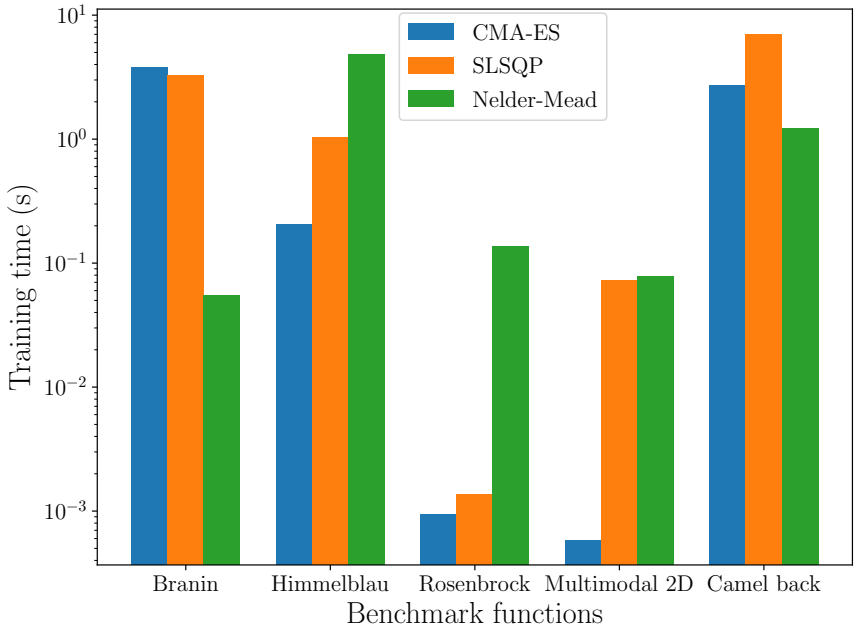


Figure 3.21: Effect of optimizer choice on training time of the MiKL model.

The effect of optimizer choice on the training time measured in seconds (s) is summarized in Table 3.8. For both the MiKL and MCKL methods, the computational time incurred by the CMA-ES is consistently several multiples of that recorded with SLSQP

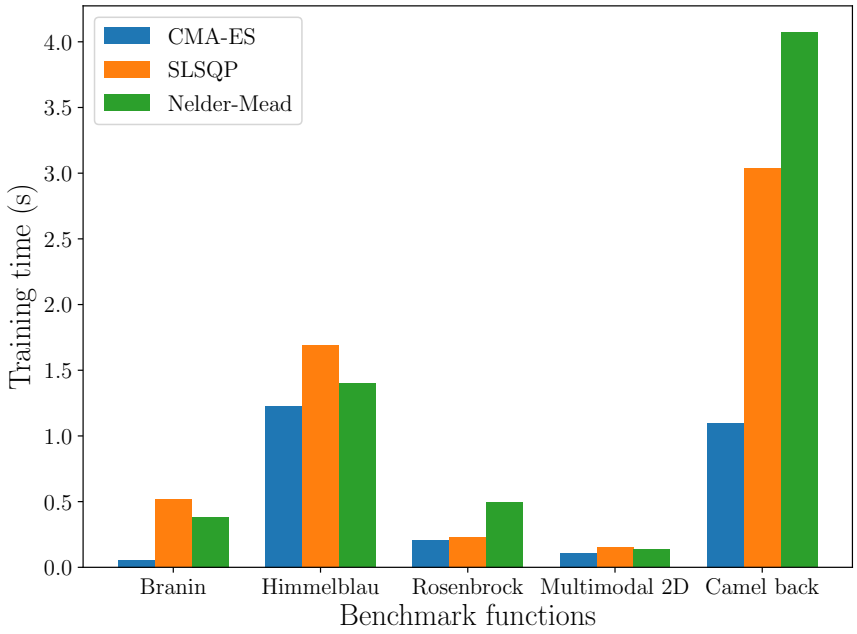


Figure 3.22: Effect of optimizer on training time of the MCKL model.

Table 3.7: Effect of optimizer choice on model accuracy.

	CMA-ES - NRMSE(%)		SLSQP - NRMSE(%)		Nelder-Mead - NRMSE(%)	
	MiKL	MCKL	MiKL	MCKL	MiKL	MCKL
Branin	3.78×10^{0}	5.67×10^{-2}	3.28×10^{0}	5.16×10^{-1}	5.50×10^{-2}	3.81×10^{-1}
Himmelblau	2.06×10^{-1}	1.23×10^{0}	1.04×10^{0}	1.69×10^{0}	4.89×10^{0}	1.40×10^{0}
Rosenbrock	9.37×10^{-4}	2.10×10^{-1}	1.38×10^{-3}	2.30×10^{-1}	1.37×10^{-1}	4.98×10^{-1}
Multimodal 2D	5.88×10^{-4}	1.07×10^{-1}	7.30×10^{-2}	1.50×10^{-1}	7.80×10^{-2}	1.36×10^{-1}
Camel back	2.75×10^{0}	1.10×10^{0}	7.00×10^{0}	3.04×10^{0}	1.22×10^{0}	4.07×10^{0}

and Nelder-Mead trained models. For MiKL method, SLSQP and the Nelder-Mead have almost similar training time for the five functions. However, the two methods show a large disparity in the computational time when MCKL is used. SLSQP notably consumes less time in training models built on the MCKL. The effect of optimizer choice on training time for the MiKL and MCKL models are shown in Figures 3.21 and 3.22 respectively.

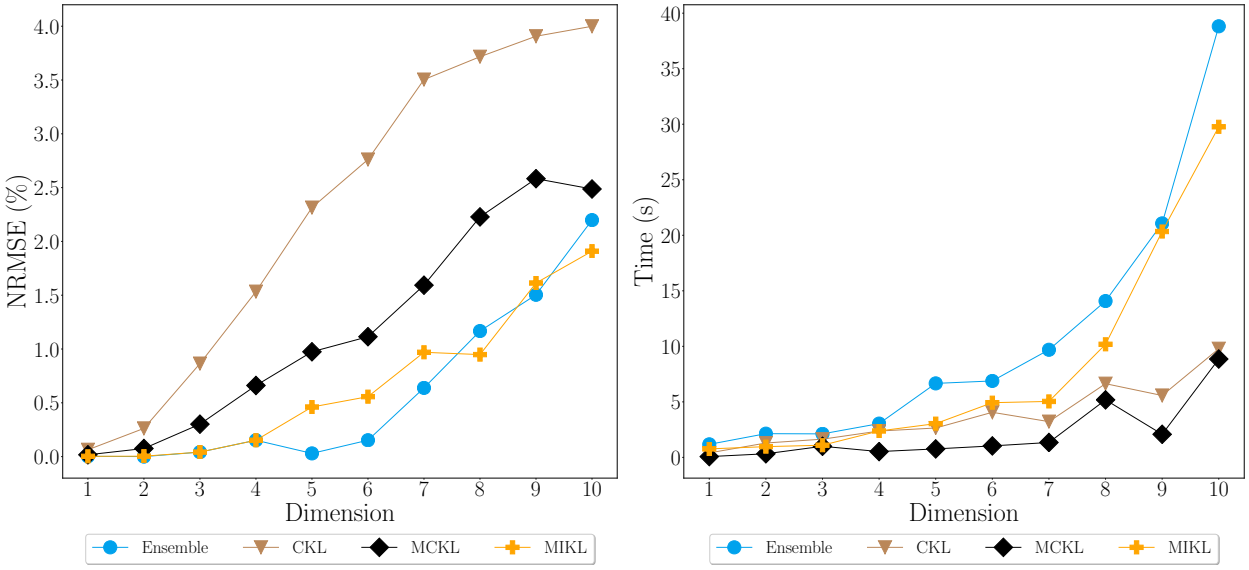
In summary, implementing CMA-ES algorithm as the choice of optimizer for both the MiKL and MCKL most likely will yield better results. However, this comes at a cost of increased computational time. This is worsened when optimizer restarts are implemented with the CMA-ES framework in order to ensure the convergence of the hyperparameter at the global optimum.

Table 3.8: Effect of optimizer choice on training time.

	CMA-ES (s)		SLSQP (s)		Nelder-Mead (s)	
	MiKL	MCKL	MiKL	MCKL	MiKL	MCKL
Branin	7.17×10^1	4.36×10^1	2.56×10^1	2.30×10^0	2.42×10^1	1.44×10^1
Himmelblau	3.23×10^1	2.54×10^1	2.49×10^1	1.23×10^0	2.48×10^1	1.30×10^0
Rosenbrock	5.39×10^1	3.31×10^1	1.56×10^1	7.85×10^{-1}	1.78×10^1	1.27×10^0
Multimodal 2D	1.45×10^1	2.11×10^1	7.23×10^1	1.58×10^0	7.02×10^1	1.84×10^0
Camel back	1.27×10^1	3.14×10^1	4.42×10^1	1.47×10^0	4.18×10^1	1.51×10^0

3.5.4 Performance across dimensions

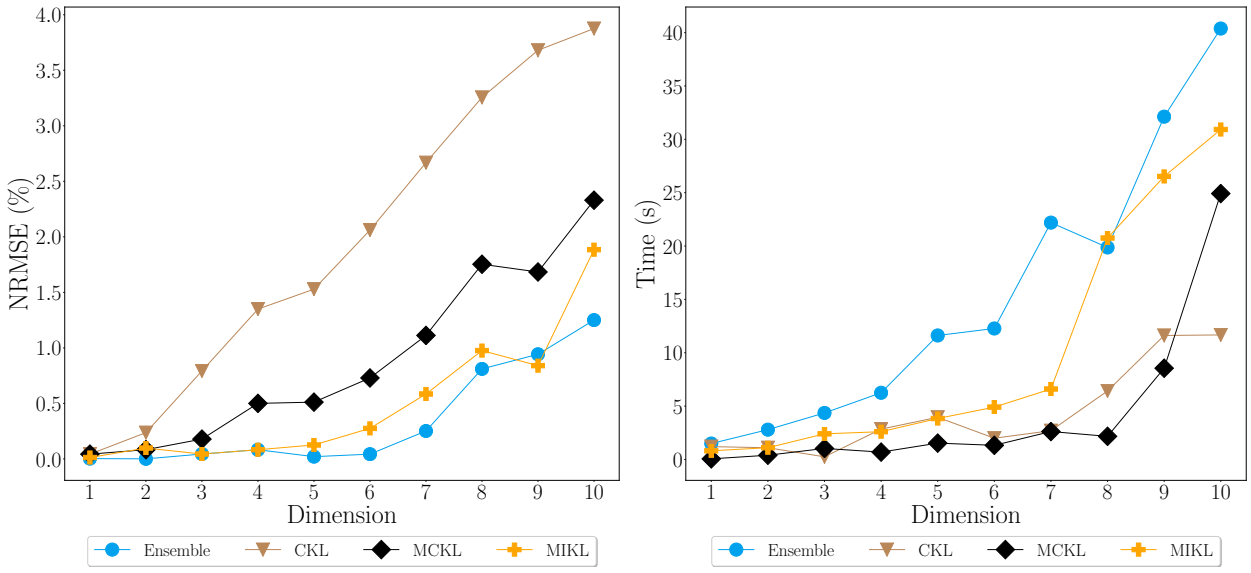
Here, we study the computational complexity and accuracy of our methods by benchmarking using the cantilever beam problem. The cantilever beam algebraic function is a common N_d-dimensional problem used for benchmarking within the surrogate modeling community. For different sample sizes (N_s = 40, 50 and 60), we vary the dimension from 1 to 10, fixing the test size as 1000 points. We then compare the NRMSE and time complexity for all multiple kernel methods. Figures 3.23, 3.24, and 3.25 show the performance benchmark using 40, 50 and 60 training points respectively.



(a) Accuracy (b) Time

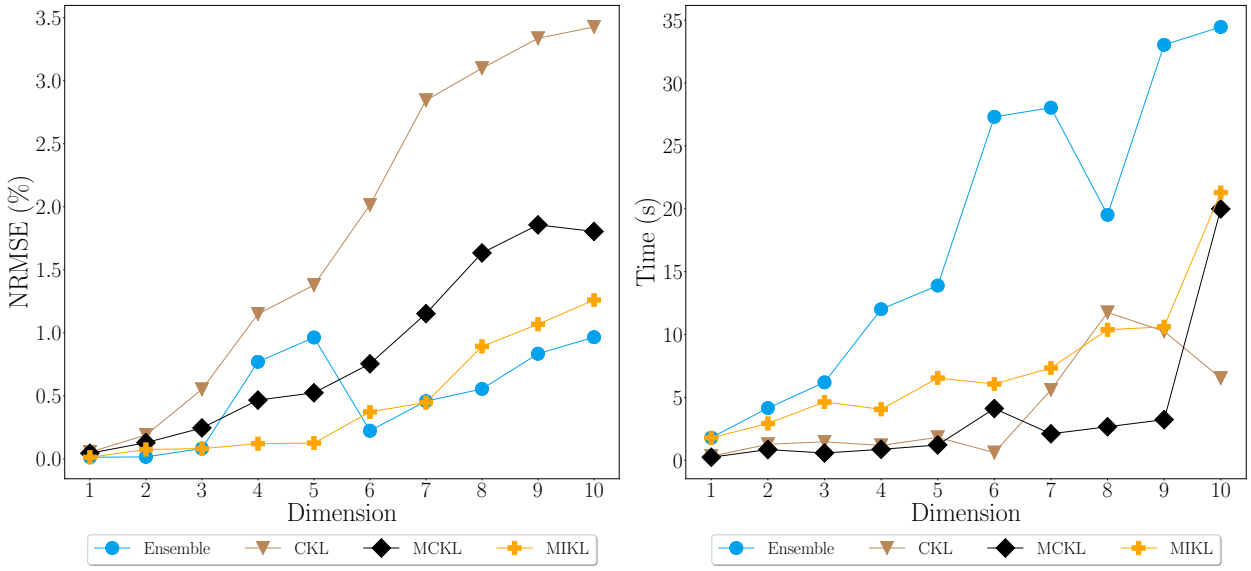
Figure 3.23: Performance benchmark across dimensions for a cantilever problem (N_s = 40).

From the results, for all models, an expected increase in the computational error is



(a) Accuracy (b) Time

Figure 3.24: Performance benchmark across dimensions for a cantilever problem (N_s = 50).



(a) Accuracy (b) Time

Figure 3.25: Performance benchmark across dimensions for a cantilever problem (N_s = 60).

observed with increase in problem dimension. This is because we keep the number of samples constant while increasing the dimension (intuitively, the complexity) of the problem. In terms of the model accuracy, the ensemble method performs best, followed closely

by MiKL. Similarly, the MCKL method performs much better than the CKL method with a noticeable increase in the performance margin with the increase in problem dimension. In terms of computational time, the slope is less steep for most of the models with the exception of the ensemble methods. From the results, we can also observe that the ensemble method incurs the highest time cost, which we believe is due to the steps involved in its training, which require training the individual kernel models to obtain their respective optimum log-likelihood (LL). To complete the training, the optimum values are then used to derive the weights for the averaging purposes (ensembles). Lastly, MCKL is shown to be the most computationally efficient, which is observed over the range of problem dimensions tested. However, a surge in training time is seen in the tenth dimension with MCKL. With MiKL, significant increase in computational time is recorded with increase in problem dimension. Upon observing the behaviour of both multiple kernel methods, we can conclude that their application should be limited to low-dimensional problems.

3.6 Summary

In this chapter, we proposed two new methods, namely MiKL and MCKL, to capture the complexity found in real-world datasets more effectively. Our main motivation was to further improve the accuracy and robustness of kriging models in multidimensional problems. Our work was based on the assumption that single kernel methods or multiple kernel learning method which uses a "super kernel" are not sufficient to capture the inherent complexity which may arise across the dimensions in some datasets. We introduced a weight matrix which combines a unique weight vector of kernels for each variable. MiKL learned to attach a unique kernel to each variable within the kriging framework. On the other hand, MCKL assumed the uniqueness and complexity of each variable and derives a weighted combination of kernels for each variable in a dataset. Common to both methods is the optimization problem where a penalty function is added to ensure that the sum of rows of the weight matrix is equal to one. The learning of the length scale and the weight matrix is done simultaneously. This we achieved by flattening the weight matrix while still enforcing the constraint conditions. The introduction of the weight matrix led to the increase in the number of hyperparamters to be tuned, hence increasing the possibilities of the models performing poorly. This possibility is exacerbated when the hyperparameters

do not converge at the global optimum. We proposed to address this problem by pre-training the models and the use of CMA-ES algorithm for hyperparameter optimization. The pre-training of models involved using the final hyperparameter from the training a kriging model with single kernel, as the starting point of a more complex optimization procedure. Using CMA-ES algorithm with multi-restarts for hyperparameter optimization to a large extent ensured that the hyperparameters converged at the global optimum.

From testing the models using several engineering datasets spanning different problem dimensions, we have demonstrated that the use of multiple kernel models could give consistent and good modeling performance when compared to single kernel models. Comparing the existing state-of-the-art multiple kernel methods with the proposed methods in this chapter, we discovered that CKL gives similar modeling performance with MCKL in most cases. This could mean that it is not necessary to use unique kernel combinations across problem dimensions. However, varying the kernel across the problem dimension, as in MiKL, was shown to be able to notably improve the performance of a surrogate model, albeit at the expense of increased computational cost. This is because the number of parameters to be trained increases greatly with increase in problem dimension. The weight matrix that has to be trained contributes largely to this increase in parameters. Hence, multiple kernel models might be more suitable for low-dimensional problems.

Lastly, in all of the multiple kernel methods, the computational complexity remained $\mathcal{O}(N_s^3)$. However, the computational efficiency of the methods can be described by the number of parameters to be optimized during model training. In MiKL and MCKL for instance, $N_d(1+m)$ and mN_d number of parameters are optimized, respectively.

CHAPTER 4

AUTOMATED KERNEL SELECTIONS FOR LOW-DIMENSIONAL PROBLEMS

Kernel selection is a non-trivial problem in kriging. Poor kernel selection, particularly for complex problems, may result in erroneous approximations. Consequently, it is crucial to choose the right kernel for particular applications. Leveraging the most frequently used kernels in the field of engineering, we proposed two fully automated algorithms for kernel selection and model training for several problems. In most of the benchmark cases, models trained using the proposed methods outperformed other models. For example, the model accuracy for the axial transonic problem with NASA rotor 37 as its datum shape improved considerably. Overall, the results were encouraging and showed that the proposed methods can perform kernel selection automation in surrogate modeling techniques like kriging.

4.1 Selection of kernels in low dimensional problems

Kernel-based methods, also known as *kernel machines*, are a type of machine learning methods that are well-suited for nonlinear problems. According to Schölkopf *et al.* [220], they can identify nonlinear patterns, while the computational elegance of matrix algebra is retained. Kriging [51, 52], kernel principal component analysis (KPCA) [127], support vector machine (SVM) [110], and kernel Fisher discriminant analysis [172] are some of the most prominent and commonly used kernel-based methods. In addition, kernel-based methods have been used in applications involving geostatistics [166], bioinformatics [98], signal processing [199], information extraction [120], data compression [80], and pattern recognition [26]. Despite their tremendous usefulness, selecting the most suitable kernel for a particular problem remains a challenge. The kernel's role is to project data onto higher-dimensional spaces, such that linear methods may then be applicable [50, 201]. In

fact, the choice of the kernel, for model training of kernel-based methods, largely determines the performance of such model.

In this chapter, we will primarily investigate kriging [195], which is one of the most prominent kernel-based method. Kernels, which are of various types, are crucial to kriging model structures. Kernels are typically composed of two basic components: the hyperparameter θ, and the distance, which is a measure of point separation $|x - x'|$. There are numerous possible kernels, each with a set of free hyperparameters whose values must be determined in order to define the model structure. Defining the most optimal kernel/covariance matrix for a given problem entails both kernel selection (i.e., comparing different kernels) and evaluating the hyperparameters [208]. According to Ginsbourger *et al.* [91], the use of unadaptive kernels or models can have a detrimental effect on model accuracy. As a result, it becomes imperative to have kriging models with enough flexibility to prevent model misspecification. A kriging kernel from a pool of kernels may be chosen for a specific application. Selecting the most optimal model or kernel from a set of competing models can be challenging, including the application of black box methods where a detailed understanding of the physics is obscure [2]. Most times, the exponential, Gaussian, Matérn 3/2, and Matérn 5/2 kernels are typically the only ones available to kriging users in the engineering community (i.e., kriging users) [193]. Recently, the cubic kernel has also become popular due to its compact support [191]. For applications where the dataset is considerably large, kernels with compact supports work well, since they produce sparse matrices [176]. For instance, Liem *et al.* [144] employed the cubic kernel to predict aerodynamic performance. However, since applications with large data are not considered in this chapter, a thorough study of such kernels is thus not included herein.

Numerous approaches have been suggested in the literature for choosing the best kernels for particular applications. These techniques are referred to as *automatic kernel learning methods*. Deep learning, greedy search, cross validation, and trial-and-error are a few of the strategies and methods used. In Section 4.2, more information about the various methods is provided. Multiple kernel learning (MKL) methods [97, 151, 146] have gained popularity recently owing to the numerous difficulties that researchers have encountered while using various automatic kernel learning techniques. Primarily, there are two causes for MKL's recent prevalence. For this, it is necessary to increase model accuracy beyond

what can be accomplished with models that were trained with single kernels. Besides, there is a need to preserve the information obtained from all considered kernels during the learning process. One way to achieve multiple kernel learning methods is the combination of the predictions from models trained with various kernels, and this is termed *ensemble learning* [96, 3, 271, 196, 197]. Another widely used method, known as *composite kernel learning method* [68, 193, 191], entails the intrinsic mixing of kernels during model training. Multiple kernel methods have been discussed in details in Chapter 3. The absence of thorough research on the behavior of the kernels in various contexts is a common theme in both automatic kernel learning and multiple kernel learning approaches from the literature. Hence, we aim to demonstrate and comprehend the behavior of various kriging kernels in this chapter. We will thereafter suggest two techniques to facilitate a more systematic kernel selection process based on the intuitions garnered from the studies on kernels' behaviour.

The rest of this chapter is structured as follows: a succinct review of techniques that have been suggested for kernel selections is provided in Section 4.2. The design and findings from the kernels' behaviour study are then presented in Section 4.3. On the basis of our observations from the previous section, Section 4.4 provides an outline of the proposed methods, including their formulations and algorithms. Then, in Section 4.5, we discuss various test cases that will be used for this study, and in Section 4.6, we present and extensively discuss the results. Lastly, we summarize the findings in Section 4.7.

4.2 Review on the selection of kernels in kriging-based applications

The efficiency of kernel-based methods depend on the kernel selections [4]. Numerous strategies have been developed for this purpose, over the years, in order to arrive at best kernel for a particular problem. When developing a model, scientists and engineers initially choose the kernel based on their prior knowledge and intuition. This strategy has been less applicable over time, especially when varieties of problems are involved. Researchers typically adopt a trial-and-error approach for kernel selection in order to identify the most optimal kernels for various problem sets. A cross-validation method for

choosing the optimum kernel for surrogate modeling problems was proposed by Viana *et al.* [252]. The approach explores all available kernels for model training and prediction, before choosing the best model in terms of accuracy. The issue with this method is that the best kernel can only be selected after all considered kernels have been trained and tested. Intuitively, the selection process' sensitivity to the considered kernel set is a major setback of this strategy. Biswas *et al.* [27] proposed a mean error-based approach to choose the best kernel (correlation function) for kriging-based applications. In their paper, various kernels are utilized for training the model, and the kernel that minimizes the overall mean error ϵ is chosen as the best. Although this method might be useful for choosing a kernel with acceptable performance, the use of a leave-one-out cross validation (LOOCV) technique in the computation of the overall mean error from the different kernels makes it difficult for the proposed method to handle problems with large sample sizes and dimensions. However, the greedy search method [63, 62] is a common method that can help select the kernel which best represents the data. The Bayesian model evidence, which may be calculated by maximizing the marginal likelihood over the specified hyperparameters, is typically used to quantify the kernel's optimality. Due to the high computational demand for calculating the evidence, especially when multi-dimensional integration is involved, the appropriateness of each kernel is often assessed using a close substitute to the model evidence, such as AIC [255], Bayesian Information Criterion (BIC) [38, 8], etc. Instead of choosing one kernel, one should take into account several options for a more method-based Bayesian approach to kernel design. A typical kernel search operates under three major assumptions; 1) contributions from all considered kernels save the one selected kernel κ^* are negligible; 2) the contributions from only the maximum likelihood hyperparameters θ^* are to be considered while others are disregarded; and 3) the considered substitute (e.g. AIC, BIC, etc.) for the model evidence is a trustworthy one. All these assumptions are not true in many settings, such as those in which the data is noisy or sparse [227].

A meta-learning strategy was used by Ali and Smith-Miles [11] to automatically choose the kernel for support vector machines. In the proposed strategy, the relationship between the most optimal kernel and the dataset's properties was identified using the decision tree algorithm. The data properties investigated include estimates of the input from probability distribution functions such as gamma, Gaussian, and Rayleigh. The RBF kernel was

suggested by the authors based on the study's findings due to its strong overall performance. However, it is unclear how the considered statistical test assesses the data properties' and affects the kernel selection. Furthermore, because the data properties are solely deduced from the input $\mathbf{x}$, the relationship between the input $\mathbf{x}$ and output $\mathbf{y}$ is not taken into account. Although the limitation highlighted may be inconsequential for classification applications, it may not be so for regression applications. Simple closed form kernels that can be used in conjunction with Gaussian processes to identify patterns and allow extrapolation were introduced by Wilson and Adams [264]. The closed form kernels are developed by modeling a spectral density with a Gaussian mixture.

For machine learning applications, Abdessalem *et al.* [2] proposed the use of the approximate Bayesian computation (ABC) method for estimation of parameters and selection of kernel in machine learning problems. With their method, evaluation of the likelihood function is bypassed. The ABC-based approach was also reported to be independent of problem dimension. However, the bottlenecks of this method are the overfitting tendencies in high-dimensional parameter spaces. Salem and Tomaso [217] proposed a general criterion for selecting the best models using internal accuracy, prediction performance, and a roughness penalty. With this approach, model accuracy is guaranteed while overfitting will be constrained. Its extensive use is nonetheless constrained by its exhaustive nature. Fischer [72] proposed a model selection framework based on the approximation set coding (ASC) concept to assess kernels for Gaussian process regression. The approximation set coding (ASC) helps in obtaining an optimal balance between a model's expressiveness and repeatability [35]. The choice of kernels for kernel-based techniques like the Gaussian process has also been studied using deep learning. Salakhutdinov and Hinton developed a deep belief networks (DBNs) method to learn Gaussian processes' covariance functions [216]. It has been reported that fine-tuning kernel covariance, using backpropagation through DBN, significantly enhance the performance of both classification and regression problems.

Simpson *et al.* [227] proposed the use of transformers for determining the kernel for certain applications. This proposed method guides kernel recommendation using a decoder, with an array of conventional kernels or product of kernels, that maps an encoded representation of a dataset to a kernel. Kernel identification through transformers (KITT) may

predict the most optimal kernels for a wide range of real datasets if it is trained with an adequate vocabulary of kernels. Although the authors asserted that the identification of kernels could be completed in 0.1 seconds or less, KITT's requirement of a supercomputer is one of its many limitations. As much as possible, methods for kernel selections and model hyperparameter evaluations (particularly within the context of the kernel-based method) should not involve advanced computations that will rely on high-performance computing, as this will hinder their wide adoption and use.

4.3 Kernel Studies

In this section, we examine the behavior of kernels in various contexts in order to gain valuable insights needed for better kernel selection in kernel-based techniques like kriging. Specifically, we investigate the impact of sample sizes and profile of the problem on the selection of kernels. We also investigate various kernels' sensitivities to changes in the values of hyperparameters. This is accomplished by changing the hyperparameter's value over a wide range and examining its effect on the likelihood estimate of one- and two-dimensional algebraic problems.

4.3.1 Effect of problem profiles on kernel selections

In this section, we compare the effectiveness of four kernels that are often employed in the surrogate modeling field. The benchmarks are carried out utilizing the robot arm, Haupt, and Branin functions, each of which has unique properties in its function profile. We use the Halton sequence sampling technique [105] to generate 40 sample points within each of the synthetic problems' input regions. Figure 4.1 presents the differing accuracy levels, in terms of RMSE, obtained by utilizing various kernel functions for all benchmark problems.

The four kernels are found to adequately simulate the Branin function in Figure 4.1, with the greatest error measurement below 2.5%, which is significantly lower than in other cases. The Branin function has the smoothest profile among the three problems considered. The exponential and Gaussian kernels exhibit the worst and best performance for

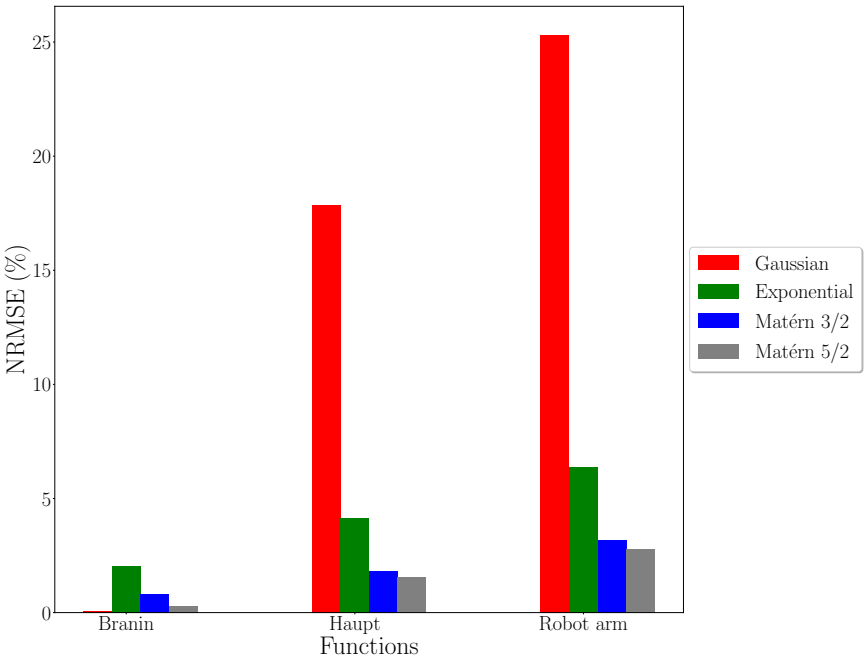


Figure 4.1: Choice of kernel can be problem dependent.

this function, respectively. The Matérn 3/2 kernel comes in right behind the Matérn 5/2 kernel as the third-best performer.

For the Haupt function, both of the Matérn kernels perform well and have comparable model accuracies, as demonstrated in their respective NRMSE values, which are less than 2%. With the highest modeling error of about 17.5% in this case, the Gaussian kernel is revealed to have the poorest model performance. With an NRMSE of about 4.75%, the exponential kernel has a better performance compared to the Gaussian kernel, although lower than those of the Matérn kernels.

The benchmark results in the robot arm case are very comparable to that of the Haupt function. Here, the Matérn kernels similarly possesses the lowest NRMSE's values, while the Gaussian kernel exhibits higher values of NRMSE. Even though the performance trend is similar to that of the Haupt function, the high values of NRMSE demonstrate poor performance of the candidate kernels in the robot arm case, and the largest discrepancy is observed in Gaussian kernel. Therefore, the model trained using the Gaussian kernel has an NRMSE of about 25%, which illustrates the kernel's inappropriateness in this particular situation. The function's response surface's shape, as shown in Figure A.2b, may be a plausible factor in the worse performance exhibited by the Gaussian kernel.

Generally, it is observed that the Gaussian kernel performs best with the Branin function and the worst with the other two functions. As a result, extrapolating the performance of the Gaussian kernel based on how well it fits a particular problem may result in the very unreliable models for other situations.

4.3.2 Effect of sample size on kernel selections

In the second experiment, based on the Haupt function, we compare the performance of the four considered kernels using various sample sizes. We vary the sample size for training the model from 20 to 40 with an increment of 5. The samples are drawn using the Halton sequence within the Haupt function's specified input space. The model is then trained using the considered kernels, followed by validation using 200 test points selected at random.

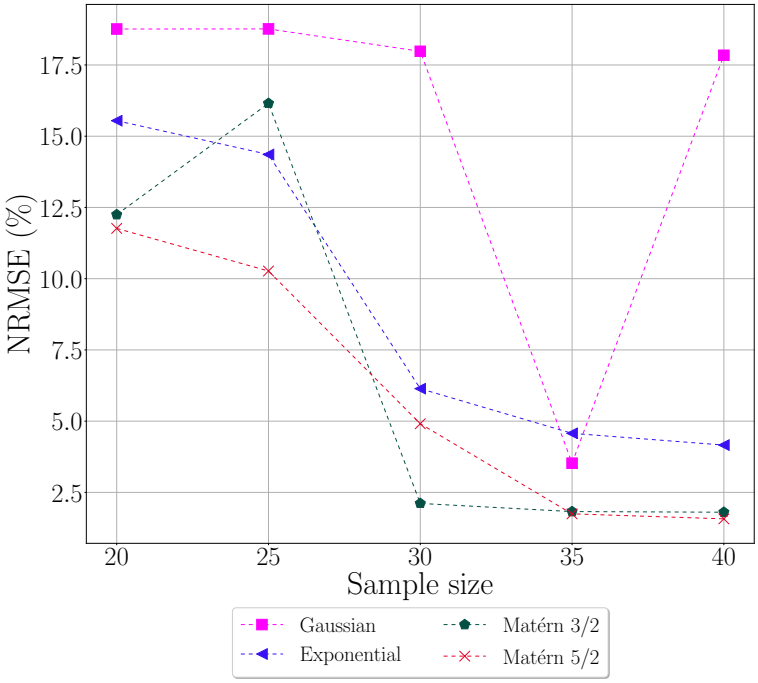


Figure 4.2: Choice of kernel can be dependent on sample size.

From Figure 4.2, it can be deduced that the Gaussian kernel's performance gradually increases when additional samples are introduced, up to 30 sample points. At 35 sample points, there is a significant improvement, with the error dropping to as low as 3.75%. A decline in performance is then seen with an increase in sample size from 35 to 40. When

the training point for the Matérn 3/2 kernel is raised from 20 to 25, a similar decline in model performance is exhibited, after which the model's performance then picks up again, although only little improvement can be observed after 30 samples. The performance of the model did not significantly improve after 30 training points were added. The exponential kernel's performance and that of Matérn 5/2 are comparable, as improvement can be observed with increment in sample points from 20 to 35, after which no significant improvement is added with increasing sample size. In general, as illustrated in Figure 4.2, the performance of a kernel may either improve or degrade with an increment in the sample size. As a result, choosing a kernel based on its strong performance with a certain sample size does not indicate or guarantee that the performance will remain strong under different sample size conditions.

4.3.3 Sensitivity of kernels to hyperparameter values

In building a model, choosing an hyperparameter is crucial, and the kernel function still heavily depends on it. We note that kriging is sensitive to the selection of hyperparameters. Choosing the incorrect hyperparameter could result in significant misspecification (i.e., models' poor performance). The hyperparameter is typically estimated using the techniques covered in Section 2.3.2. However, it is not always essential to utilize a lot of computational resources for model training. The length scales are the primary hyperparameters influencing a kriging model. The final accuracy and applicability of the kernel are affected by these parameters. The selection of an appropriate length scale range for kriging models is still being studied in the field of surrogate modeling [100]. To evaluate the sensitivity of various kernels to these hyperparameters, a systematic approach is presented in this section, based on some algebraic functions. Specifically, utilizing the four kernels covered in Section 2.3.1, we investigate the impact of the hyperparameter on various functions with differing function profiles.

We explore employing algebraic problems with profiles having distinctive attributes to study the patterns in intricate problems. For simplicity of understanding, we only include one- and two-dimensional problems in the test cases. We employ six different one-dimensional algebraic functions with differing attributes for the investigation. The considered functions comprise algebraic problems with profiles that may be *contrasting*,

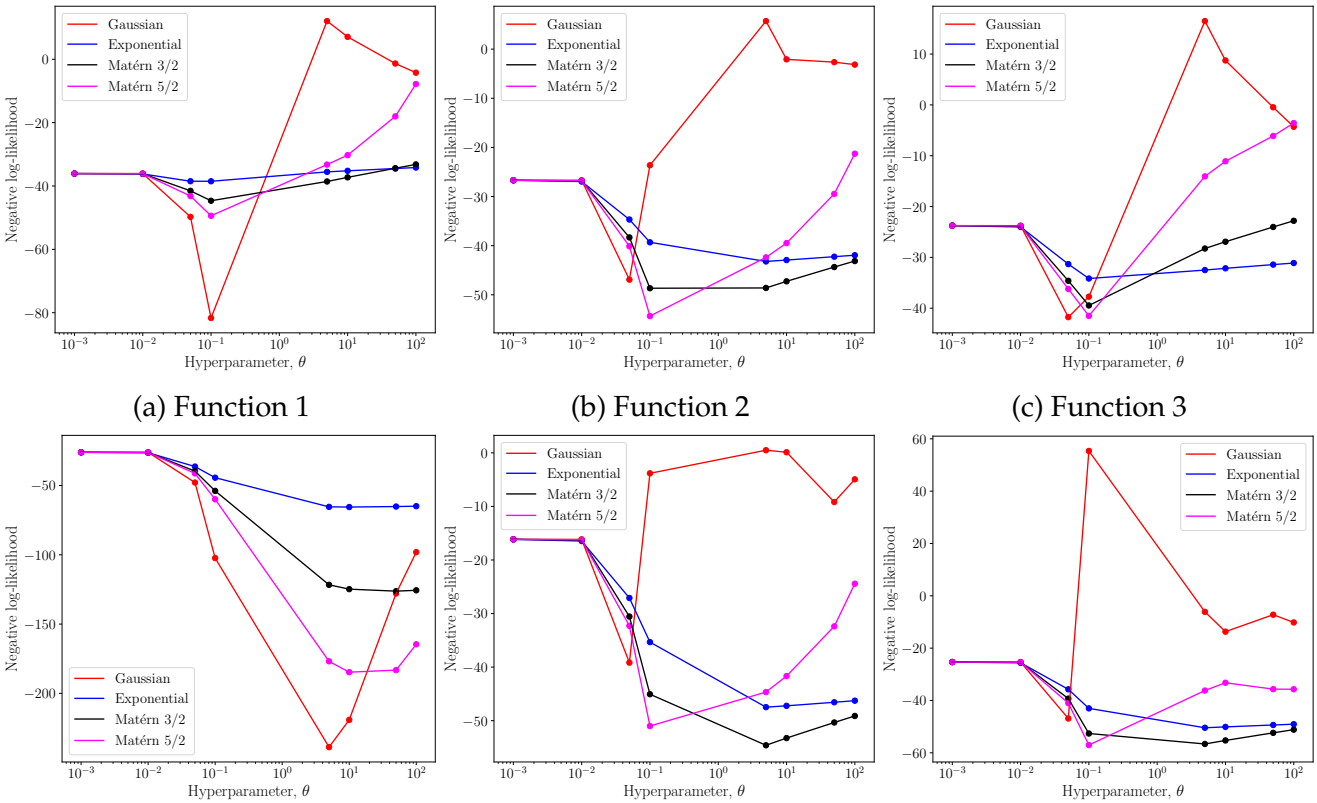
constrained or *symmetrically-constrained*, which are designated as Function 1, 2, and 3, respectively, for convenience. Appendix A.11, A.12 and A.13 provide more details regarding these three functions. The one-dimensional analysis also makes use of the cantilever beam, tensor product hyperbolic tangent (TPHT), and robot arm functions. On the one hand, it is well known that the cantilever beam problem has a linear profile which is presumed to be smooth. In Appendix A.4, more details on the cantilever beam problem are provided. On the other hand, it is established that the TPHT and robot arm functions are non-linear, with the robot arm function having vertical symmetry. More details regarding additional functions employed in this investigation are provided in the appendix.

Furthermore, we examine how the values of the hyperparameters affect the estimate of the negative log-likelihood (NLL). The value estimated by Equation 2.11 is negated to obtain the NLL for each problem. The impact of hyperparameter values on 1D model performance is also investigated in this study. The performance of each model is then adjudged using NRMSE.

Effect of hyperparameter value on the likelihood estimate in one-dimensional problems

The sensitivity of NLL estimate to variations in the hyperparameter values for the six 1-D algebraic problems that were introduced is examined herein. For the development of each model, a large input space $\theta \in [10^{-4} - 10^{2}]$ is defined, and 25 sample points are generated using the Halton sequence [105]. We evaluate the function using the generated sample points to get the equivalent response $\mathbf{y}$. The correlation matrix is then constructed using the various kernels, and the NLL estimate is evaluated by negating Equation 2.11. The experiment's results are shown in Figure 4.3.

According to Figure 4.3d, the NLL values are between -70 and -250. With the exception of the Gaussian kernel, the NLL of the kernels tends to decrease as the hyperparameter values increase. Some level of convergence for all kernels is observed at some point, which persisted for a significant hyperparameter range. Beyond the optimum points for the Gaussian kernel's hyperparameter value, further increases in the hyperparameter value simply made the model worse.A similar trend is seen with the kernels in the Function 1, Function 2, Function 3, TPHT, and Robot arm plots, as depicted in Figures 4.3a, 4.3b, 4.3c,



(a) Function 1 (b) Function 2 (c) Function 3

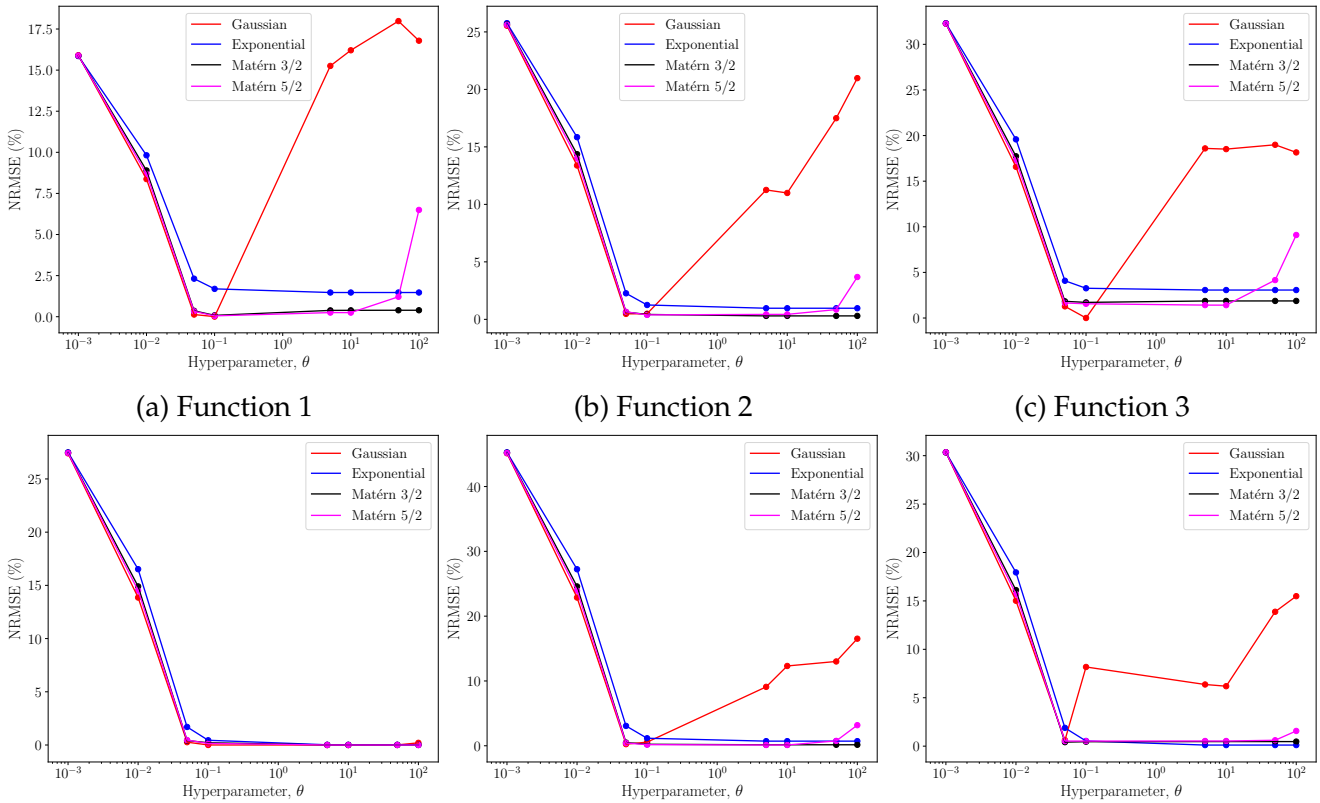
(d) Cantilever beam (e) TPHT (f) Robot Arm

Figure 4.3: Effect of hyperparameter values on likelihood estimates.

4.3e and 4.3f respectively. It is noteworthy to state that the Matérn 5/2 kernel exhibits a trend comparable to that of the Gaussian kernel. As illustrated in Figure 4.3c, in certain instances, the NLL value estimated from the Matérn 3/2 got worse as the hyperparameter value increased, after the optimum point. Although the estimate of the NLL is expected to be positive, this might not always be so. In some circumstances, as shown in Figures 4.3c and 4.3f, the Gaussian kernel exhibits positive NLL within a given range of the hyperparameters. As a result, when defining the hyperparameter range for a particular problem, proper attention must be paid to this phenomenon.

Effect of hyperparameter on model performance for different function profiles

After studying the sensitivity of the NLL values to changes in the hyperparameter values, we now seek to assess how changes in the value of hyperparameters for 1-D problems affect the performance of the model. We investigate how the model performance estimate responds to variations in the hyperparameter values for the six 1-D algebraic problems that were first presented. For the development of each model, we define a large input space $\theta \in [10^{-4} - 10^{2}]$ and generate 25 sample points using the Halton sequence [105]. We then evaluate the function using the generated sample points to get the equivalent response $\mathbf{y}$. Using Equations 2.9 and 2.10, we obtain the corresponding model parameters by constructing the correlation matrix based on the various considered kernels. Afterwards, we examine each functional response utilizing 100 generated points for validation purposes. The cross-correlation matrix is then calculated, and the outputs from each of the validation points are obtained by evaluating Equation 2.1. We estimate the model performance by utilizing the NRMSE, after obtaining the validation points' outputs. The results of this experiment are shown in Figure 4.4. As illustrated in Figure 4.4d, all kernels, including the Gaussian kernel, exhibit good performance as expected in the cantilever beam problem regardless of the hyperparameter value. However, amongst all the evaluated kernels, the Gaussian kernel has a divergent performance in all other considered problems, with the model even getting worse as the hyperparameter values stray away from the optimum point.



(a) Function 1 (b) Function 2 (c) Function 3

(d) Cantilever beam (e) TPHT (f) Robot Arm

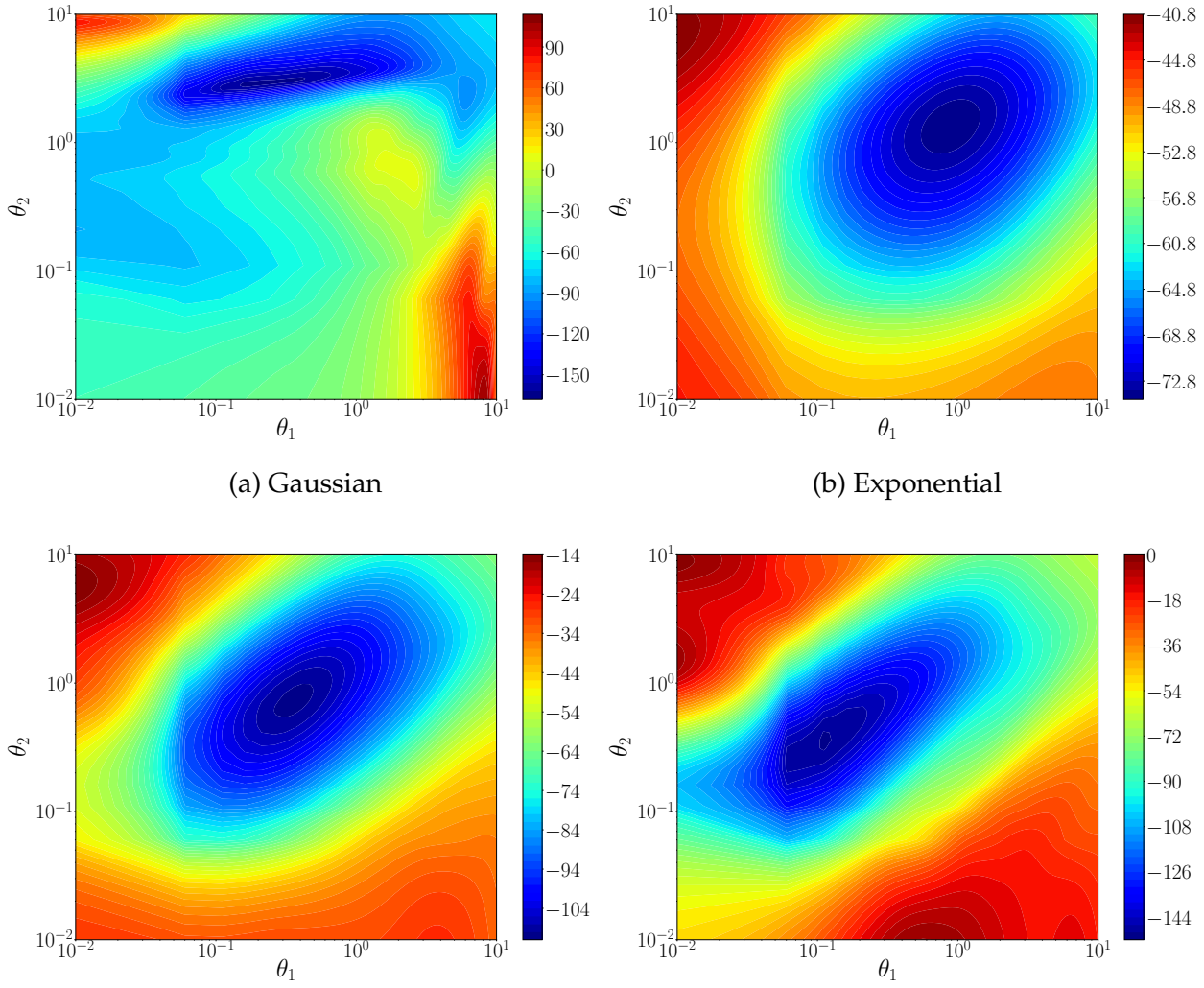
Figure 4.4: Effect of hyperparameter values on model performance.

Effect of hyperparameter value on the likelihood estimate in two-dimensional problems

Using one-dimensional functions, we examine the impact of hyperparameter value on the likelihood estimate in Section 4.3.3. In order to better understand how kernels behave in situations with multiple dimensions, we broaden the study here to two-dimensional problems. To examine how the NLL estimate responds to changes in the hyperparameter values, we analyze eight 2-D problems in this study. For the development of each model, we define a large input space $\theta \in [10^{-2} - 10^{1}]$ and generate 40 sample points using the Halton sequence [105]. We evaluate the function using the generated sample points to get the equivalent response $\mathbf{y}$. The correlation matrix is then constructed using the various kernels, and the NLL can then be obtained by taking the negative value of Equation 2.11. The results of this experiment are showcased using contour plots. θ_1 and θ_2 are plotted on the x- and y- axes of each plot, respectively, and the colour bar represents the negative NLL values.

Figures 4.5- 4.12 demonstrate that, in addition to the previously highlighted concerns, the choice of an initial guess will also have an impact on the convergence to the optimal solution. For the Branin and cantilever functions, as depicted in Figures 4.5 and 4.6 respectively, this phenomenon might not be immediately apparent. The solution space in their case is sightly convex. The Gaussian kernel and occasionally the Matérn kernels, however, deviate from convex behavior when we examine the more complex problems. In those circumstances, a clear division of the solution space into several (often two) different zones with asymmetric optimum behavior can be observed. It is abundantly evident that these are local minima zones divided by sub-optimal ridges. The solution converges to the nearest of the local minima depending on which side of the ridge the initial guess is selected. If the initial guess lies on the ridge side with a local minima which is not global, the converged solution is then sub-optimal.

Essentially, it can be difficult to select the most appropriate kernel for a given task. According to the findings above, a variety of factors may influence the kernel of choice. The effectiveness of a model developed using a certain kernel may vary depending on the problem itself. When utilized on a different task, a particular kernel that performs well for one task may perform poorly or even worse for others. As demonstrated in Figure 4.2,



(a) Gaussian

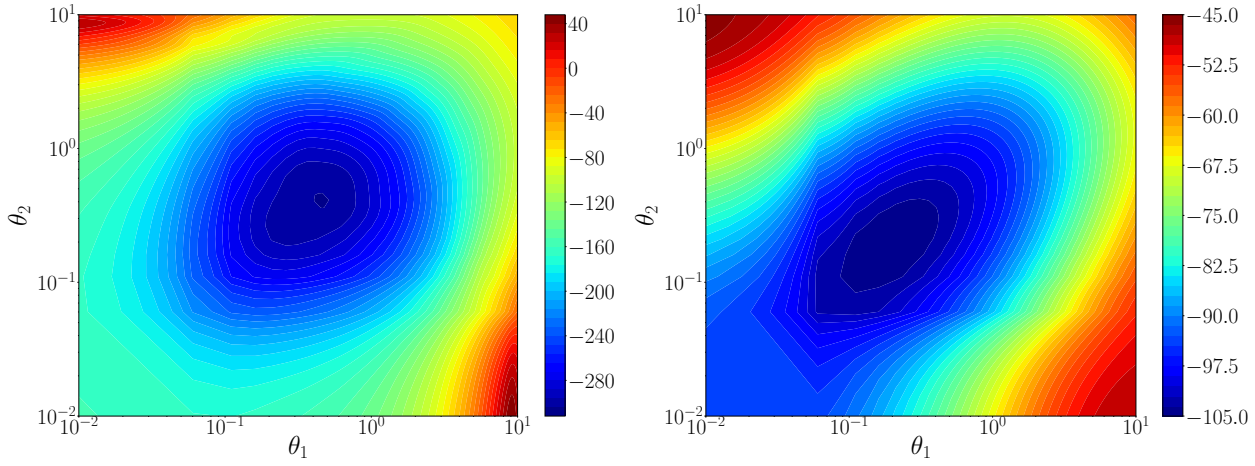
(b) Exponential

(c) Matérn 3/2

(d) Matérn 5/2

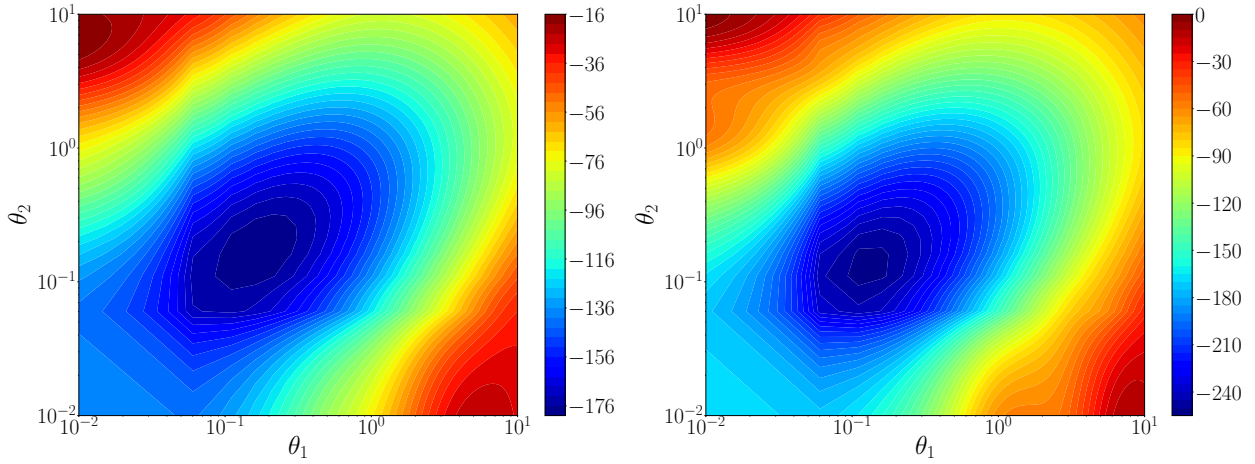
Figure 4.5: Effect of hyperparameter values on the likelihood estimate of Branin function.

the selection of a kernel may also depend on the size of the problem's sample. As a result, caution must be taken when applying kernel-based techniques in applications where new sample points are regularly introduced. Perhaps by strategically adding more sample points using adaptive sampling techniques [159, 65, 147, 276], an improved model could be ensured. However, as it often entails creating a surrogate model and optimizing a predetermined acquisition function for each additional point added, implementing strategies of this sort could prove to be computationally intractable. Additionally, we were able to demonstrate that some kernels, most notably the Gaussian kernel, are more sensitive to hyperparameter values than others.



(a) Gaussian

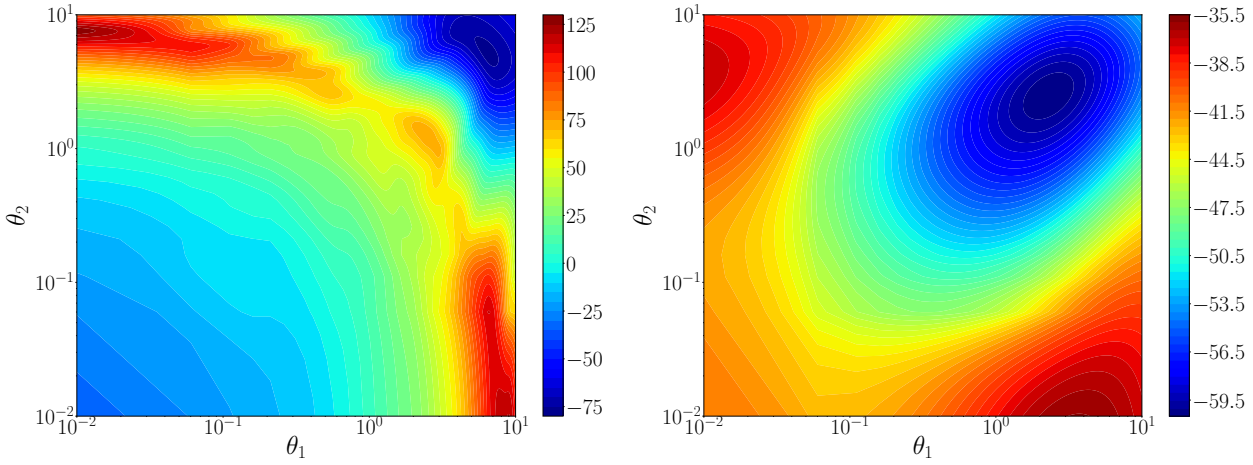
(b) Exponential



(c) Matérn 3/2

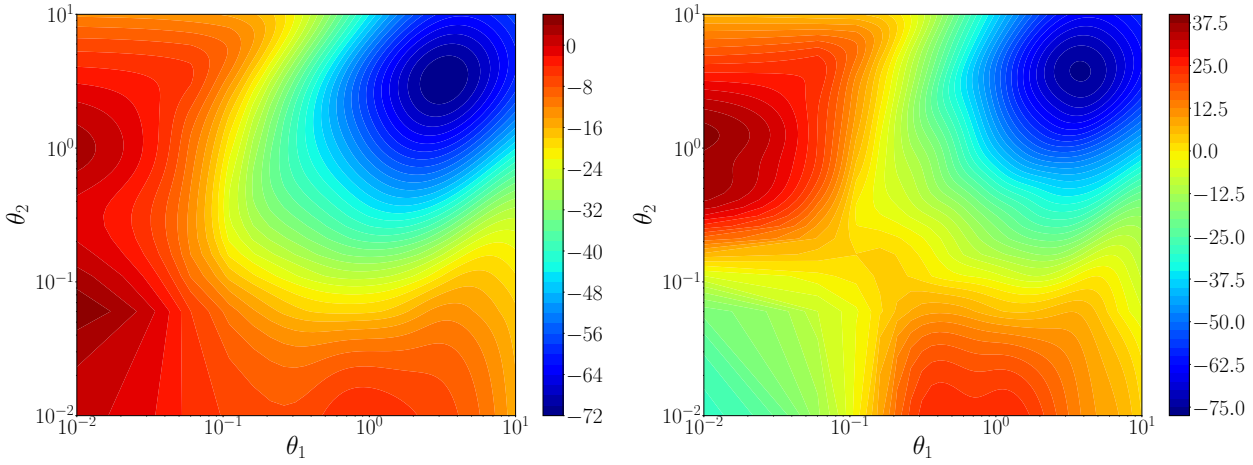
(d) Matérn 5/2

Figure 4.6: Effect of hyperparameter values on the likelihood estimate of cantilever beam function.



(a) Gaussian

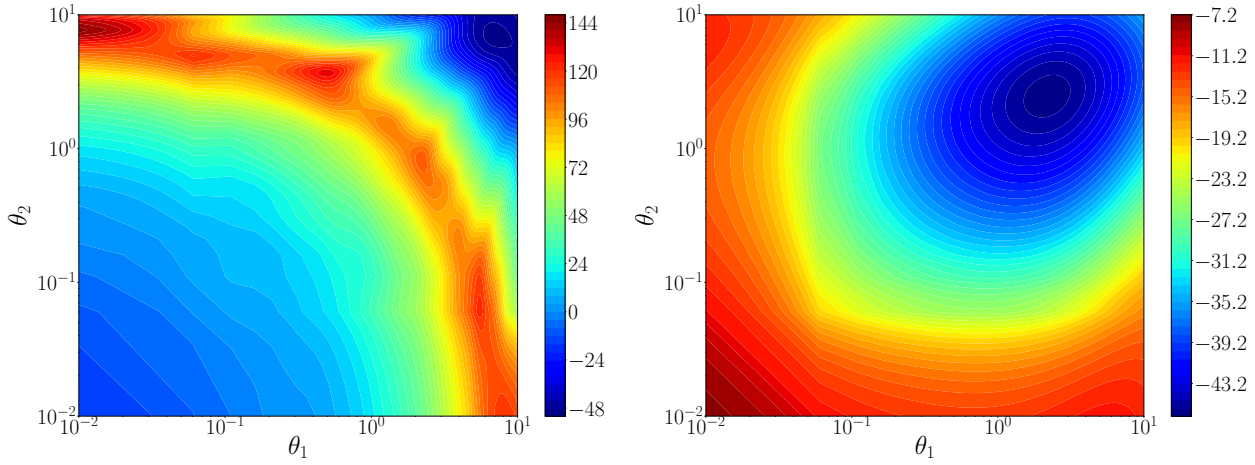
(b) Exponential



(c) Matérn 3/2

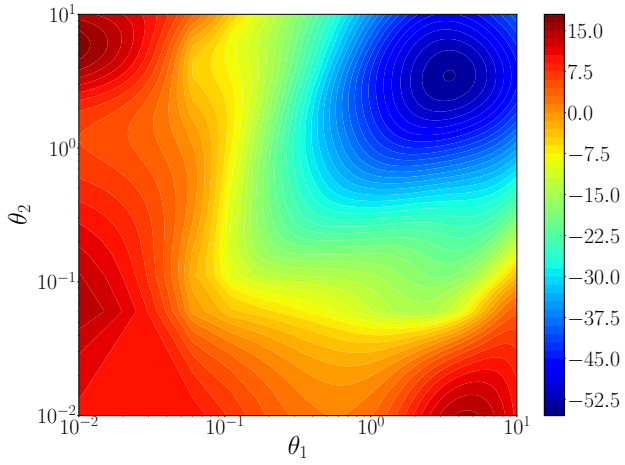
(d) Matérn 5/2

Figure 4.7: Effect of hyperparameter values on the likelihood estimate of robot arm function.

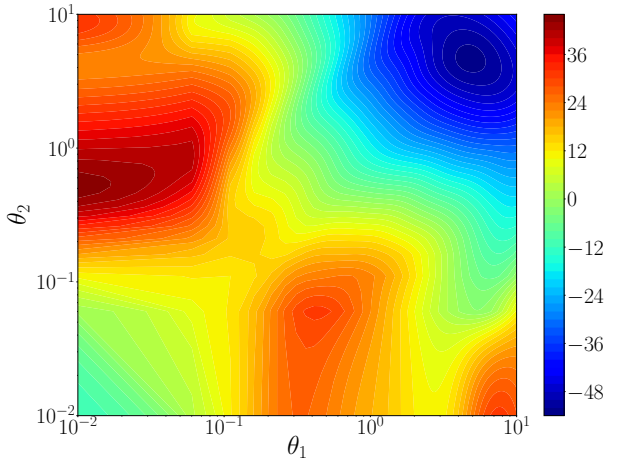


(a) Gaussian

(b) Exponential

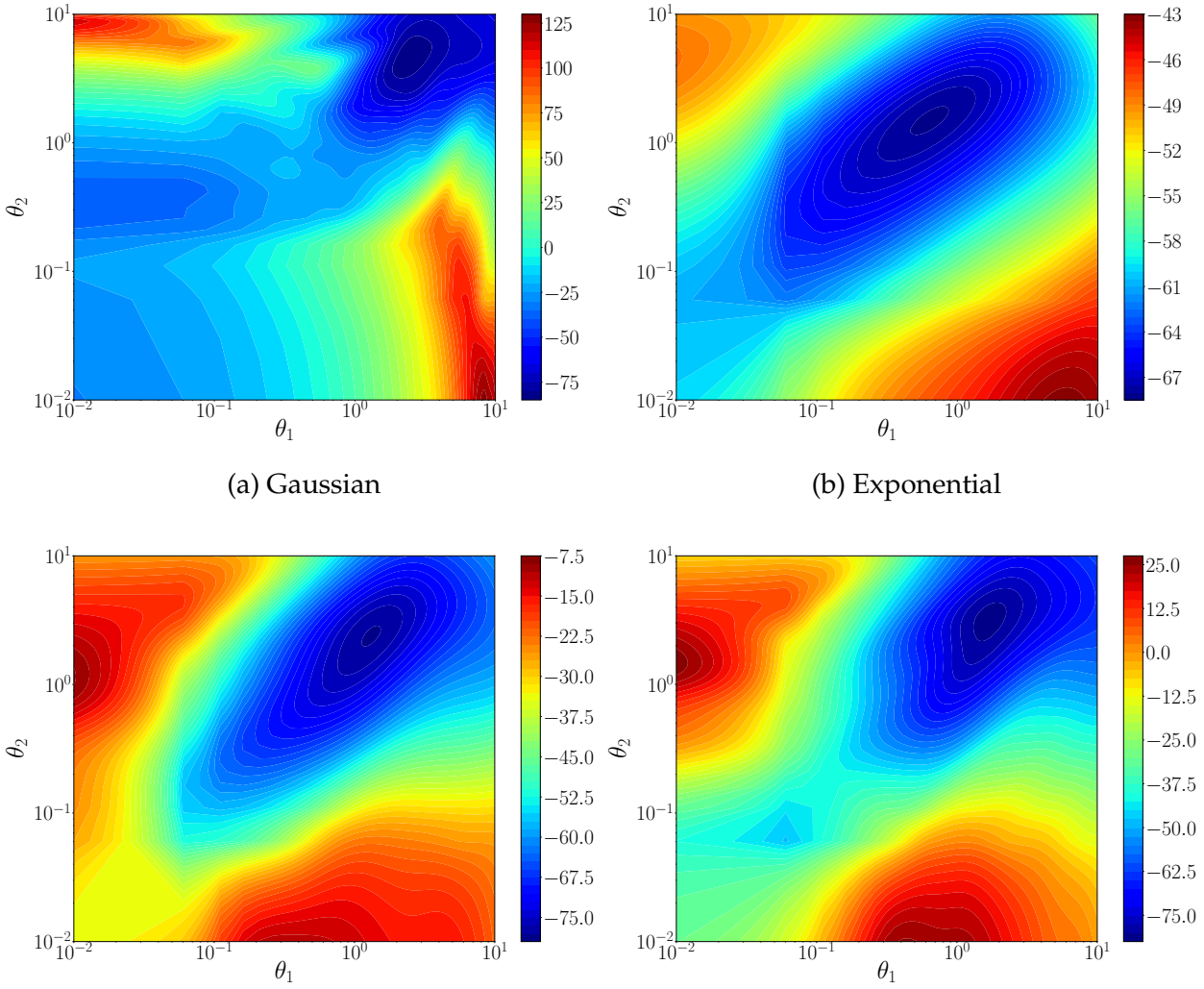


(c) Matérn 3/2



(d) Matérn 5/2

Figure 4.8: Effect of hyperparameter values on the likelihood estimate of TPHT function.



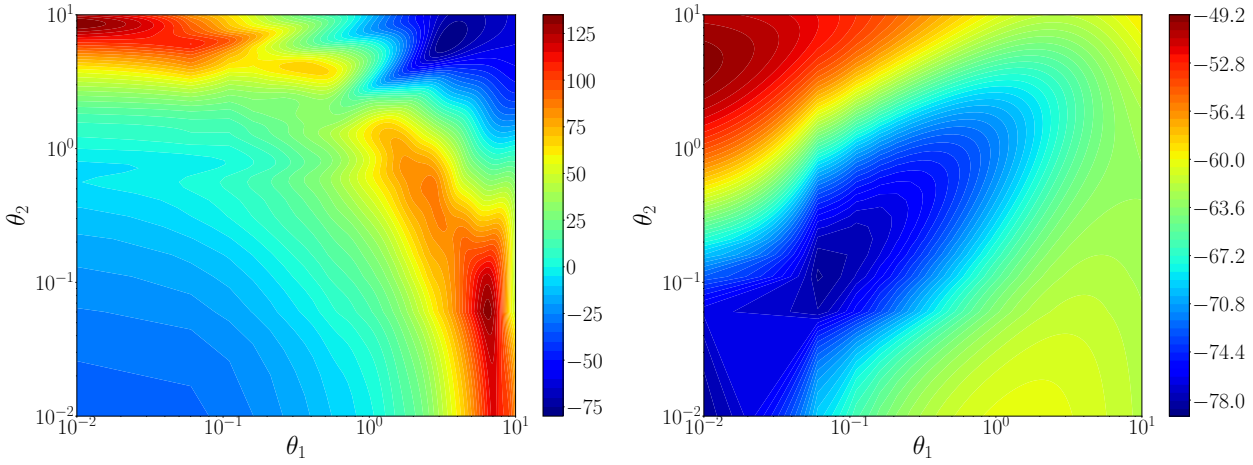
(a) Gaussian

(b) Exponential

(c) Matérn 3/2

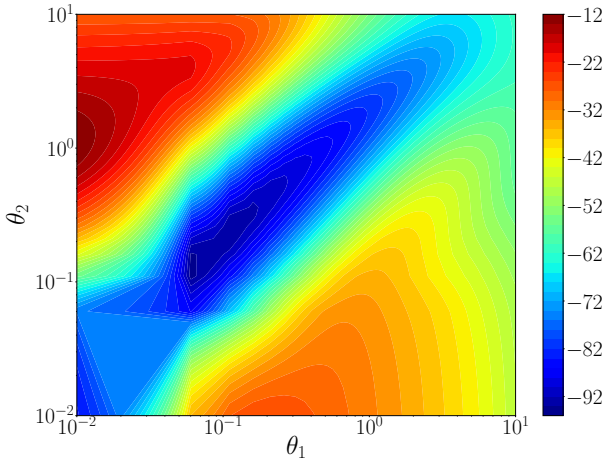
(d) Matérn 5/2

Figure 4.9: Effect of hyperparameter values on the likelihood estimate of Hosaki function.

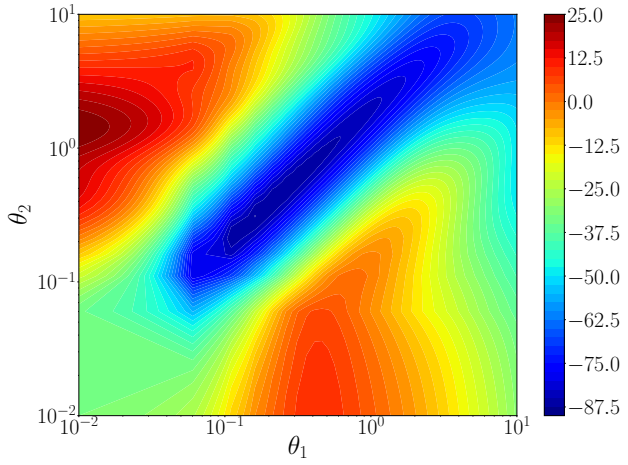


(a) Gaussian

(b) Exponential

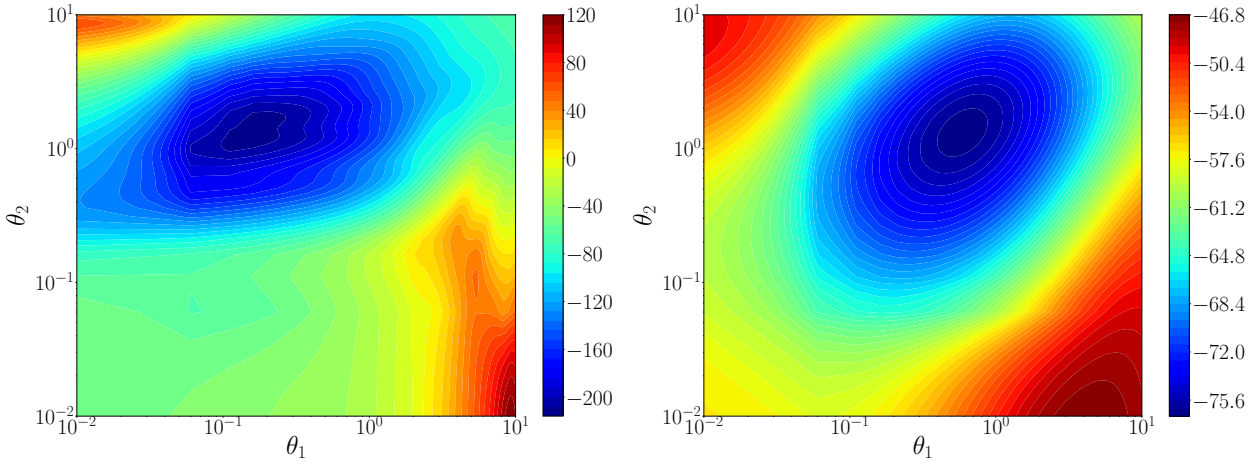


(c) Matérn 3/2



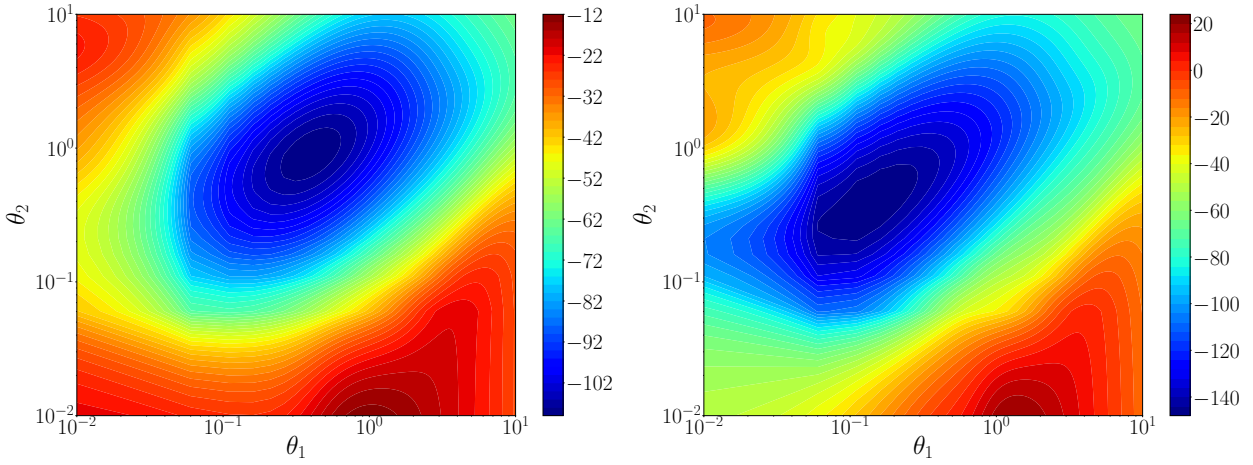
(d) Matérn 5/2

Figure 4.10: Effect of hyperparameter values on the likelihood estimate of Haupt function.



(a) Gaussian

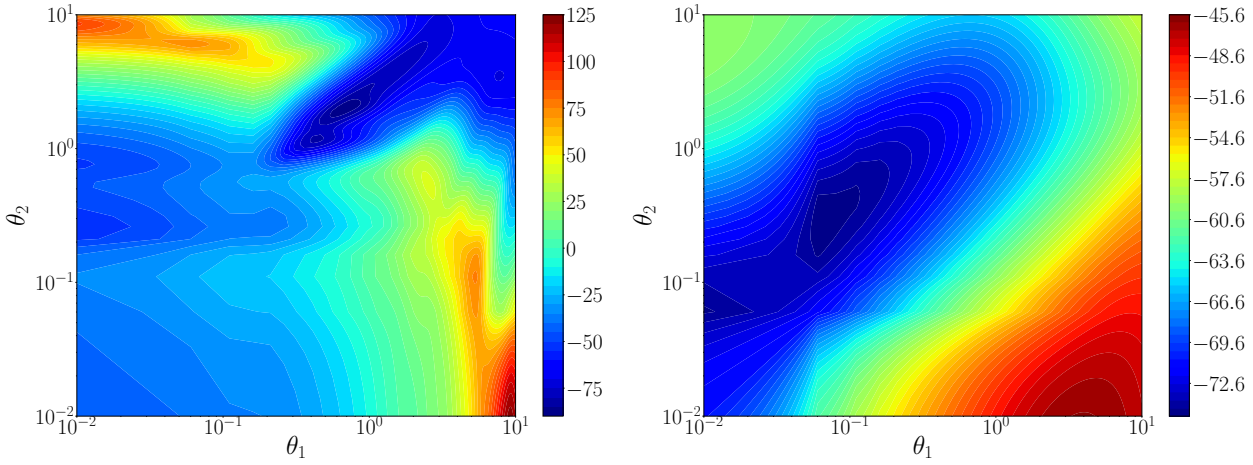
(b) Exponential



(c) Matérn 3/2

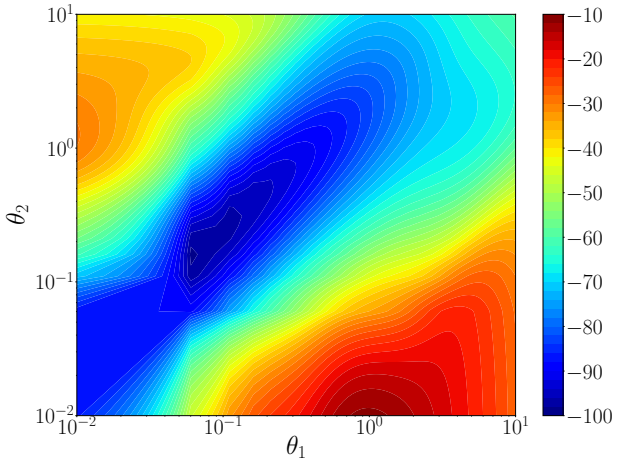
(d) Matérn 5/2

Figure 4.11: Effect of hyperparameter values on the likelihood estimate of Rosenbrock function.

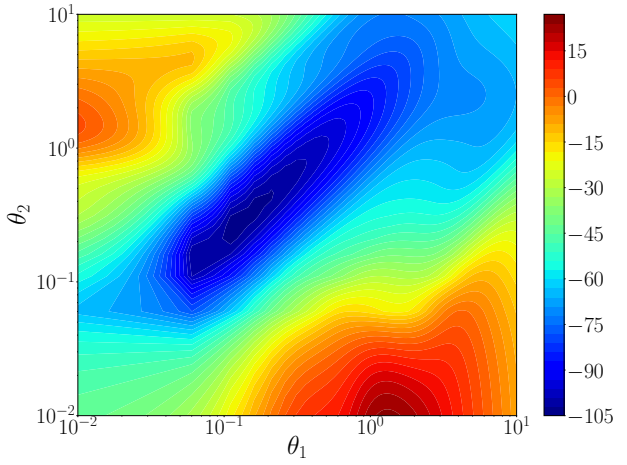


(a) Gaussian

(b) Exponential



(c) Matérn 3/2



(d) Matérn 5/2

Figure 4.12: Effect of hyperparameter values on the likelihood estimate of camel back function.

4.4 Proposed Methods

The findings and inferences discussed in Section 4.3 imply that the Gaussian kernel is typically more sensitive to the selection of the hyperparameters. Additionally, it has been found that the kernel's sensitivity grows with the problem's complexity. The requirement to use an optimization procedure to find the hyperparameter values that optimize the likelihood function, even in one-dimensional problems, is a direct result of the behavior of the Gaussian kernel. Using basic optimizers like the pattern search may produce less-than-ideal results when the complexity of the problem to be modeled increases. An indication of this can be observed in NLL plots of the robot arm, TPHT and Haupt functions shown in Figures 4.7a, 4.8a and 4.10a respectively. The choice of initial points has an impact on the optimal hyperparameters obtained when using pattern search or some other popular optimization method. Despite this restriction, there are two main reasons why the usage of the Gaussian kernel for real-world problems cannot be totally neglected. There are still numerous applications where it is crucial to employ the Gaussian kernel. For instance, it is not uncommon that the Gaussian and exponential kernels work best for the kriging with partial least squares (KPLS) method [282, 29, 31], which can model high-dimensional problems efficiently with little or no loss in accuracy. Bouhlel *et al.* [30] stated that the KPLSK model, which is an upgrade of the KPLS model, is best suited to the Gaussian kernel. Moreover, it is possible that model performance remains unenhanced when one explicitly select alternative kernels over the Gaussian kernel. As shown in Figures 4.7d and 4.8d, complex cases like the Robot arm and TPHT functions are seen to exhibit similar intricacies in the Matérn 5/2 kernel's likelihood profile. It has been demonstrated that in these situations, the intricacy restricts the performance of the kernel. In light of these factors, effective techniques to enhance the stability and efficiency of the Gaussian kernel in challenging situations are desired.

Two techniques are proposed for automatic selection of kernels in kriging models, with the goal of either enhancing the Gaussian kernel's performance or completely avoiding it in cases where improvement might be impossible. First, we propose the optimal-ν method. For this method, the general Matérn kernel, which serves as the foundation of widely-used kriging kernels among engineers, is used for pre-training purposes. In fact, all four main kernels discussed in Section 2.3.1 are derived from the Matérn equation.

Equation 4.1 gives the mathematical expression of the general Matérn equation. Other kernels can be obtained by varying the values of ν. The ν-values and their respective kernels generated from the Matérn equation are presented in Table 4.1. For our method, we propose optimizing ν at the same time with other kriging hyperparameters. The upper limit (UB) of ν, is set to 3.0. The reason for putting UB at this value is to ensure that it is sufficiently above the ν-value for the Matérn 5/2 kernel, and below the ∞ value required for the Gaussian kernel. The lower limit (LB), on the other hand is set to 0.5, which is the exponential kernel. This is done to make our method efficient. The Gaussian kernel is employed for training the model when the absolute difference between the optimal ν and the upper limit ν_{UB} is smaller than a predetermined value ($\epsilon < 0.05$). The best hyperparameter from the first step is used for initialization with the Gaussian kernel in order to minimize the model training time as well. This proposed approach is outlined in Algorithm 4, and Figure 4.13 illustrates how model training is achieved.

$$\kappa(h, \theta, \nu) = \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(2\sqrt{\nu}\frac{|h|}{\theta}\right)^{\nu} K_{\nu} \left(2\sqrt{\nu}\frac{|h|}{\theta}\right) \tag{4.1}$$

where $\nu \geqslant 1/2$ is the shape parameter, Γ is the Gamma function, and K_{ν} is the modified Bessel function of the second kind.

Table 4.1: ν for different kernels.

ν	Kernel
$\frac{1}{2}$	Exponential
$\frac{3}{2}$	Matérn 3/2
$\frac{5}{2}$	Matérn 5/2
∞	Gaussian

$$\begin{aligned} \max_{\nu,\theta} \quad & -\frac{N_s}{2}\ln(2\pi) - \frac{N_s}{2}\ln\hat{\sigma}(\nu,\theta) - \frac{1}{2}\ln|R(\nu,\theta)| \\ \text{s.t.} \quad & \frac{1}{2} \leqslant \nu \leqslant \frac{6}{2} \\ & \theta^{(d)} \geqslant 0 \end{aligned} \tag{4.2}$$

The hyperparameters that need to be tuned in some machine learning methods typically include a combination of continuous, discrete, and categorical variables. The hy-

Algorithm 4 Optimal-ν method.

Input: $\mathbf{x}$, $\mathbf{y}$, $\mathbf{X}$
Output: θ^f, ν, $\hat{\mathbf{y}}$

Initialize θ, ν
Assemble $\mathbf{R}(\theta) = \prod_{d=1}^{N_d} \kappa\left(\theta^{(d)}, \nu\right)$
Obtain ν^{opt}, θ^f by maximizing the likelihood estimate (Equation 4.2) using the constrained Nelder-Mead algorithm.
if $||\nu^f - \nu_{UB}|| \geqslant \epsilon$ **then**
    $\nu^f \leftarrow \nu^{opt}$
else
    $\nu^f \leftarrow \infty$
    Obtain θ^f by maximizing the likelihood estimate (Equation 2.11) using constrained Nelder-mead algorithm.
end if
Assemble $\mathbf{R}(\theta) = \prod_{d=1}^{N_d} \kappa\left(\theta_d^f, \nu^f\right)$
Compute the kriging weights $\boldsymbol{\omega} = \mathbf{R}^{-1}(\mathbf{y} - \boldsymbol{\mu})$
Estimate $\hat{\mathbf{y}} = \boldsymbol{\mu} + r'(\mathbf{X}) \cdot \boldsymbol{\omega}$

perparameters used in kriging are typically continuous, and their optimization is carried out within a predetermined search space. We propose the Optuna-κ approach in which the kernel is introduced as a variable that may be optimized within the kriging framework. With this strategy, kernel selection and model training are done concurrently. To accomplish this, the predefined search space is provided for the values of the continuous hyperparameters together with a kernel list which is designated as categories. Nelder Mead and Hooke-Jeeves [179], two widely used algorithms for hyperparameter optimization, are not able to handle the optimization of categorical variables. Therefore, the optimization problem in our approach will be solved using the `Optuna` [9, 6] optimization framework. The Optuna toolbox uses a Bayesian optimization-based algorithm, known as the tree-structured parzen (TPE) as its default algorithm. With the selected framework, we expect to achieve suitable kernel selection, enhancement of model performance, and non-dependence of the kriging model performance on the initial hyperparameter. Our method generates t-distinct kernel and hyperparameter value combinations, where t is the total number of trials run parallelly, while the number of kernel categories is denoted as p. Due to the fact that we only employ low dimensional test cases in this chapter, we utilize $t = 20$. Although one can change the value as desired, it is recommended to use

a somewhat small number in order to achieve model training more quickly. Algorithm 5 provides a summary of the algorithm employed for the model training and prediction.

Algorithm 5 Optimal-κ method.

Input: $\mathbf{x}$, $\mathbf{y}$, $\mathbf{X}$, kernel list $\mathbf{K} = [\kappa_1, \kappa_2, ...\kappa_p]$, t
Output: θ^f, κ^f, $\hat{\mathbf{y}}$

1: Generate t unique combinations
2: $\mathcal{S} = []$; $LL_S = []$
3: **for** each combination **do**
4: Set $\theta \leftarrow \theta_t$
5: Assemble $\mathbf{R}(\theta) = \prod_{d=1}^{N_d} \kappa_t \left(\theta^{(d)}\right)$
6: Obtain $\{\kappa_t^{opt}, \theta_t^{opt}\}$ by maximizing the likelihood estimate (Equation 4.3)
7: Obtain the MLE, LL from Step 6.
8: $\mathcal{S}$.append($\{\kappa^{opt}, \theta^{opt}\}$)
9: LL_S.append(LL)
10: **end for**
11: $\{\kappa^f, \theta^f\} = \arg\max_{\kappa,\theta} LL_S$
12: Assemble $\mathbf{R}(\theta) = \prod_{d=1}^{N_d} \kappa^f \left(\theta^{(d)}\right)$
13: Compute the kriging weights $\boldsymbol{\omega} = \mathbf{R}^{-1}(\mathbf{y} - \boldsymbol{\mu})$
14: Estimate $\hat{\mathbf{y}} = \boldsymbol{\mu} + r'(\mathbf{X}) \cdot \boldsymbol{\omega}$

$$\begin{aligned} \max_{\kappa,\theta} \quad & -\frac{N_s}{2} ln(2\pi) - \frac{N_s}{2} ln\hat{\sigma}(\kappa, \theta) - \frac{1}{2} ln|R(\kappa, \theta)| \\ \text{s.t.} \quad & \theta^{(d)} \geqslant 0 \end{aligned} \tag{4.3}$$

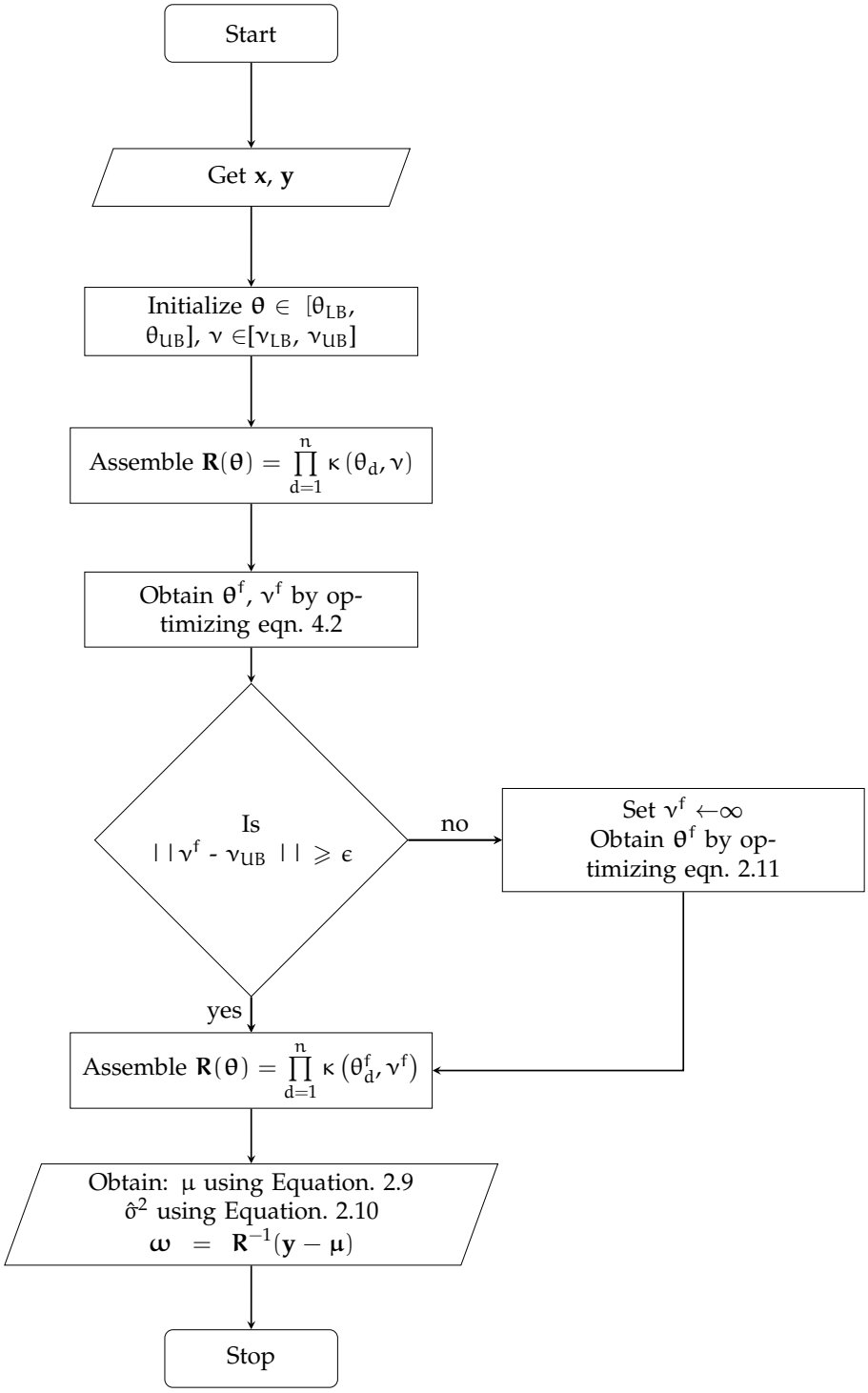


Figure 4.13: Flowchart showing the use of proposed Optimal-ν method in model training.

4.5 Test cases

We benchmark our proposed method for choosing kernels using both algebraic functions and a real-world problem, in order to assess its performance. We specifically investigate three 2-D algebraic, three 3-D algebraic, and two real engineering problems. The 2-D problems are the same as those utilized in Section 4.3 previously, and are the Branin, TPHT, and the Haupt functions. Descriptions of the considered problems are presented in the Appendix. We select the shear stress from a welded-beam [57], the water flow through a borehole [177], and the Hartmann three-dimensional function, as the three-dimensional algebraic problems to be analyzed. The axial transonic rotor scenario with NASA rotor 37 [191] is considered to represent a real-world engineering problem.

4.5.1 3-D Welded Beam Problem

One common example of complex design problems that occur in structural engineering is the welded beam design problem [178]. It entails designing the form of steel beams as well as their connections to create intricate structures. This problem, as displayed in Equation 4.4 is a 3-D function of the shear stress developed in a welded beam, with the height, and length and thickness of the beam as the input variables. In Table 4.2, the variables and output are fully described.

$$\tau = \sqrt{\tau'^2 + \tau''^2 + \frac{l\tau'\tau''}{\sqrt{0.25(l^2 + (h+t)^2)}}} \tag{4.4}$$

where,

$$\tau' = \frac{6000}{\sqrt{2}hl}$$

$$\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2[0.707hl(l^2/12 + 0.25(h+t)^2)]}$$

Table 4.2: 3-D Welded Beam Problem.

Input/Output	Quantity	Description
y_1	τ	Shear stress
x_1	$5 \leqslant t \leqslant 10$	Beam thickness
x_2	$0.125 \leqslant h \leqslant 1$	Beam height
x_3	$5 \leqslant l \leqslant 10$	Beam length

4.5.2 3-D Water Flow Problem

The water flow problem is an eight-dimensional function that depicts the amount of water flowing through a borehole [177]. By identifying the three major variables that influence the amount of water in the borehole tank, one can transform the function into a three-dimensional analytical problem. Thus, the length of the borehole, the influence radius, and the borehole's radius are designated as the input variables, while the five other variables such as the hydraulic conductivity of the borehole, the transmissivity of the upper aquifer, potentiometric head of the upper aquifer, transmissivity of the lower aquifer, and potentiometric head of the lower aquifer are fixed to be 10,000 m/y, 70,000 m^2/y, 1,000 m, $80m^2/y$, and 750 m, respectively. Equation 4.5 shows the analytical form of this function; while the full explanations of the variables and its output are presented in Table 4.3.

$$f = \frac{2\pi T_u (H_u - H_l)}{\ln(r/r_w)\left[1 + \frac{2LT_u}{\ln(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right]} \tag{4.5}$$

4.5.3 Hartmann 3D problem

The Hartmann three-dimensional function is an algebraic function commonly evaluated in the domain: $x_i \in [0, 1]$. There are four local minima and a global minimum reported for this function. Its global minimum is achieved at point $\mathbf{x}^* = (0.1146, 0.5556, 0.8525)$, such that $f(\mathbf{x}^*) = -3.86278$. Equation 4.6 presents this function's analytical form.

Table 4.3: 3-D Water Flow Problem.

Input/Output	Quantity	Description
y_1	f	Water flow
x_1	$0.05 \leqslant r_w \leqslant 0.15$	Radius of borehole (m)
x_2	$100 \leqslant r \leqslant 50000$	Radius of influence (m)
x_3	$1120 \leqslant L \leqslant 1680$	Length of borehole (m)
	$63070 \leqslant T_u \leqslant 115600$	Transmissivity of upper aquifer (m^2/y)
	$990 \leqslant H_u \leqslant 1110$	Potentiometric head of upper aquifer (m)
	$63.1 \leqslant T_l \leqslant 116$	Transmissivity of lower aquifer (m^2/y)
	$700 \leqslant H_l \leqslant 820$	Potentiometric head of lower aquifer (m)
	$9855 \leqslant K_w \leqslant 12045$	Hydraulic conductivity of borehole (m/y)

$$f(x) = -\sum_{i=1}^{4} \alpha_i exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right), \tag{4.6}$$

where,

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T,$$

$$A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},$$

$$P = 10^{-4} \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

4.5.4 Axial transonic rotor case

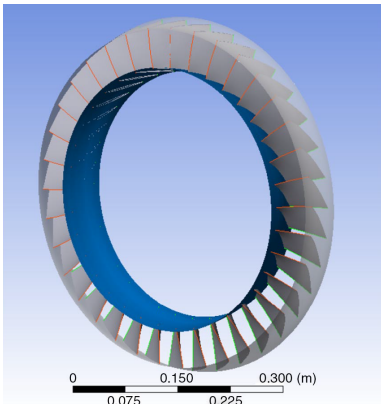
This problem is considered to exemplify a common real-world engineering problem. The 3-D computer-aided design of the axial transonic rotor is shown in Figure 4.14. For this problem, the twist and sweep of the datum are the input variables.

The quantities of interest (QOI) are the overall pressure ratio and adiabatic efficiency. The Reynolds-averaged Navier Stokes (RANS) solver from ANSYS is used to evaluate the

QOIs. The CFD simulations are run for both outputs at a rotor speed of 1.7×10^4 rpm. Latin hypercube sampling (LHS) is used to generate a dataset with 147 samples. The data for this case are obtained from Palar *et al.* [191]. Figure 4.15 (a) and (b), respectively, depict the approximated response surfaces of the adiabatic efficiency and total pressure ratio.

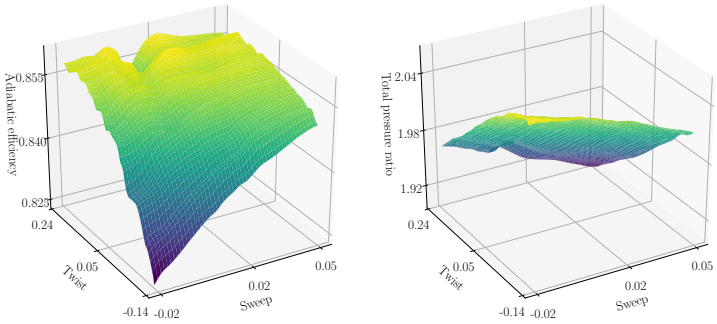
Table 4.4: Value ranges for the axial transonic rotor input variable.

	Lower limit	Upper limit
Twist (radians)	-0.125	0.22
Sweep (radians)	-0.02	0.05



(a) 2D

Figure 4.14: Computer-aided design of the axial transonic rotor [191].



(a) Adiabatic efficiency (b) Total pressure ratio

Figure 4.15: Generated response surfaces of the quantities of interest.

4.6 Results

The results for each benchmark test case described in Section 4.5 are presented in this section. We use two separate criteria (the NRMSE and R^2 score) to compare the performance of the proposed techniques to models trained with single kernels. We benchmark the four single kernels in Section 2.3.1 for the considered problems using the proposed approaches. As for the algebraic test cases, we employ adaptive sampling to select the optimal sample points. In the adaptive sampling technique employed, in line with the recommendation in Jones *et al.* [125], initial sample sizes N_s, of $10N_d$, where N_d is the problem dimension. LHS [48] is used to generate these first sample points. For each problem, new sample points are then continuously added to the initial points until the specified convergence criteria are satisfied. The final sample sizes are 44, 117, 50, 36, 47, and 64 for the Branin, TPHT, Haupt, water flow, welded beam, and Hartmann problems, respectively. We train our kriging models for total pressure ratio (TPR) and adiabatic efficiency in the axial transonic rotor problem using 40 training samples. The models are then benchmarked against a well-defined test set after training. Tables 4.5 and 4.6 present a summary of the results for the algebraic problems. In the real world problem cases, the NRMSE and R^2 values are presented in Tables 4.7 and 4.8, respectively. After each set of benchmarked values, the choice kernels selected by our proposed models are enclosed in parenthesis, in Tables 4.5 and 4.6. We use G, E, M3, and M5 to denote the Gaussian, exponential, Matérn 3/2, and Matérn 5/2 kernels, respectively. The two best-performing models are highlighted in bold text in the benchmarking results.

Table 4.5: NRMSE (%) results for the algebraic test cases.

	Branin (2D)	TPHT (2D)	Haupt (2D)	Water Flow (3D)	Welded Beam (3D)	Hartmann (3D)
Gaussian	0.0739	12.0957	24.2145	**0.0388**	1.7552	1.5491
Exponential	9.8358	2.9078	1.9755	3.1023	4.1305	3.7306
Matérn 3/2	2.2200	**1.4607**	0.5201	0.3642	2.1062	**1.5197**
Matérn 5/2	**0.0469**	15.7560	**0.3450**	0.0924	**1.4122**	20.7910
Optimal - ν method	**0.0651(G)**	3.2820(G)	1.2208(G)	0.054(G)	**1.5677(G)**	**1.5491(G)**
Optimal - κ method	0.0662(G)	**2.5830(M3)**	**0.4686(M5)**	**0.042(G)**	1.8183(M5)	1.5837(M3)

As can be observed from its function profile in Figure A.5, the Branin function is well known for being a smooth function. This is also supported by the likelihood profiles of the function using the four kernels, as shown in Figure 4.5. The Gaussian and Matérn

Table 4.6: R^2 results for the algebraic test cases.

	Branin (2D)	TPHT (2D)	Haupt (2D)	Water Flow (3D)	Welded Beam (3D)	Hartmann (3D)
Gaussian	0.9999	0.8599	0.4008	**0.9999**	0.9919	0.9965
Exponential	0.7919	0.9956	0.9902	0.9835	0.9550	0.9798
Matérn 3/2	0.9894	**0.9989**	0.9993	0.9997	0.9883	**0.9967**
Matérn 5/2	**0.9999**	0.6623	**0.9997**	0.9999	**0.9947**	0.5674
Optimal-ν method	**0.9999(G)**	0.9948(G)	0.9962(G)	0.9999(G)	**0.9935(G)**	**0.9965(G)**
Optimal-κ method	0.9999(G)	**0.9967(M3)**	**0.9994(M5)**	**0.9999(G)**	0.9912(M5)	0.9964(M3)

5/2 kernels outperform the other kernels in this benchmark, with NRMSE estimates of 0.0739% and 0.0469%, respectively. The Gaussian kernel is likewise chosen by the proposed methods, which produces benchmarking results of about 0.066% NRMSE.

The TPHT problem approximates a step function, and is understood to exhibit oscillation when certain surrogate modeling techniques are employed [115]. The likelihood profiles, particularly the Gaussian kernel likelihood profile as illustrated in Figure 4.8a, demonstrate this phenomenon. The likelihood profile of the Matérn 5/2 kernel, as presented in Figure 4.8d, also exhibits similar level of intricacy. The probability that the hyperparameters converge at a local optimum during optimization is increased, as seen in the figures. The performance of both kernels is observed to be impacted by this complexity. Compared to the exponential and Matérn 3/2 kernels, which perform well for this problem, the Gaussian and Matérn 5/2 kernel have R^2 values of 0.8599 and 0.6623, respectively. It is noteworthy that the optimal-ν approach, which does well, also selects to utilize the Gaussian kernel. Our methods' use of informative hyperparameter initialization is responsible for the performance gain. The Matérn 3/2 kernel is chosen for model training by the optimal-κ method, thereby successfully evading the under-performing kernels. After benchmarking, the optimal-κ method results in an NRMSE and R^2 values of 2.5830% and 0.9967, respectively.

The likelihood profile for the Gaussian kernel when used on the Haupt function, as presented in Figure 4.10a is comparable to that of the TPHT problem. The benchmarking analysis indicates that the complexity of the Gaussian kernel's likelihood profile has an impact on the kernel's performance, leading to a poor performance of 24.2145% NRMSE. Other kernels are observed to do rather well, with the Matérn 5/2 performing best and having an NRMSE of about 0.35%. Substantial improvement in model performance is recorded with both optimal-ν and optimal-κ. Specifically, even though optimal-ν chooses

the Gaussian kernel, the performance gain is still fairly large, with a R^2 value of 0.9962 compared to a mere 0.40 with the single kernel benchmark.

For the water flow problem in the three-dimensional test case, all the four kernels perform well. Despite the strong modeling performance, it is shown that the Gaussian kernel performs the best, with NRMSE and R^2 values of 0.0388% and 0.9999, respectively. Both optimal-ν and optimal-κ methods select the Gaussian kernel for the training of their respective models. Also, their performance is similar to that of the single Gaussian kernel. The NRMSE values for the optimal-ν and optimal-κ methods are 0.054% and 0.042%, respectively.

The benchmark test employing the welded beam problem shows the superiority of the Gaussian and Matérn 3/2 kernels. With NRMSE and R^2 values of 1.4122% and 0.9947, respectively, the Matérn 3/2 has the best performance. The optimal kernel choices from the two proposed methods are different. The optimal-ν method suggests utilizing the Gaussian kernel, yielding a better model performance compared to the single Gaussian kernel, by significantly lowering the NRMSE by approximately 10.7%. The Matérn 5/2 kernel is chosen via the optimal-κ method, resulting in a model with an R^2 value of 0.9912. In this benchmark, the exponential and Matérn 3/2 kernels are not selected nor used by our proposed methods.

The Matérn 5/2 kernel performs quite poorly when used in the Hartmann three-dimensional problem. Upon benchmarking, the model trained with the kernel records R^2 and NRMSE values of 0.5674 and 20.79%, respectively. Thus, if the Hartmann 3D function is trained using the Matérn 5/2 kernel, a good generalization cannot be obtained. The proposed methods performed well when compared to single kernel methods in benchmark tests. In this test case, the proposed techniques also managed to evade the underperforming kernels. After training of the model, the optimal-ν method selects the Gaussian kernel, which performs well, with an approximate R^2 value of 0.9965. However, the Matérn 3/2 kernel selected by the optimal-κ method also has a comparable performance level when used for model training with the single kernel models.

We compare the performance of the proposed methods to that of the single kernel methods for the real-world problems considered. The proposed approaches vary in their kernel choices for TPR. During model training, the optimal-ν chooses the Gaussian ker-

Table 4.7: NRMSE (%) results for the real-world test cases.

	Total Pressure Ratio	Adiabatic Efficiency
Gaussian	**0.7786**	1.7655
Exponential	1.1650	2.2920
Matérn 3/2	**0.6906**	2.7961
Matérn 5/2	1.6718	3.6433
Optimal-ν method	0.7793(G)	**1.7639(G)**
Optimal-κ method	0.8067(M3)	**1.6713(G)**

Table 4.8: R^2 results for the real-world test cases.

	Total Pressure Ratio	Adiabatic Efficiency
Gaussian	**0.9989**	0.9943
Exponential	0.9977	0.9904
Matérn 3/2	**0.9992**	0.9857
Matérn 5/2	0.9953	0.9757
Optimal-ν method	0.9989(G)	**0.9943(G)**
Optimal-κ method	0.9989(M3)	**0.9949(G)**

nel, whereas optimal-κ selects the Matérn 3/2 kernel. The R^2 values of the two methods are approximately 0.9989, indicating almost equal performance. This performance is comparable to that of the single kernel benchmark for the Gaussian and Matérn 3/2 kernels. Once more, our methods allow us to train the model with the best-performing kernels, thereby yielding models with better performance.

The Gaussian kernel performs better than other kernels in the adiabatic efficiency problem, resulting in a low NRMSE of 1.7655%. During training, the Gaussian kernel is chosen by both of the proposed methods, yielding NRMSE values of 1.7639% and 1.6713%, respectively, for the optimal-ν and optimal-κ methods. The performance are comparable to that of the Gaussian kernel in the single kernel approach.

4.7 Summary

Throughout this chapter, the challenges that could arise when choosing kernels for kernel-based techniques like kriging were discussed explicitly. In order to better understand the properties of various kriging models when applied to diverse complex problems, we presented several investigations in the foregoing sections. We began by providing the find-

ings from a number of studies centered on kernel choices and selection. We specifically looked at how sample sizes and problem profiles affected the choice of kernels. We also looked into the four typical kernels' sensitivity to hyperparameter values. For each kernel, we examined how the likelihood and error estimates responded to changes in the hyperparameter values. The complexity of the likelihood profile of the Gaussian kernel was demonstrated via some numerical experiments. This intricate profile was demonstrated to restrict the model generalization of the Gaussian kernel in certain scenarios. As observed in Tables 4.5 and 4.6, our benchmark studies also demonstrated instances in which the Matérn 5/2 fared badly. We presented two methods to enhance model performance and investigated the behavior and effectiveness of our proposed methods on both algebraic and real-world test cases. It is important to mention that the computational efficiency of the proposed methods depends on the number of parameters to be trained. For instance, the two-staged optimal-ν method required $N_d + 1$ and N_d parameters to be optimized in the first and second stages, respectively.

We selected three distinct situations from the kernel studies as well as three additional benchmark problems, all of which were three-dimensional, for the algebraic problem benchmark. For the real-world test case, we examined the overall pressure ratio and adiabatic efficiency from the axial transonic rotor problem. Benchmarking our proposed methods against the four kernels revealed consistently good results. The methods successfully evaded underperforming kernels, and in some instances the optimal-ν method greatly enhanced the Gaussian kernel's performance. In the benchmarks carried out, it was also deduced that the Matérn 3/2 kernel had good modeling ability, and as such, would be recommended to be the initial choice of kernel in any problem.

As a general remark, the two approaches are recommended for kernel selections in other kernel-based methods. The first entails identifying a shared parameter across the frequently used kernels, and then optimizing such parameter during the model training process. The second involves framing the hyperparameter optimization for kernel-based algorithms as a mixed-optimization problem. This will make it easier to learn kernels while simultaneously optimizing their hyperparameter values.

CHAPTER 5

EFFICIENT KRIGING MODELS AND KERNEL SELECTIONS FOR HIGH-DIMENSIONAL PROBLEMS

It is not uncommon for real-world problems to be highly-dimensional. The issues with modeling high-dimensional problems is that more training points are needed. Due to this, large computational resources are expended by surrogate methods used for these problems. The amount of time needed to sufficiently train the models limits the use of such models for applications that require real-time training and prediction. Existing methods which help to reduce the computational training time typically sacrifice a certain degree of accuracy. Hence, we propose to develop a new computationally-efficient method based on information theory. With the proposed method, we adequately model high-dimensional problems without notable loss in model accuracy. We compare the performance of the proposed method with the conventional kriging method as well as two other state-of-the-art methods. The benchmark test cases used for the comparison range from 20-D to 80-D. The results show improvement in modeling accuracy with our method in most of the benchmark cases. Afterwards, we systematically study the influence of kernel selections on high dimensional problems. From our extensive studies, we recommend that either the exponential or the Matérn 3/2 kernel be used to model high dimensional problems.

5.1 Introduction

Oftentimes, researchers in engineering communities shy away from clearly defining what makes a problem highly dimensional. This has been the reason behind the discrepancies in recent times. It is important to note that terms differ between communities, including the definition of how many dimensions constitute a high-dimensional problem. In computer science communities, it is common for datasets to have features in several hundreds. A high-dimensional problem is hence defined as a problem that has far more features than

the number of observations [109]. High dimensionality is common in healthcare [260], financial [70], genomics [261] and proteomic [183] problems. In fact, a genomic dataset could have features as much as 1 000 [109]. However, this is not the case in the surrogate modeling and engineering communities, where most datasets have fewer than 100 dimensions. Hence, for the sake of clarity, in this chapter we define a high-dimensional problem as such in which the problem dimension is greater than 15. These types of problems are common among current real-world problems because of several reasons such as growing technological advancements [150]. This has made it easier to capture more different types of data. While this can help with the development of more accurate models, it could also have a significant downside. One notable difficulty is in the estimation of the parameters/hyperparameters that describe the models. Different surrogate models such as SVM [110, 194], kriging [51, 52], radial basis function (RBF) [115] among others have been used to tackle high-dimensional problems.

Among the multitude of surrogate models available, kriging is one of the commonly used methods, especially in engineering applications [196]. The provision of the output variance which can be used to estimate uncertainty has made kriging one of the go-to methods in Bayesian optimization and uncertainty quantification applications [197].

Kernels are essential parts of kriging [191]. There are different types of kernels but they all have basically two components, namely the distance which is a measure of the separation of points $|x - x'|$ and the hyperparameter θ [208]. The multitude of kernels available can either be isotropic or anisotropic [142]. This classification is based on the numbers of hyperparameters in the kernels. For isotropic kernels, only one hyperparameter is used across the problem's variable dimension. Anisotropic kernels have the number of hyperparameters equal to the number of design variables. In essence, each variable has its own hyperparameter. The hyperparameter in this case captures the relevance of the variable to the output. Anisotropic correlation functions are advised when each variable has a distinct physical meaning [108]. This approach gives more flexibility in modeling at the expense of a more complex MLE [53, 154]. While the anisotropic kernel model is advised, there is the issue of training many hyperparameters especially in high dimensional problems. It is important to mention that the optimizer code searches for the optimum hyperparameter values that optimizes a defined criteria. In most cases, the MLE is used as the criteria.

Hence, the task is usually to select the hyperparameter values that maximizes the MLE. In high-dimensional problems, it is expected that this takes a much longer process and time, compared to lower dimensional problems, that have fewer hyperparameter values. Each step of the hyperparameter optimization process requires that the correlation matrix be computed and inverted [14]. The inversion of the correlation matrix is an expensive process. It gets even more expensive with high-dimensional problems. This is because the correlation matrix size is fully dependent on the sample size, N_s. The sample size needed to train a model is suggested to be ten times the problem dimension. This shows the issue with high problem dimension and how it indirectly affects model training time. This is known as *curse of dimensionality* [107]. For instance, it takes between 2000s to 3500s to train a 80-dimensional problem with kriging [89]. This could be limiting, especially for real-time applications. Real-time applications require that model training and prediction happen within the shortest time possible.

From literature, many methods have been proposed to tackle this challenge. Bouhlel *et al.* [29] proposed the use of partial least squares (PLS) regression [94] technique to reduce the number of hyperparameters to the number of components. PLS [254] is used to attempt to capture the relationship between the inputs and responses. Then, a modification is made to the covariance function such that the PLS coefficients are used. Readers are referred to the papers on KPLS by Bouhlel *et al.* [29, 31] for more information on the method. The method was reported to be less effective with multimodal functions.

Hence, a two-stage method, KPLS+K method was developed. In the new method, the first stage consists of the normal KPLS method. For the second stage, the hyperparameters are estimated in the original space with a number of dimensions equal to the original problem dimension. The estimated value is then used as a starting point for an internal optimization process. This new method takes more time, with no promise for better results. While this method was proposed to tackle the poor performance of KPLS models with multimodal problems, the KPLS+K model could still perform poorly, regardless of the significant increase in training time. This situation might occur when the estimated hyperparameter value from the first stage is in the region of the local optimum. This limitation is further supported by the evidence provided in the paper [29]. For the benchmark carried out on a multimodal problem, the KPLS still performed better in terms of

model accuracy and training time than the newly proposed KPLS+K method especially with lower training samples. Common to the KPLS and KPLS+K methods is the PLS regression technique. While these methods aim to estimate the relationship between inputs and the output by obtaining the PLS coefficients, the relationship between these coefficients and kriging hyperparameters is not clearly established. As such, this remains one of the reasons for the loss in model accuracy when compared with the conventional kriging model. Also, multimodality is not clearly addressed.

Zhao *et al.* [279] attempted to estimate kriging hyperparameters by using maximal information coefficient (MIC). MIC [88] is obtained by estimating the mutual information (MI) between each input variable and the response. After obtaining the MIC, a weighted parameter is obtained by maximizing the MLE. Hence, the number of hyperparameter needed is reduced to one. The method greatly reduces training time as evident in the benchmarks. The authors derived the kriging with MIC (denoted as KMIC) by claiming that (1) global sensitivity measure of a variable could be used to estimate for kriging hyperparameters and (2) MIC is a good measure of the global sensitivity of a variable on the response. Hence, MIC can be used to estimate kriging hyperparameters. The authors provided proof of the first claim by showing the correlation between the optimized hyperparameter θ_{opt} and the Sobol index using the 20-D ellipsoid function. Then, the authors proceeded to provide proof for the second claim by showing the correlation between the MIC estimate and the optimized hyperparameter θ_{opt} using an entirely different function: the g07 function. While the first claim might be true, the second claim does not have a solid proof. This might be the reason for the significant loss incurred by the KMIC model when compared with the conventional kriging model. This is evident in some of the benchmark cases provided in the paper by Zhao *et al.* [279].

Another method proposed in recent times is the kriging with distance correlation (KDIC) [78]. The distance correlation employed measures the distance between the joint characteristic function and the product of the marginal characteristic functions. The assumptions behind the adoption of this method is similar to the KMIC. The authors claimed that the DIC [141] can be used to estimate kriging hyperparameter because it could be a good measure to check the importance of input variables. Similar to the MIC computation, the DIC coefficients are computed between each input and the response. Also, as seen in

the KMIC paper, the authors proved their claim by showing that the correlation between the DIC estimate and the optimized hyperparameter θ_{opt} using the g07 function [279]. The hyperparameter trend of the g07 function is not intuitive enough to be used as a proof of concept. Perhaps, the ellipsoid function is considered more ideal.

Our work is deeply motivated by the identified limitations. We aim at deriving an intuitive and efficient method for hyperparameter estimation for high-dimensional problems. We propose a new approach, which combines the joint mutual information estimate with kriging. The joint mutual information estimate will be computed using the joint mutual information maximization (JMIM) technique [24]. With this technique, we capture the relationship between the input variables and the response, as well as the relationship between the variables themselves. We claim that these two conditions are necessary to capture the actual trend of kriging hyperparameter. After obtaining the JMIM estimate, we use a weight-bias approach towards approximating the hyperparameter values. More information about the proposed method will be given in Section 5.2.

With the theory behind our method explained, we benchmark with other models. Then, we explore kernel selections in high-dimensional problems. We also compare the performance of conventional kriging model with the proposed method using the four common kernels used in the engineering community. We aim to address the multimodality problem by comparing the performance of different kernels on the benchmark problems sets.

The rest of this chapter is structured as follows. The existing state-of-the-art techniques are presented in section 5.1.1. Section 5.2 gives an overview of the proposed approaches based on our insights from the preceding section. The overview include both the formulations and algorithm. We then introduce the different test cases to be used for the study in Section 5.3.2 and present the results and discussions in Section 5.3. In Section 5.4, we highlight the conclusions from this chapter and briefly discuss some imminent potential future directions.

5.1.1 Existing State-of-the-art Methods

Kriging with partial least squares: The kriging with partial least squares (KPLS) method was derived by Bouhlel *et al.* to preserve accuracy while drastically decreasing the computing cost. KPLS technique significantly shortens the time required to build the model by reducing the number of hyperparameters that must be estimated from N_d to N_h, where $N_h \ll N_d$. The hyperparameter estimation procedure makes use of the well-known PLS regression approach. A smaller subspace produced by the principal components, or latent variables, is used by the PLS regression to optimize the variance between the input and output variables. According to Bouhlel *et al.*, the PLS approach can be used to estimate kriging hyperparameters because it provides information on the contribution of any variable to the outcome. The PLS information is used to add weights to the hyperparameters.

We examine the relationship between the PLS estimate and actual kriging hyperparameter by using a 20-D ellipsoid function. We draw 200 sample points with LHS and obtain the corresponding function values. After obtaining these data, we train our kriging model by maximizing the MLE to obtain the optimal kriging hyperparameter. We also obtain the PLS estimate for the same data. We plot both the PLS estimate and the optimal kriging hyperparameter for each of the variable indices. Figure 5.1 shows the comparison between the optimal kriging hyperparameter and the PLS estimate. From the plot, a monotonic decrease in the kriging hyperparameter is observed with the change in variable index. With the PLS estimate, the value varies between -0.43 and 0.6 with no explainable pattern. Perhaps, this explains the loss in model accuracy in some benchmark cases in literature [31, 279].

Kriging with maximal information coefficient (KMIC): This method was recently introduced by Zhao *et al.* [279] to reduce the computational cost involved in training kriging models especially in high dimensions without significant loss in model accuracy. The method uses MIC) in the approximation of kriging hyperparameters. The MIC used is an optimized version of the mutual information (MI) [202], which is based on information theory. While the value of MI can range from 0 to $+\infty$, the MIC value can only vary between 0 and 1. This is because the MIC estimate is obtained upon normalization of possible MI values between two variables. The MIC values for a

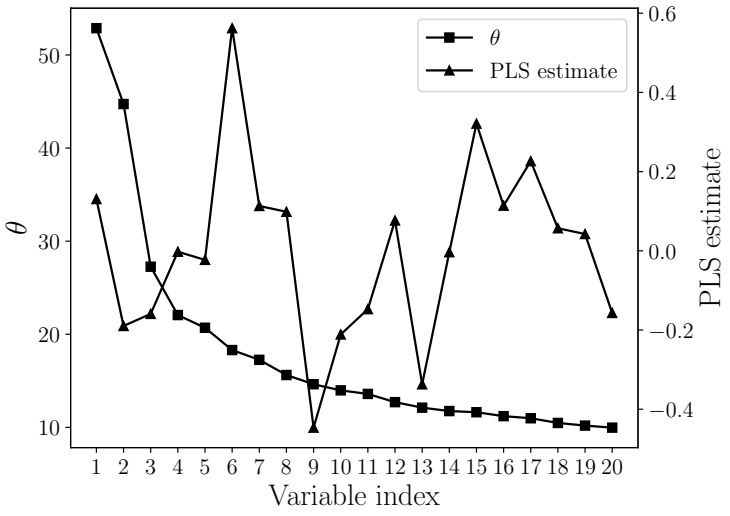


Figure 5.1: Plot showing the variation of the optimal kriging hyperparameter, θ and PLS estimate with variable dimension of the 20-D ellipsoid problem.

multidimensional problem is computed by estimating the MIC value between each variable $\mathbf{x_i}$ and the output $\mathbf{y}$. The authors claimed that the MIC method can be used to efficiently identify the influence of variables on the output and as such, be useful in the estimation of kriging hyperparameters.

Similar to the KPLS comparison as shown in Figure 5.1, we compute the MIC estimate for a 20-D ellipsoid problem by using 200 data points. We plot the values obtained against the variable indices and compared it with the optimal kriging hyperparameter. Figure 5.2 shows the comparison between the optimal kriging hyperparameter and the MIC estimate across the variable dimension. From Figure 5.2, the MIC estimates are observed to vary between 0.18 and 0.30 with no clear trend. Because of this behaviour, the kriging with maximal information coefficient will yield similar performance to isotropic kriging.

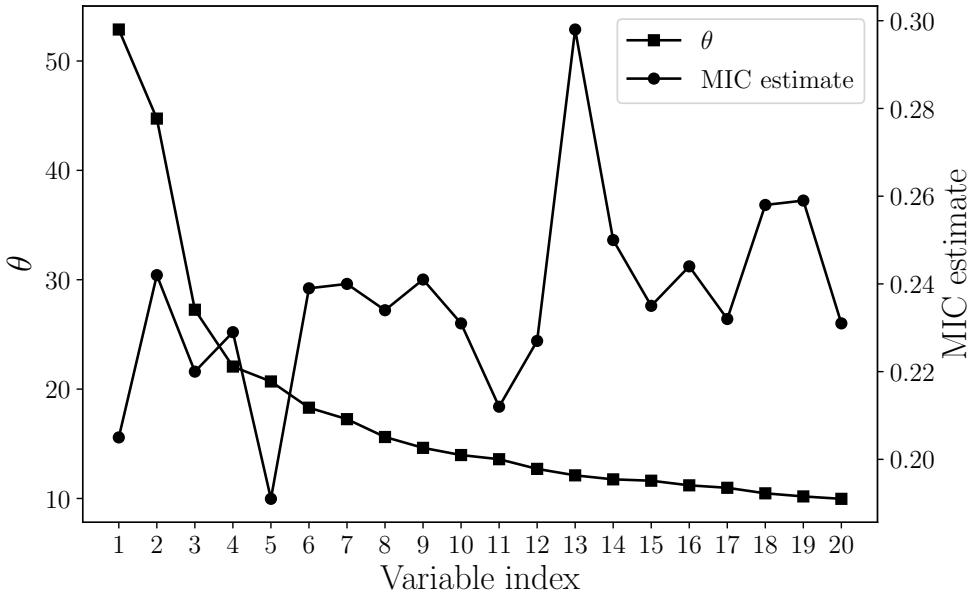


Figure 5.2: Plot showing the variation of the optimal kriging hyperparameter, θ and MIC estimate with variable dimension of the 20-D ellipsoid problem.

5.2 Kriging with Joint Mutual Information Mazimization

In the previous section, we gave a detailed background into kriging and hyperparameter estimation. We also discussed two state-of-the-art methods and highlighted their limitations. Based on the limitation, we explore the use of newly-introduced feature selection methods in the approximation of kriging hyperparameter in this section. We then propose a new method for efficient modeling of high-dimensional problems.

The KMIC method has been used to greatly reduce the computational time of kriging models. However, this comes with loss in model accuracy in many cases. This can be observed with some of the benchmarks in literature [279]. We trace this limitation to the theory behind the method. Using an intuitive algebraic function, the ellipsoid function, we estimate the MIC and compare its trend with kriging hyperparameters over the variable dimension. We use the ellipsoid function because the importance of the problem variables can be easily explained. For the ellipsoid function, the variable importance increases with the variable index. That is, x_2 is more important than x_1 and x_3 is of more importance than x_2. With our kriging covariance function notation, where $\kappa \propto h/\theta$, an increase in variable dominance will be reflected in a decrease in the value of θ. Hence, we expect θ to decrease progressively across the variable dimension. We observe that it is not the case with the MIC values. The MIC values fluctuate within a certain range that the behaviour can be classified as isotropic. Hence, the MIC values does not capture the true trend of the kriging hyperparameter, as shown in Figure 5.2. In the case of the KPLS method, two limitations can be identified. The partial least squares coefficient does not capture the trend in kriging hyperparameter. Also, the method is only limited to the Gaussian and exponential kernels [190]. This makes its adoption limiting for applications which might require other kernels.

With the limitations of the two common methods highlighted, it becomes imperative to use a method that is capable of capturing the inter-relationship between kriging hyperparameter values as well as the relationship of the input variables with the output. To achieve this, we propose to further investigate the effectiveness of using mutual information to understand the inherent relationship between variables.

5.2.1 Learning from mutual information

The amount of information between several variables can be obtained by estimating their joint entropy. Entropy is a measure of uncertainty of random variables [226]. For a continous random variable, X with probability density function, $p(x)$, entropy can be obtained using Equation 5.1 [106].

$$H(X) = -\int p(x) \log p(x) dx \tag{5.1}$$

When there is another variable, Y, the joint entropy between variables X and Y can be obtained using Equation 5.2.

$$H(X,Y) = \iint p(x,y) \log p(x,y) dx dy, \tag{5.2}$$

where p(x,y) is the joint probability density function of both variables.

The dependency between the two variables can be estimated by computing the shared information between the variables, as expressed by the mutual information (MI). The MI of variables, X and Y can be defined by Equation 5.3.

$$I(X;Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dx dy, \tag{5.3}$$

where $p(x)$ and $p(y)$ are the marginal probability density functions of the X and Y variables respectively. In terms of the entropy, MI can be expressed using Equation 5.4.

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{5.4}$$

With more than two variables, the MI between all the variables can also be obtained. The MI equation for high-dimensional cases with m variables is provided in Equation 5.5

$$I(X_1, X_2, ..., X_m) = \iint ... \int p(x_1, x_2, ...x_m) \log \frac{p(x_1, x_2, ...x_m)}{p(x_1)p(x_2)...p(x_m)} dx_1 dx_2 ... dx_m \tag{5.5}$$

where $p(x_1, x_2, ...x_m)$ is the joint pdf and $p(x_1)$, $p(x_2)$, ..., $p(x_m)$ are the marginal pdfs.

When it is also important to capture the dependency between the multiple variables and a defined target, Y, it is ideal to use a joint MI. The joint MI can be obtained using Equation 5.6.

$$I(X_1, X_2, ..., X_m; Y) = \iint ... \int p(x_1, x_2, ...x_m, y) \log \frac{p(x_1, x_2, ...x_m, y)}{p(x_1)p(x_2)...p(x_m)p(y)} dx_1 dx_2 ... dx_m dy \tag{5.6}$$

Unlike with the MI estimate between two variables, the JMI [268] captures the dependency between the variables and the target, Y, and also the internal correlation between the variables.

The minimum redundancy maximum relevance (mRMR) [198, 205] is another method that has been proposed for this purpose. Common limitation of both the JMI and mRMR methods are the overestimation of the significance of features [163]. As such, Bennasar *et al.* [24] proposed to use the "maximum of the minimum" approach [253, 74] with JMI method to overcome the issues of overestimating the significance of features. The newly proposed method was termed *joint mutual information maximization* (JMIM) [24].

With JMIM, the selection of features is done after a measure of feature importance is done. Features having negligible importance scores are then removed. Since the importance scores show the influence of the input variables on the target response, it might be worthwhile to use this in the estimation of kriging hyperparameters. In JMIM, a subset of features S with dimension K where $K \leqslant N$, and $S \subseteq F$. For a feature set $F = f_1, f_2, ..., f_N$ of a data set D of dimension N, the JMIM criterion is given in Equation 5.7 below:

$$f_{JMIM} = \arg \max_{f_i \in F-S} \left(\min_{f_s \in S} \left(I\left(f_i, f_s; C\right)\right)\right), \tag{5.7}$$

where,

$$I\left(f_i, f_s; C\right) = \left[-\sum_{c \in C} p(c) \log(p(c)) \right] - \left[\sum_{c \in C} \sum_{f_i \in F-S} \sum_{f_s \in S} \log \left(\frac{p(f_i f_s, c/f_s)}{p(f_i/f_s)p(c/f_s)} \right) \right]. \tag{5.8}$$

From the equations, f_i, f_s are the candidate and selected features respectively. C is the class label. Readers are referred to the by paper by Bennasar *et al.* [24] for more information on joint mutual information maximization. We use the MIFS [112] python package by Daniel Homola for the implementation of the JMIM method.

5.2.2 Proposed method

Since we can obtain the significance of each variable without overestimation with JMIM, we propose its use in the estimation of the kriging hyperparameter trend. We hope that by using information theory in our method, we can derive a new method that greatly reduces the computational cost without significant loss in model accuracy. It will also be desirable if this method can improve model performance when compared to the conventional ordinary kriging. Lastly, we aim for a method that can easily be extended to other kernels.

With 200 training points drawn with LHS, we obtain the responses by evaluating the ellipsoid function. We then compute the mutual information using JMIM. It takes about 0.15 s to compute this coefficients for the 20-dimensional ellipsoid problem. The comparison between the JMIM coefficients and the kriging hyperparameter is shown in Figure 5.3a. From the figure, it is evident that trend of the kriging hyperparameter is well captured by the JMIM estimate.

Hence, the next task is to approximate the kriging hyperparameter from the JMIM estimate. To do this, we propose using a weight-bias approach. The equation for this approach is shown in Equation 5.9.

$$\theta = \omega \boldsymbol{\lambda}_{\mathbf{JMIM}} + \beta \mathbf{I}, \tag{5.9}$$

where β is the bias term and ω is the weight term. ω is introduced to scale the JMIM estimate (λ_{JMIM}) towards the true kriging hyperparameter while the bias term is used for corrective purposes.

To obtain both values, we need to optimize the MLE equation. As such, we modify Equation 2.11 by introducing the weight and bias as the optimization variables. The modified MLE equation is given in Equation 5.10. After achieving this, we initialize the weight and bias, then solve the new optimization problem using the COBYLA optimization code. After optimization, we obtain values of 48.47 and 6.17 as the weight and bias, respectively. By combining the weight with the JMIM coefficient, followed by correction with bias term, we obtain the approximated hyperparameter values. We compare the approximated values with the true hyperparameter values in Figure 5.3b. From Figure 5.3b,

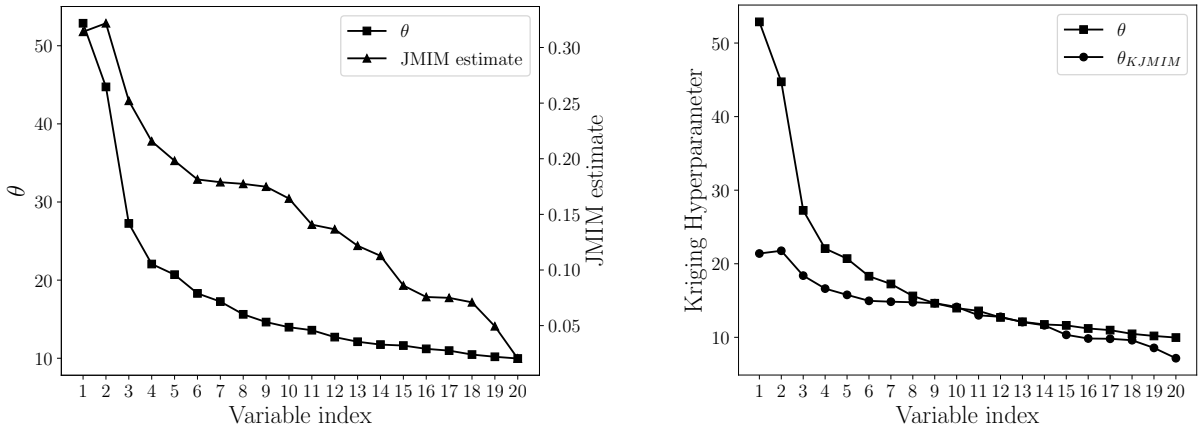
it is evident that the kriging hyperparameter is well approximated by the approximated hyperparameter. The proposed approach is summarized in Algorithm 6.

$$\max_{\omega,\beta} \quad -\frac{N_s}{2}\ln(2\pi) - \frac{N_s}{2}\ln\hat{\sigma}(\omega,\beta) - \frac{1}{2}\ln|R(\omega,\beta)| \tag{5.10}$$

Algorithm 6 Kriging with Joint Mutual Information Maximization (KJMIM).

Input: $\mathbf{x}$, $\mathbf{y}$, $\mathbf{X}$
Output: θ^f, ω^f, β^f, $\hat{\mathbf{y}}$

1: Evaluate $\boldsymbol{\lambda}_{\mathbf{JMIM}}$
2: Initialize ω, β
3: Assemble $\mathbf{R}(\boldsymbol{\omega}, \boldsymbol{\beta}) = \prod_{d=1}^{N_d} \kappa(\omega, \beta)$
4: Obtain ω^f, β^f by maximizing the likelihood estimate (Equation 5.10) using COBYLA algorithm.
5: Approximate $\theta^f = \omega^f \boldsymbol{\lambda}_{\mathbf{JMIM}} + \beta^f \mathbf{I}$
6: Assemble $\mathbf{R}(\theta) = \prod_{d=1}^{N_d} \kappa\left(\theta_d^f\right)$
7: Compute the kriging weights $\boldsymbol{\omega} = \mathbf{R}^{-1}(\mathbf{y} - \boldsymbol{\mu})$
8: Estimate $\hat{\mathbf{y}} = \boldsymbol{\mu} + r'(\mathbf{X}) \cdot \boldsymbol{\omega}$



(a) θ and JMIM estimate

(b) θ and θ_{KJMIM}

Figure 5.3: Plots showing comparisons between θ and MIC, JMIM estimate, θ_{KJMIM} for the 20-D ellipsoid problem.

5.3 Results and Discussion

In this section, we present the test cases and give a description of the different performance metrics that will be used for model comparison. Lastly, we present and discuss the benchmark results for the various test cases.

For the model comparison, we benchmark the performance of our proposed method (KJMIM) with four other methods. We select KMIC as one of the four methods because it is one of the best state-of-art methods used for efficiently modeling high-dimensional problems in recent times. We also use KPLS with three and four components. Our reason behind this stems from our observation in recent papers. For example, Zhao *et al.* [279] carried out their benchmark using KPLS method with components varying from one to three, that is, KPLS-1, KPLS-2, KPLS-3. It is observed that the performance of the models increased with the number of components. A corresponding increase in training time was also recorded. Similarly, Bouhlel *et al.* [29] used KPLS models with one to three components. Zuhal *et al.* [282] used KPLS with components ranging from one to four. The authors then recommended using the KPLS model with four principal components. Lastly, we use the ordinary kriging, as this remains the base model for any comparison in kriging.

We perform all experiments using a personal computer with the specifications as follows: Intel® Core™i7-8700 CPU at 3.20 GHz and with 16 GB of RAM, in which all experiments were performed in Python™.

5.3.1 Performance metrics

In comparing the performance of the proposed approach with other model, we employ four metrics. The likelihood estimate, commonly used in statistical communities to compare model performance is employed. This is computed using Equation 2.11. We use the NRMSE and R^2 metrics to compute the model accuracy. Essentially, R^2 denotes the correlation between the true model and the surrogate model [238]. The surrogate model is more accurate if R^2 is closer to one. Lastly, we capture the training time to measure the computational efficiency of each model. The time is measured in seconds.

5.3.2 Test cases

To evaluate the performance of our proposed approach and also assess the performance of different kernels, it is important to benchmark using algebraic functions. For this purpose, we choose four algebraic problems with their dimensions varying between 20 and 80. Description of the problems are provided in Table 5.1.

Since it is difficult to visualize high-dimensional problems, we propose presenting such problems based on their characteristics. It is equally complex to properly characterize high-dimensional problems. For this reason, we attempt to characterize our test cases based on the peaks present in their profile. To do this, we devise using different starting points in optimization of the functions. After the optimization, we compute the variance of the optimum points. We assume that by doing this, we can detect if a particular function has many peaks, that is, multimodal. We vary the problem dimension from two to 80. For each problem dimension, we then use LHS [47] to generate 30 starting points within the input range of the problems. We then use the Nelder-Mead algorithm [158, 28] for optimization of each sub-steps. We compute the variance of the minima obtained after the minimization of the functions in Table 5.1. We plot the variance against the problem dimension. This is shown in Figure 5.4.

From the Figure 5.4, we observe that the Rosenbrock, ellipsoid and Dixon-Price have a small variance in their minima points in low dimensions. This could be interpreted as being unimodal. However, as problem dimension increases, the variance values for the Dixon-Price and Rosenbrock functions increased greatly. This is noticeable when $N_d \geqslant 7$. This could mean the presence of many local optima within their function space and/or that the optimum points are farther apart. Shang and Qiu [225] proved that the behaviour of perceived unimodal N_d-dimensional functions like Rosenbrock tends to change with an increase in dimension. The authors showed that the Rosenbrock function starts showing signs of multimodality from four dimensions. The Griewank function, which is a well-known multimodal problem has its variance values almost constant, even with increase in problem dimension from two to 80. This suggests that the complexity of the function in lower dimensions is the same as that in higher dimensions. The ellipsoid function, on the other hand, has its variance below 10 for as much as the 30 minima obtained after the optimization process, suggesting that the function is largely unimodal. In summary,

Table 5.1: Algebraic test cases.

Name	N_d	N_s	Expression
Ellipsoid	20	200	$f(x) = \sum_{i=1}^{20} ix_i^2, x_i \in [-5, 5], i = 1, ..., 20$
Dixon-Price	30	300	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{30} i(2x_i^2 - x_{i-1})^2, x_i \in [-10, 10], i = 1, ..., 30$
Rosenbrock	40	400	$f(x) = \sum_{i=1}^{39} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right], x_i \in [-5, 10], i = 1, ..., 40$
Griewank	80	500	$f(x) = \sum_{i=1}^{80} \frac{x_i^2}{4000} - \prod_{i=1}^{80} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, x_i \in [-5, 5], i = 1, ..., 80$

three out of the four benchmarking functions are multimodal. Multimodality is a key characteristics of real-world problems.

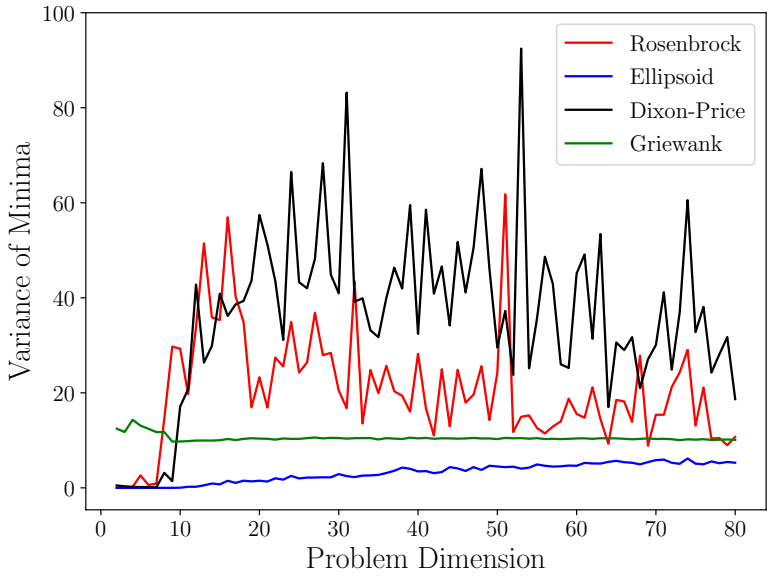


Figure 5.4: Plot showing variance of minima for varying dimensional problems.

5.3.3 Experimental study

In this study, we seek to understand the performance of our method. Our approach towards this is to compare the performance with the conventional ordinary kriging (OK).

We also compare the performance of the proposed method with the KPLS 3, KPLS 4 and KMIC models. We draw training points of 200, 300, 400 and 500 using LHS for the ellipsoid, Dixon-Price, rosenbrock and Griewank functions respectively. While we use sampling points with their sizes equal to $10N_d$ for the ellipsoid, Dixon-Price and Rosenbrock functions, we use just 500 points points for the 80-D Griewank function. We limit the training points to 500 based on the intuition we gained from Figure 5.4. After drawing the samples, we evaluate the responses of the algebraic functions. Using both the inputs and outputs, we train the different models with the Gaussian covariance function. We capture the time it takes for each model to complete the training process. For each of the problem sets, we draw 5 000 test points also using LHS. We then use the trained model to predict the response at the test points. We obtain the true responses of the test points and compared them to the prediction. This is done for validation of the models. We repeat this process 20 times for each of the benchmark case. The results obtained from the benchmarking process are presented in Figures 5.5-5.8, and they will be adequately discussed next.

Likelihood comparison

The likelihood estimate is the value obtained from maximizing the MLE function. Essentially, it is the point in the parameter space that maximizes the likelihood function. The logic behind the likelihood estimate is intuitive and flexible. Consequently, it has become a dominant means of statistical inference. Hence, it is of utmost importance and should be discussed. From our earlier study into model likelihood in this thesis, we observe that the likelihood ranges differently for different kernels. The Gaussian kernel is observed to have the highest achievable likelihood estimate and also the lowest achievable likelihood estimate for a given hyperparameter space. The exponential kernel tends to have the lower ranging achievable likelihood estimate. We also observe that this does not necessarily limit the modeling accuracy of the exponential kernel. Hence, it is important that the likelihood should only be used as a statistical inference between models that use the same kernel. Since we use the Gaussian kernel as the covariance function for all the models, our comparison is valid. The ordinary kriging model (OK) will be used as the base model. For the ellipsoid function, the KPLS-3 records the least likelihood, with its

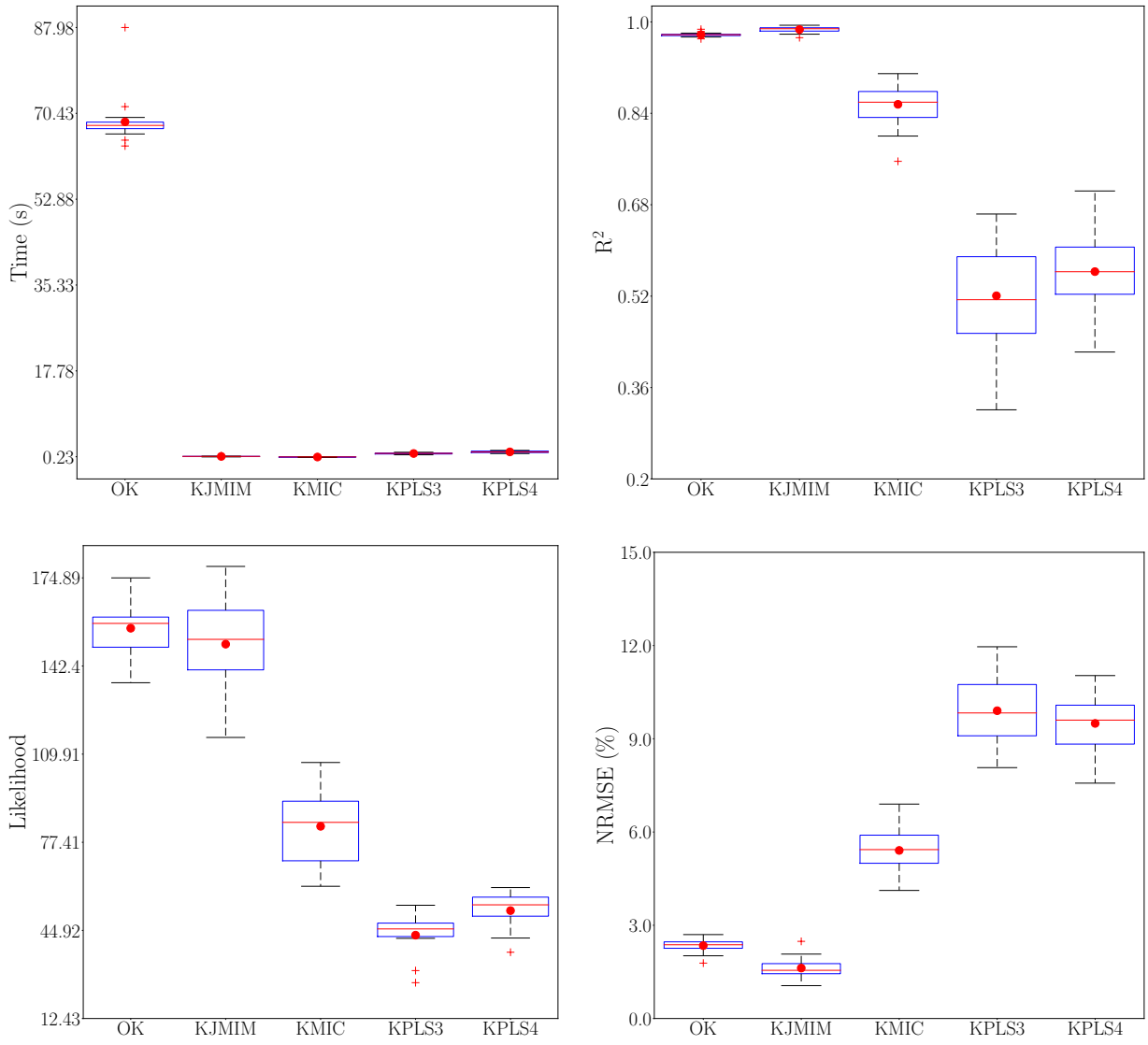


Figure 5.5: Box-plots showing the model benchmarking results for the 20-D ellipsoid function.

mean likelihood estimate slightly less than one-third of the mean likelihood estimate of the ordinary kriging model. A slight improvement in the likelihood estimate is observed in KPLS-4 model. However, it is still approximately one-third of the likelihood obtainable by the ordinary kriging model. A mean likelihood estimate of approximately 78 is recorded for the KMIC model. This value is only about half the mean likelihood estimate of the ordinary kriging model. This shows a significant loss in model likelihood with the KMIC model. For the proposed KJMIM model, the mean likelihood estimate is close to that of the ordinary kriging model. This shows that for the ellipsoid problem, the KJMIM model best approximated the OK model.

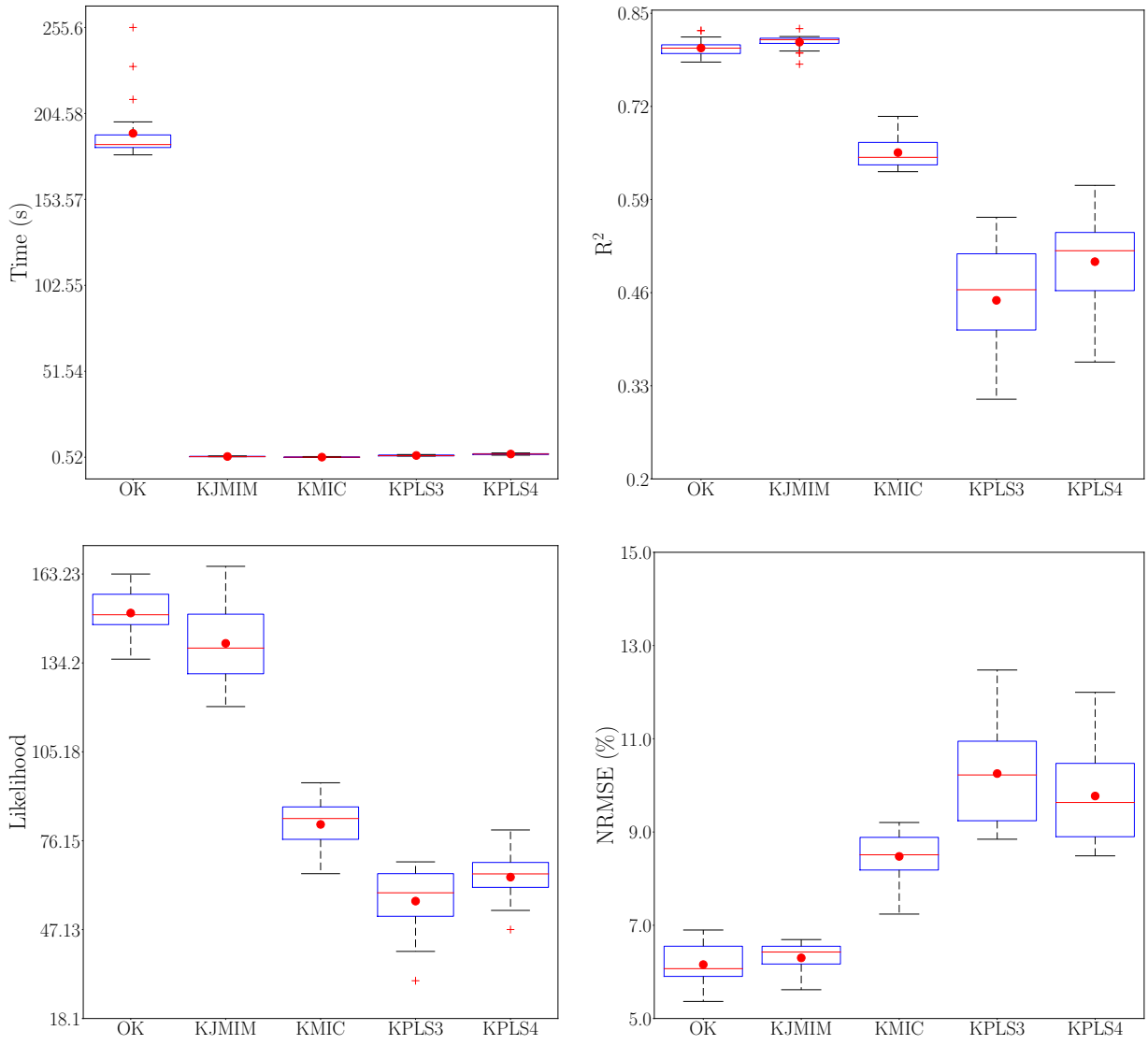


Figure 5.6: Box-plots showing the model benchmarking results for the 30-D Dixon-price function.

For the Dixon-Price problem benchmark results as shown in Figure 5.6, we observe similar patterns as seen in the ellipsoid benchmark. The KPLS-3 model has the least mean likelihood estimate, followed closely by the KPLS-4 model. The KMIC model here also has a mean likelihood estimate which is about half of the mean likelihood estimate of the OK model. The KJMIM model also closely approximates the OK model in this benchmark case.

As in the case of the 40-D Rosenbrock problem with the results presented in Figure 5.7, the likelihood estimate of the KPLS-3 and KPLS-4 models are 150.72 and 162.32 respectively. This shows a marginal improvement with the addition of one more PLS compo-

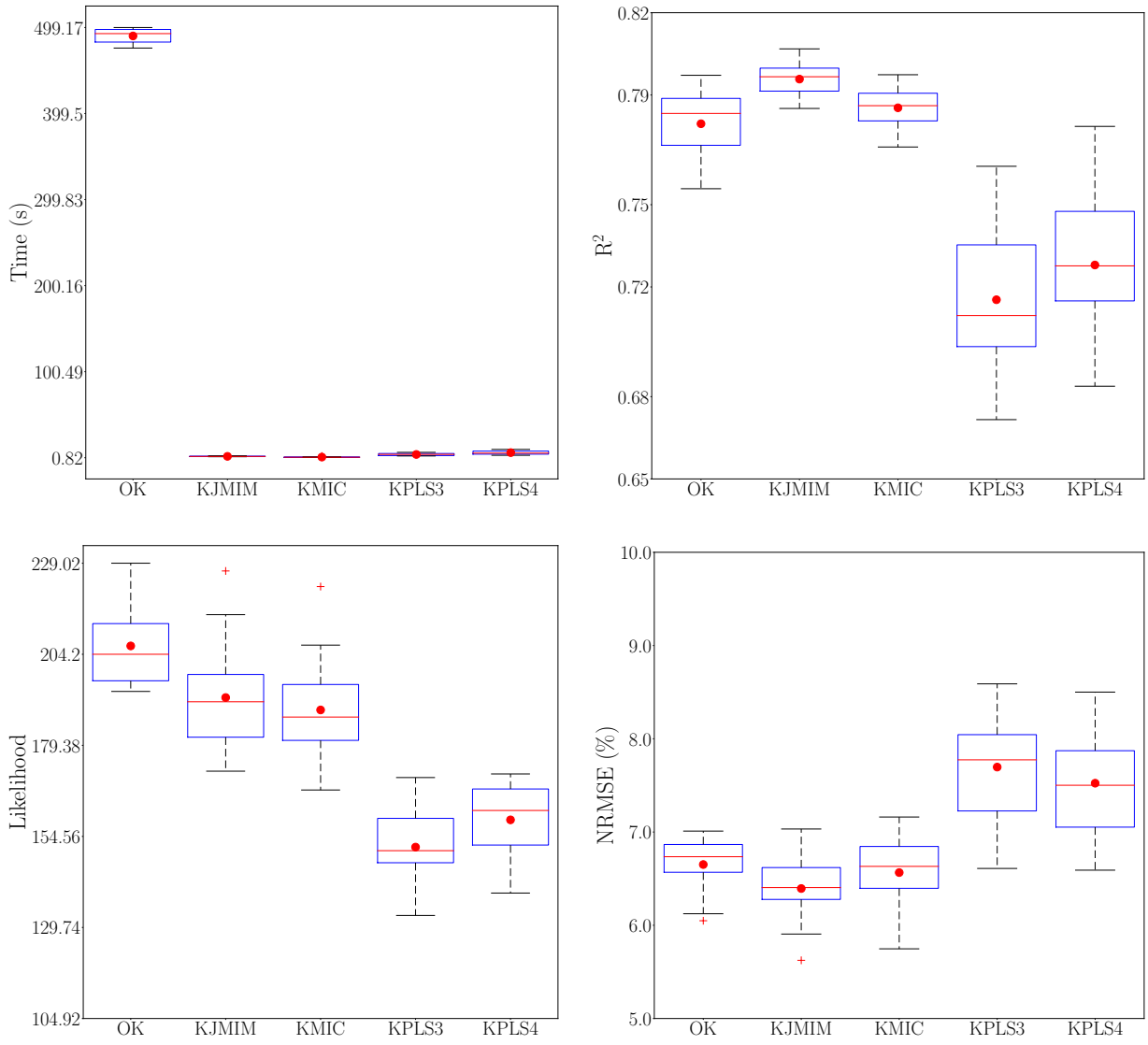


Figure 5.7: Box-plots showing the model benchmarking results for the 40-D Rosenbrock function.

nent to the KPLS-3 model. Both KPLS models still perform worse when compared to the base model, OK, for the benchmark case. In this case, the performance of the KJMIM and KMIC models are similar, with the KJMIM performing slightly better than the KMIC model. The KJMIM and KMIC models have a mean likelihood estimate of 195.2 and 189.3 respectively. Based on these observation, the KJMIM model still best approximates the OK model.

The KJMIM and KMIC model have a much similar performance in terms of the likelihood for the 80-D Griewank benchmark as presented in Figure 5.8. Only a marginal improvement can be observed in the KJMIM over the KMIC model. In this case, they both

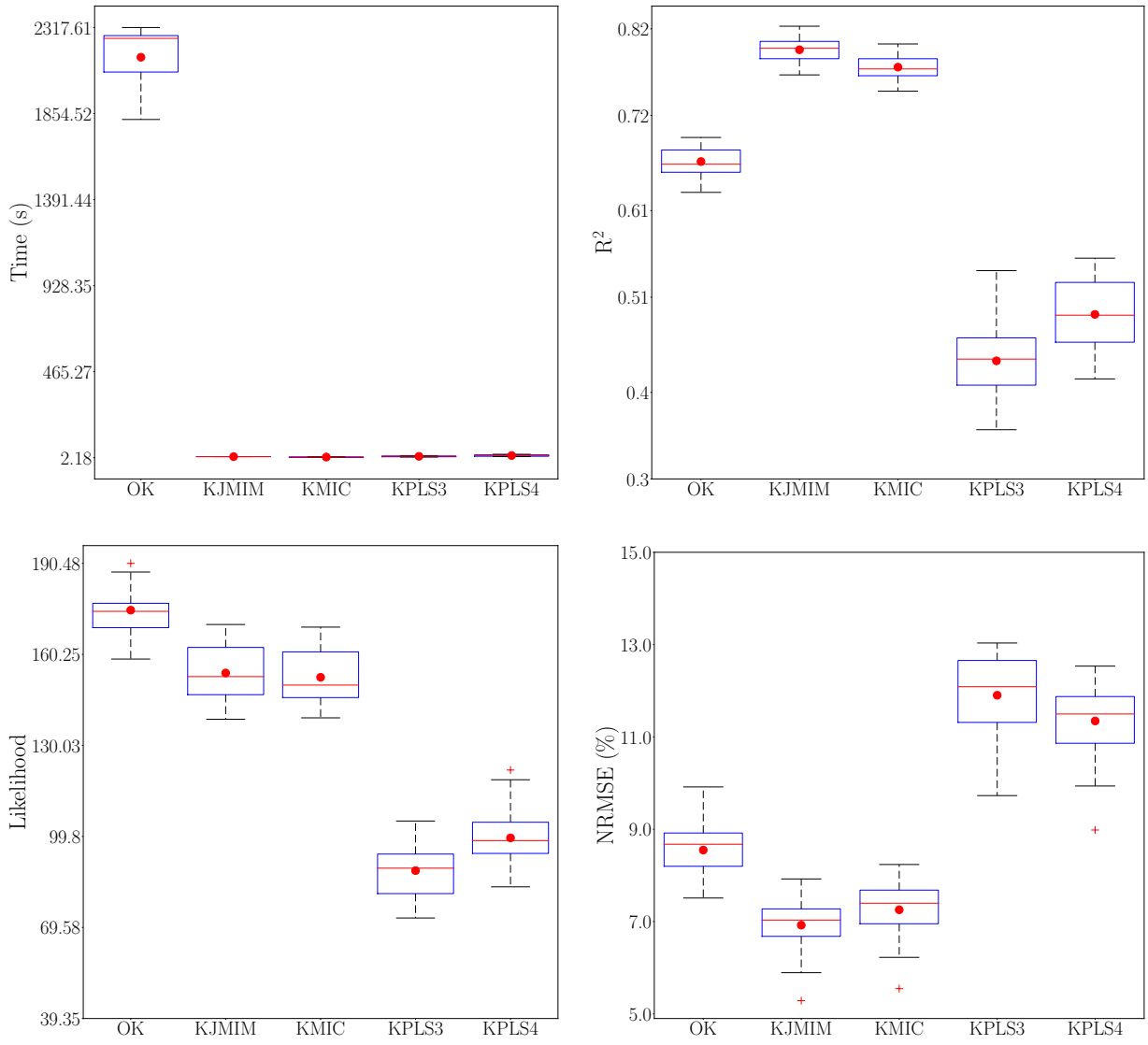


Figure 5.8: Box-plots showing the model benchmarking results for the 80-D Griewank function.

closely approximate the OK model when compared to both KPLS-based models. The summary of the likelihood estimate benchmark is shown in Table 5.2.

Accuracy comparison

While the likelihood estimates for models can be used for statistical inference, it is desirable to measure the predictive performance of the models. This is usually done by comparing the prediction made by using a trained model with the actual responses. A strong assessment of a model performance can be evaluated this way. To that end, we use the R^2 and the NRMSE to assess the performance of our model's accuracy. We use a test

Table 5.2: Table showing the summary of the likelihood estimates for different models.

Model	Statistic	Ellipsoid (20-D)	Dixon-Price (30-D)	Rosenbrock (40-D)	Griewank (80-D)
OK	Mean	156.33	150.49	206.48	174.94
	Std	10.18	7.52	11.02	9.13
KJMIM	Mean	150.45	140.60	192.41	154.07
	Std	16.01	12.98	13.13	9.53
KMIC	Mean	83.29	81.47	189.03	152.64
	Std	14.43	8.44	13.33	9.41
KPLS-3	Mean	43.16	56.43	151.64	88.46
	Std	8.29	11.72	10.47	8.64
KPLS-4	Mean	52.21	64.26	159.07	99.32
	Std	7.25	8.49	10.41	9.75

Table 5.3: Table showing the summary of the computational training time for different models.

Model	Statistic	Ellipsoid (20-D)	Dixon-Price (30-D)	Rosenbrock (40-D)	Griewank (80-D)
OK	Mean	68.69	192.79	489.39	2157.67
	Std	4.39	17.72	8.62	172.49
KJMIM	Mean	0.29	0.97	2.41	7.93
	Std	6.57e-2	0.18	0.37	0.34
KMIC	Mean	0.17	0.59	1.56	5.24
	Std	4.81e-2	0.16	0.52	1.94
KPLS-3	Mean	0.89	1.59	4.61	9.45
	Std	0.17	0.31	1.30	2.30
KPLS-4	Mean	1.22	2.43	6.72	13.85
	Std	0.22	0.37	2.17	3.88

set with size of 5 000 for all the benchmark cases. We choose the two metrics because they are intuitive and strong measure of comparison can be derived using them. We also use the OK model as the baseline model here.

First, we compare the model accuracy of the different models using the R^2 values. The results are also presented in Figures 5.5-5.8. In all of the benchmark cases, a great loss in the R^2 value is observed in both KPLS-based models. The performance appears worse in the ellipsoid benchmark case where the KPLS-3 model had an average R^2 value

Table 5.4: Table showing the summary of the model accuracy metrics for different models.

Model	Statistic	R^2	NRMSE (%)	R^2	NRMSE (%)
		Ellipsoid (20-D)		Dixon-Price (30-D)	
OK	Mean	0.98	2.35	0.80	6.16
	Std	3.49e-3	0.21	1.12e-2	0.40
KJMIM	Mean	0.99	1.62	0.81	6.30
	Std	5.05e-3	0.31	1.04e-2	0.32
KMIC	Mean	0.86	5.41	0.65	8.48
	Std	3.66e-2	0.73	2.19e-2	0.50
KPLS-3	Mean	0.52	9.90	0.46	10.25
	Std	9.38e-2	1.13	7.05e-2	1.05
KPLS-4	Mean	0.56	9.49	0.52	9.77
	Std	7.88e-2	0.90	6.73e-2	0.91
		Rosenbrock (40-D)		Griewank (80-D)	
OK	Mean	0.78	6.65	0.67	8.55
	Std	1.31e-2	0.27	1.81e-2	0.70
KJMIM	Mean	0.80	6.39	0.80	6.92
	Std	6.34e-3	0.35	1.52e-2	0.67
KMIC	Mean	0.78	6.57	0.77	7.25
	Std	7.14e-3	0.38	1.59e-2	0.69
KPLS-3	Mean	0.72	7.70	0.44	11.90
	Std	2.61e-2	0.56	4.03e-2	0.90
KPLS-4	Mean	0.73	7.52	0.49	11.35
	Std	2.38e-2	0.53	3.82e-2	0.85

of approximately 0.45 and the baseline OK model has a high R^2 value of approximately 0.98. This is shown in Figure 5.5. A poor performance is also recorded with the Griewank case, as shown in Figure 5.8. The KPLS-3 had a R^2 value of approximately 0.45, while the baseline model records a value of 0.69.

Comparing the performance of the KMIC models with the baseline model, a significant loss in the R^2 value is seen in the ellipsoid and Dixon-Price benchmark cases. For this case, R^2 losses of about 15% and 20% are observed with the ellipsoid and Dixon-Price cases, respectively. For the 40-D Rosenbrock benchmark case, with the results presented in Figure 5.7, the performance of the KMIC model is similar to the baseline model. A close median R^2 value is noticed with the two models.

An improvement of about 13.75% in the R^2 value is observed for the KMIC model in

the 80-D Griewank case. As for the R^2 performance of the KJMIM model, an improvement is observed in all the benchmark cases. The model has similar average R^2 value with the baseline model in the ellipsoid and Dixon-Price cases. For the KJMIM model, a significant improvement in model R^2 value is observed in 40-D Rosenbrock and 80-D Griewank cases. In the Rosenbrock case, for instance, the KJMIM has the lower boundary of its R^2 values slightly above both mean and median R^2 value of the baseline model. This shows the superiority of the model, when the R^2 value is used for comparison. With the multimodal Griewank function, the model developed with the proposed method, KJMIM records a great improvement in the model R^2 value. The NRMSE is negatively correlated to the R^2 value for a model, that is, a model with high error will have a low R^2 value, and vice versa. Using the NRMSE metric, the KPLS-based models record high model errors. The KPLS-3 model records high average errors of approximately 12.8% and 10.8% for the Griewank and ellipsoid benchmark cases, respectively. High comparative error is recorded with the KMIC models in the ellipsoid and Dixon-Price benchmark cases. For instance, in the ellipsoid case, errors of approximately 5.8% and 2.8% are recorded for the KMIC and the baseline OK models, respectively.

Based on both metrics, it is clear that the KPLS models are not suitable in modeling these high dimensional problems due to the poor performance. The summary of the model accuracy benchmark is shown in Table 5.4.

Training time comparison

The training time is the time it takes a model to obtain the hyperparameter values that optimizes the maximum likelihood function. The training time could be used as a means of measuring the computational efficiencies of models. The training time is a function of many factors. The optimizer settings, for example, could significantly affect the time it takes a model to train. For fairness sake, we use the same setting for all the models, so that we can have a valid comparison. We capture the training time in seconds. We observe that the training time of all the models increased with more training points as seen in the benchmark cases. If we use the ordinary kriging (OK) model as the baseline, we observe that compared to other model, the OK model requires more time to train. For instance, the KMIC is approximately 411 times faster to train than the OK model for the

Griewank case. The KJMIM model is observed to be 272 times faster than the OK model for the same case. With the excessive training time of the OK model established, it will be worthwhile to also compare other models. The KMIC model tends to require the least training time, closely followed by the KJMIM model. Apart from the OK model, the KPLS-4 model tends to require more training time. The reason behind the increase in training time is clear. The KMIC models requires the optimization of a single parameter, while our proposed model requires that two parameters are optimized simultaneously. The KPLS-3 and KPLS-4 require the optimization of three and four reduced hyperparameters. In essence, the more the number of parameters to train, the higher the computational cost. Comparing the computational efficiency between our proposed model and the KMIC model, we discover that the KMIC model is only 1.54 times faster than the KJMIM model. For instance, the KJMIM and KMIC models trained for 2.41 seconds and 1.56 seconds respectively in the Rosenbrock benchmark. The summary of the model computational time benchmark is shown in Table 5.3.

5.3.4 Influence of kernel selection in high-dimensional problems

We have been able to assess the superiority of our method over both state-of-the-art techniques on all the benchmark cases. We also show that our method can greatly reduce the computational time required by models, without a significant loss in model accuracy. Now, we compare the performance of different kernels on the different cases using both the baseline model and our proposed method. To do this, we use the four covariance functions to build for both the OK and KJMIM models. We also use 200, 300, 400 and 500 training samples here for the ellipsoid, Dixon-Price, Rosenbrock and the Griewank functions respectively. We draw these point using LHS. For all the cases, we draw 5 000 validation points randomly, also using LHS. We assess model performance using the NRMSE and the R^2 value. We present the results of the experiments in boxplots as shown in Figures 5.9-5.12. The red and blue boxes in the boxplots represent the OK and KJMIM models respectively.

A summary of the kernel comparison benchmark for the ellipsoid function is presented in Figure 5.9. We observe that kernels have similar performance for the OK models. The Matérn 3/2 performed best with a very low NRMSE mean value of 0.48%. The Matérn

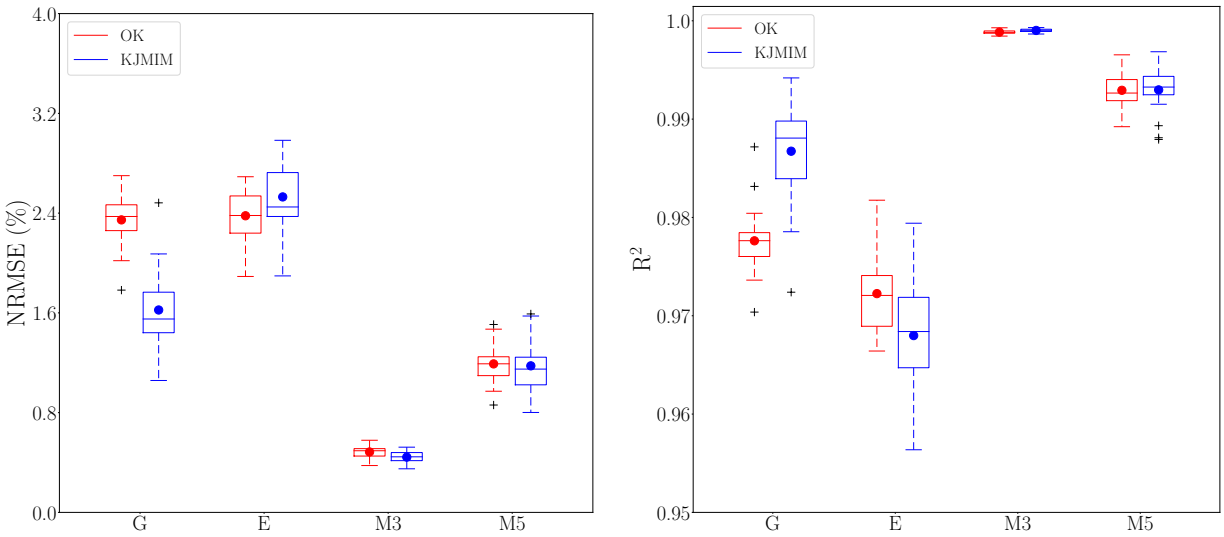


Figure 5.9: Box-plots showing the comparison between kernels for the 20-D ellipsoid function.

5/2 also records a good model performance with a mean R^2 score slightly above 0.99 and a mean NRMSE of 1.19%. The exponential and Gaussian kernels have similar model performance with mean NRMSE of 2.38% and 2.35% respectively. For the KJMIM model benchmark, the Matérn 3/2 and Matérn 5/2 models record great model accuracy with their mean R^2 score almost close to 1.0. They both record low NRMSE values of 0.44% and 1.18% respectively. The Gaussian kernel model with KJMIM also has similar R^2 score of 0.99 and very low NRMSE of 1.62%. Comparing the performance of the kernels for both OK and KJMIM models, the exponential, Matérn 3/2 and Matérn 3/2 have similar model performance. However, with the Gaussian kernel used for modeling the problem, a significant improvement of approximately 31.1% is recorded with the KJMIM model over the baseline OK model. Overall, the performance of the kernels is similar in both OK and KJMIM models. We explain this by referring the readers to the variance plot in Figure 5.4. From the plot, the ellipsoid function is clearly unimodal. This shows that the ellipsoid problem is of little complexity and the probability of encountering difficulties during modeling is small. As such, the selection of kernel for relatively simple problems such as the 20-D ellipsoid function might not be of key importance, especially when enough training sample points are used.

The comparison for the different kernels with Dixon-Price function using both OK and KJMIM models is presented in Figure 5.10. With OK, a similar performance is observed with the Gaussian and Matérn 5/2 kernels. Models developed with both kernels have a

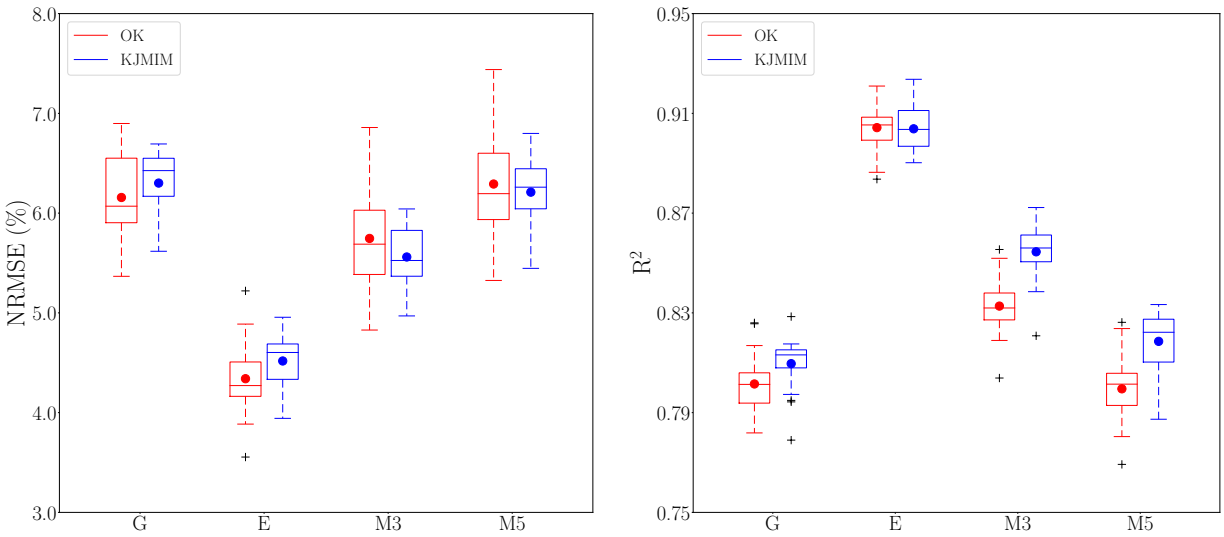


Figure 5.10: Box-plots showing the comparison between kernels for the 30-D Dixon-Price function.

R^2 score of 0.80. A slight improvement of about 3.75% R^2 score from the Gaussian kernel is recorded with the Matérn 3/2 kernel. While the model performance significantly improved with the exponential kernel when compared to other kernels. To be specific, a 12.5% improvement in the mean R^2 score is observed. With the KJMIM models, a similar performance is also observed with both the Gaussian and Matérn 5/2 kernels. Likewise, a slight improvement is observed with the Matérn 3/2 kernel and a significant improvement in model performance with the exponential kernel. Comparing the performance of the kernels for both OK and KJMIM models, a similar performance is observed. This shows that the model structure of the OK is well-approximated by the KJMIM models. The relatively lower performance with the Gaussian and Matérn 5/2 kernels for both OK and KJMIM models can be explained by the presence of several modes in the Dixon-Price profile as depicted in Figure 5.4.

The comparison for the different kernels with Rosenbrock function using both OK and KJMIM models is presented in Figure 5.11. With the OK models, a similar performance is observed with the Gaussian and Matérn 5/2 kernels. Models developed with both kernels have a R^2 score of 0.78. A significant improvement in model performance is observed in both exponential and Matérn 3/2 kernels. Using the Gaussian kernel as a baseline, 16.7% and 19.2% improvement in the mean R^2 scores is observed in the exponential and Matérn 3/2 kernels respectively. With the KJMIM models, a similar performance is also observed

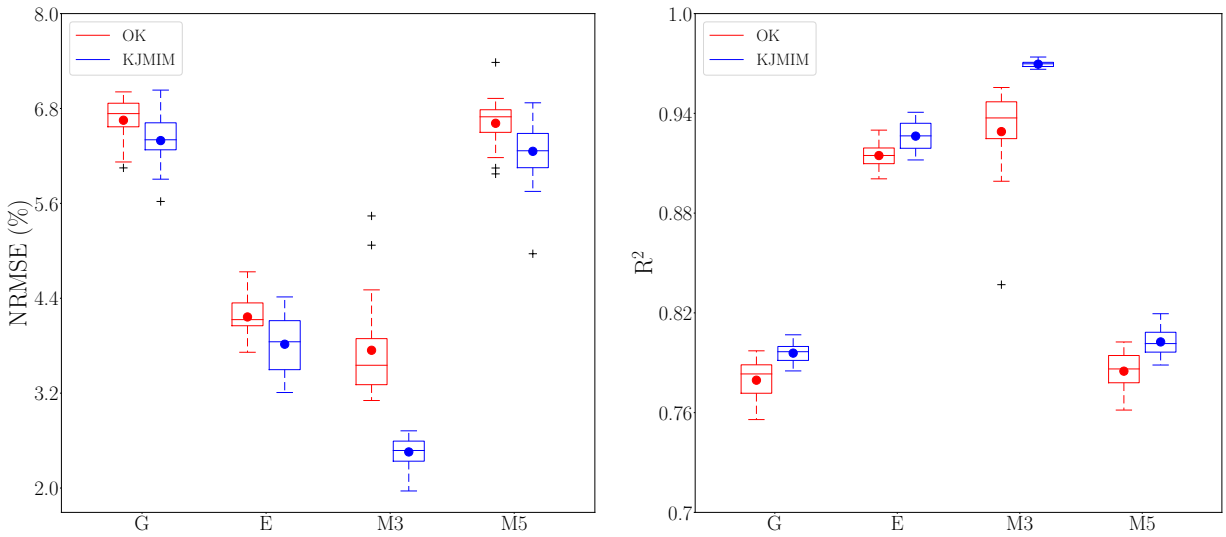


Figure 5.11: Box-plots showing the comparison between kernels for the 40-D Rosenbrock function.

with both the Gaussian and Matérn 5/2 kernels. Likewise, significant improvement in model performance is observed with the Matérn 3/2 and exponential kernels, using the Gaussian kernel as the baseline model. In this case, the Matérn 3/2 model performance records a R^2 score of 0.97, thereby moving closer to the 1.0 mean R^2 score. Comparing the performance of the kernels for both OK and KJMIM models, a similar performance is observed, with the KJMIM performing better than OK in most cases. This shows that the model structure of OK is well-approximated by the KJMIM models. Also, it shows the possibility of improving model performance when reduced models such as KJMIM are employed. The relatively lower performance with the Gaussian and Matérn 5/2 kernels for both OK and KJMIM models can be explained by the presence of several modes in the Rosenbrock function profile as depicted in Figure 5.4.

The comparison for the different kernels with Griewank function using both OK and KJMIM models is presented in Figure 5.12. With the OK models, a similar performance is observed with the Gaussian and Matérn 5/2 kernels. The Gaussian and Matérn 5/2 kernels have close R^2 scores of 0.67 and 0.68 respectively. The model performance is further improved with the exponential and Matérn 3/2 kernels, using the Gaussian kernel as a baseline model. The exponential and Matérn 3/2 kernels have mean R^2 scores of 0.74 and 0.71 respectively. With the KJMIM models, similar performance is observed with all the kernels. The kernels have their models with R^2 score ranging between 0.79 and 0.81.

Comparing the performance of the kernels for both OK and KJMIM models, with all the kernels used in the benchmark, there is a significant improvement in the model performance across board. Relatively low performing models like the Gaussian and Matérn 5/2 kernels with OK witnessed more improvement in model performance. The KJMIM Gaussian and Matérn 5/2 kernels record 19.4% and 16.2% improvement in the mean R^2 score respectively. These results show that the model structure of OK is well-approximated by the KJMIM models. Also, it shows the possibility of improving model performance when reduced models such as KJMIM are used for modeling. The relatively lower performance with the Gaussian and Matérn 5/2 kernels for OK can be explained by the presence of several modes in the Griewank function profile as depicted in Figure 5.4. The similar performance between kernels when the KJMIM model is used show that using reduced models such as KJMIM model can further simplify complexity associated with high dimensionality, especially when the multimodality is not too much as in the case of the Dixon-Price and Rosenbrock functions.

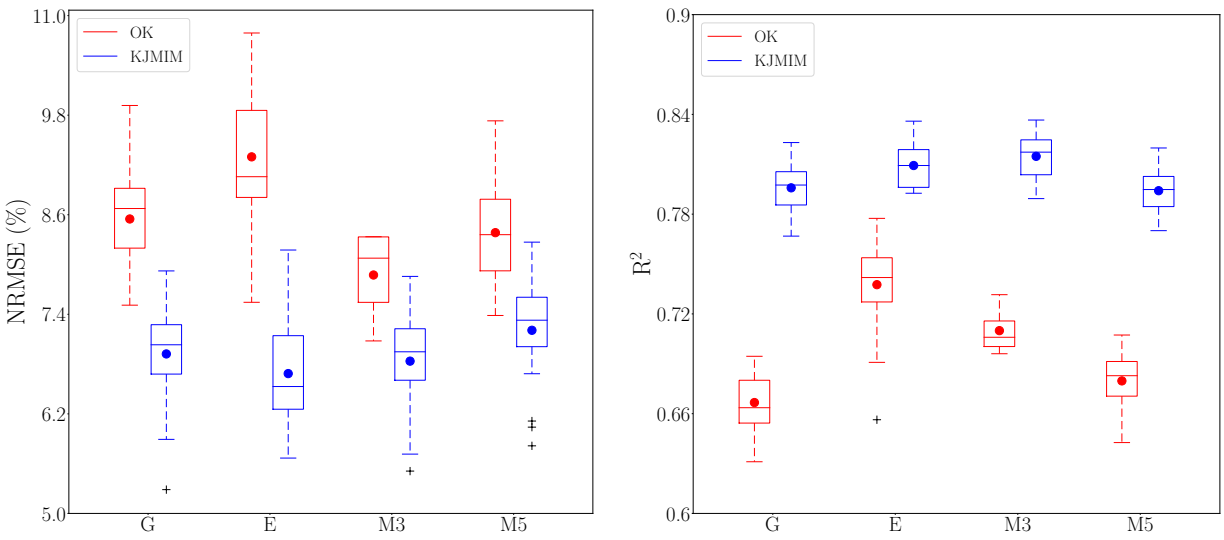


Figure 5.12: Box-plots showing the comparison between kernels for the 80-D Griewank function.

In almost all the cases and kernels, an improvement in the model accuracy is observed with the KJMIM models. The only exception is seen with the exponential kernel model in the 20-D ellipsoid benchmark case. Perhaps, reducing the number of hyperparameters to be optimized could help build more accurate models. The magnitude of improvement varies in the different models. The most significant improvement in model accuracy is seen in the Griewank function benchmark case. An interesting observation is the reduced

performance of the Gaussian and Matérn 5/2 kernel models compared to the exponential and Matérn 3/2 kernel models. This is evident in the Dixon-Price, Rosenbrock and Griewank benchmark cases. The multimodal nature of these functions might be a possible reason behind this observation. Now, we compare the performance of the exponential and Matérn 3/2 kernels in the KJMIM models. We observe that the Matérn 3/2-based KJMIM models tend to show more noticeable improvement in model accuracy. The summary of the performance for different kernels with both OK and KJMIM models is given in Table 5.5.

5.4 Summary

In this chapter, we reviewed the performance of existing model reduction methods for high-dimensional problems. We observed that with the existing state-of-the-art methods, there are significant losses in model performance. We traced these losses to the significant changes in model structures imposed by the methods during hyperparameter approximations. As such, we proposed an intuitive method to greatly reduce the computational cost of kriging models without significant loss in model accuracy. We used a weight-bias approach to obtain the approximated kriging hyperparameter by optimizing the MLE equation, after the estimation of the hyperparameter trend with JMIM. It is important to mention that the computational efficiency of the proposed method is also dependent on the number of parameters to be optimized during model training. With the proposed method, the number of parameters to be trained has been reduced from N_d to two.

Thereafter, we compared the performance of the novel method to the baseline ordinary kriging model as well as two state-of-the-art methods. With our proposed method, an improvement in performance is evident in both the computational efficiency and model accuracy. The proposed method is approximately 272 times faster than the conventional OK while retaining its model structure. The method also outperformed the two state-of-the-art methods in terms of model accuracy.

Afterwards, we studied the influence of kernels on model performance in high-dimensional problems in both OK and reduced kriging models. Our extensive study showed that for relatively simple functions such as the ellipsoid problem, the choice of the kernel used

Table 5.5: Table showing the summary of the model accuracy metrics for different models and kernels.

Model	Kernel	Statistic	R^2	NRMSE (%)	R^2	NRMSE (%)
			Ellipsoid (20-D)		Dixon-Price (30-D)	
OK	Gaussian	Mean	0.98	2.35	0.80	6.16
		Std	3.49e-3	0.21	1.12e-2	0.40
	Exponential	Mean	0.97	2.38	0.90	4.34
		Std	4.11e-3	0.22	9.47e-3	0.35
	Matérn 3/2	Mean	0.99	0.48	0.83	5.75
		Std	1.97e-4	4.80e-2	1.09e-2	0.47
	Matérn 5/2	Mean	0.99	1.19	0.80	6.29
		Std	1.80e-3	0.15	1.27e-2	0.54
KJMIM	Gaussian	Mean	0.99	1.62	0.81	6.30
		Std	5.05e-3	0.31	1.04e-2	0.32
	Exponential	Mean	0.97	2.53	0.90	4.52
		Std	5.96e-3	0.27	9.09e-3	0.27
	Matérn 3/2	Mean	0.99	0.44	0.85	5.56
		Std	1.73e-4	4.66e-2	1.10e-2	0.31
	Matérn 5/2	Mean	0.99	1.18	0.82	6.21
		Std	2.25e-3	0.19	1.23e-2	0.35
			Rosenbrock (40-D)		Griewank (80-D)	
OK	Gaussian	Mean	0.78	6.65	0.67	8.55
		Std	1.31e-2	0.27	1.82e-2	0.70
	Exponential	Mean	0.91	4.16	0.74	9.30
		Std	7.15e-3	0.23	2.68e-2	0.86
	Matérn 3/2	Mean	0.93	3.74	0.71	7.87
		Std	2.70e-2	0.64	1.27e-2	0.49
	Matérn 5/2	Mean	0.78	6.61	0.68	8.38
		Std	1.25e-2	0.32	1.88e-2	0.73
KJMIM	Gaussian	Mean	0.80	6.39	0.80	6.92
		Std	6.34e-3	0.35	1.52e-2	0.67
	Exponential	Mean	0.93	3.82	0.81	6.69
		Std	8.10e-3	0.34	1.30e-2	0.63
	Matérn 3/2	Mean	0.97	2.46	0.81	6.84
		Std	1.81e-3	0.19	1.30e-2	0.60
	Matérn 5/2	Mean	0.80	6.26	0.79	7.21
		Std	9.39e-3	0.44	1.38e-2	0.63

in modeling might not be of utmost importance. However, with multimodal functions such as the Griewank function, the choice of kernel became of great importance, especially when OK is used. However, the necessity of choosing a suitable kernel is lessened when the reduced models such as KJMIM was used. The case was different with highly multimodal high-dimensional functions such as Dixon-Price and Rosenbrock functions. It became imperative to carefully select kernels even when reduced models are used for modeling. Lastly, we discovered that multimodality which is usually common with high-dimensional problems only had negative effects on certain covariance functions. Within the framework of the general Matérn covariance function, multimodality might become a problem when $\nu > 3/2$. Hence, the use of Matérn $3/2$ and exponential kernels are advised in modeling high-dimensional problems.

CHAPTER 6

ROBUST STOPPING CRITERIA FOR ACTIVE LEARNING STRATEGIES

In this chapter, we present a detailed summary on active learning strategies. We also explain the fundamental difference between active learning and Bayesian optimization. Then, we discuss the importance of stopping criteria in active learning strategies. After this, we propose a novel method to define robust stopping criteria for active learning strategies. We present the motivation behind the development of the method and show benchmarking results. Lastly, we discuss the results from an extensive study of the effect of kernel selections on the learned functions.

6.1 Active learning strategies

In surrogate-aided design, it is imperative to develop surrogates that are capable of effectively replacing the computationally-expensive process. Hence, surrogates are adequately validated to check if they meet certain criteria, usually a defined model accuracy. In the case of the criteria not being met, more points are usually added. If these points are chosen at random, it might not help the surrogate move closer to the defined goal. Hence, the new points are usually added where they are deemed important. The process of adding more informative points to existing sample data is known as *active learning* [223, 245]. Active learning strategies are also commonly referred to as *adaptive sampling*. These strategies rely on information extracted from the current metamodel [79]. With these methods, new samples are selected based on information extracted during previous iterations. The new samples are usually found in locations of high interest.

Active learning strategies are proposed to circumvent the limitation of one-shot sampling. When one-shot sampling is used, the chances of under/oversampling is high and thus, resulting in poor system approximation [87]. With active learning strategies, the tendencies of the occurrence of under/oversampling can be greatly reduced. Because of the

immense advantages of active learning strategies, it has received increasing attention in recent years. The steps involved in active learning strategies are explained in Section 6.1.1.

6.1.1 Steps involved in active learning

With several development in recent times, it is important to give the fundamental algorithm behind the active learning framework. The steps involved in a typical active learning method is given below;

1. Select the initial samples from the defined domain. Usually, space-filling methods such as LHS is used to draw some samples.
2. Evaluate the objective function to obtain the functional values of the drawn sample points in 1.
3. Build the surrogate model with the existing sample data. Gaussian process is usually the *go-to* choice of surrogate model. This is because of the ease of obtaining a measure of uncertainty with the modeling technique. SVM can also be used for this purpose [135].
4. Choose new points by searching the design space. The point that maximizes the defined learning function is selected. The number of sample points added in each iteration can either be one or multiple. When one new sample point is added in each iteration, this is known as single-selection strategy. Batch-selection strategies are those where multiple points are added in each iteration. Batch-selection strategies are preferred because they tend to converge faster, with a defined criteria.
5. Add the new points from 4 to the sample data pool.
6. If the stopping criterion is met, the active learning process is stopped. Else, Steps 3 - 5 are repeated.

The active learning process is shown in Figure 6.1 and summarized in Algorithm 7. The success of the active learning strategy is dependent on the learning function, which determines the quality of new points.

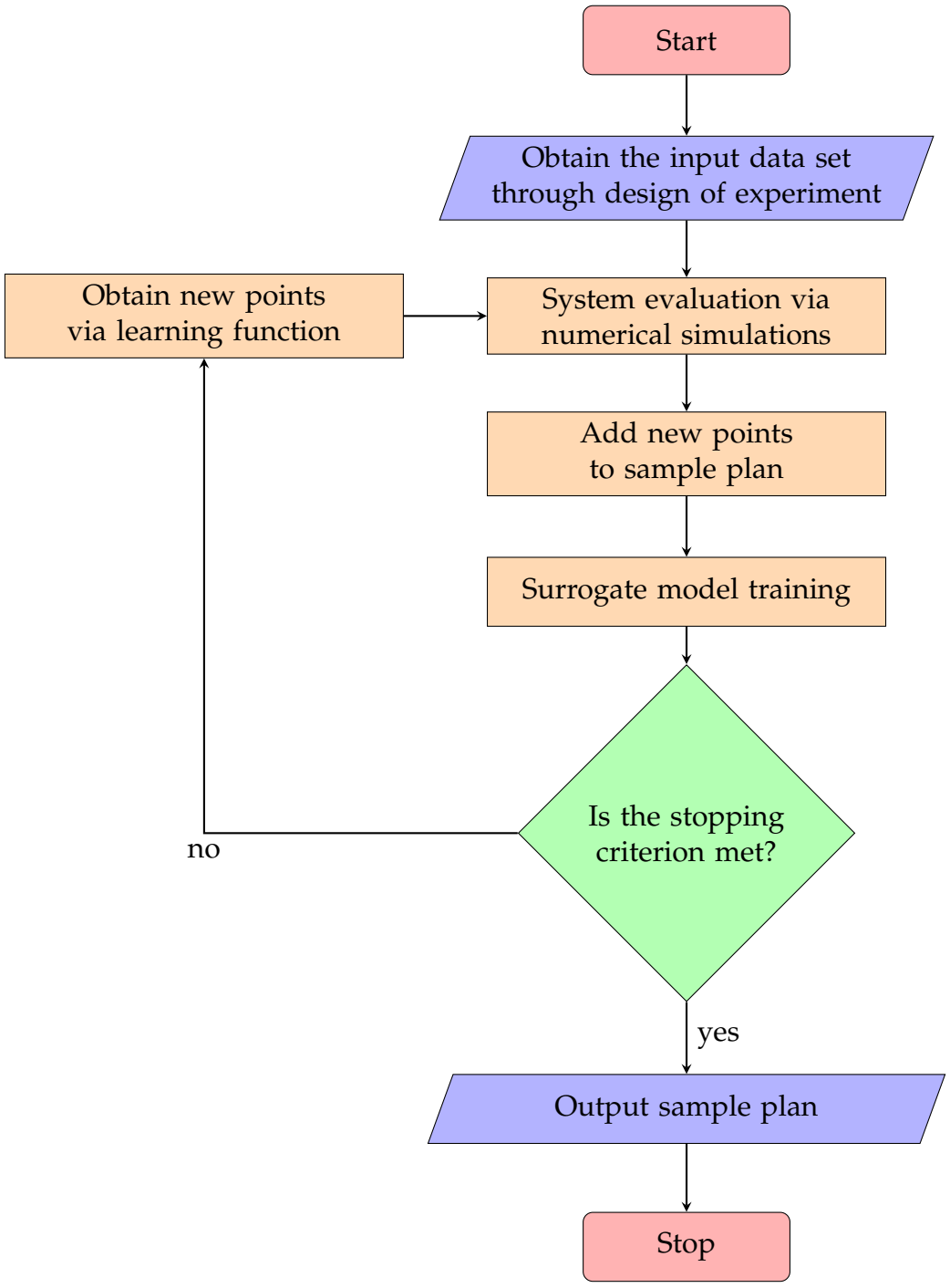


Figure 6.1: Flowchart showing the steps of active learning strategies.

Algorithm 7 Typical active learning strategy

Input: Function, $f(.)$, Initial sample size, n_i
Output: $\mathbf{x}^{opt}$, $\mathbf{y}^{opt}$
Draw initial samples, $\mathbf{x_1}$
Obtain the functional value of the input data, $y_1 = f(\mathbf{x}_1)$
Initialize iteration, $t = 1$
while stopping criterion is not met **do**
    Calculate GP posterior $q_t(f|S_t)$
    Obtain next proposed point by maximizing the learning function.
    Obtain functional value of new point: $\mathbf{y}_{t+1} = f(\mathbf{x}_{t+1})$
    Update dataset: $S_{t+1} \leftarrow S_t \cup \{(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})\}$
    $t \leftarrow t + 1$
end while

6.1.2 Learning functions

To select a new, information-rich point to be added to a sample plan, knowing the importance of the points within the defined continuous input domain is beneficial. Hence, a function which can be used to estimate this importance is desired. Such a function is known as the *learning function*. In a continuous domain, the process of obtaining new informative points is formulated as an optimization problem [79]. When the optimization problem is solved, new points are obtained in every iteration. With single-selection schemes, only one sample point $\mathbf{x}^{n+1}$ is added to the sample plan after every iteration. The new addition is obtained by maximizing the learning function or the refinement criterion (RC) using Equation 6.1 as follows;

$$\mathbf{x}^{n+1} = \underset{\mathbf{x}^* \in \mathcal{X}}{\operatorname{argmax}}\, RC(\mathbf{x}^*). \tag{6.1}$$

For active learning methodologies, two options are typically presented: local exploitation and global exploration. Exploration seeks to thoroughly scan the entire input domain in order to understand the whole space. As a result, the pure exploration technique ignores the results of earlier evaluations while performing active learning [79]. In contrast, exploitation is based on the previously extracted observations.

While more advanced procedures integrate both perspectives, some sample methods suggested in the literature are based only on one attribute. Exploration techniques are

Table 6.1: Common active learning strategies that combines both exploitation and exploration [79].

Exploitation \ Exploration	Distance-based	Variance-based
Cross-validation	SSA	MEPE
Geometry-based	-	EIGF

either distance- or variance-based. Exploitation techniques are usually based on cross validation or geometry. Liu *et al.* [148] and Deschrijver *et al.* [58] advised that an effective active learning approach should contain three parts; exploration, exploitation and the trade-off between the two strategies. The hybrid techniques that seek to balance between exploration and exploitation techniques tend to have a fair mix of the different methods. Table 6.1 shows the combined techniques used by common active learning strategies.

Expected Prediction Error (EPE)

Kriging is one of the methods used in developing active learning strategies and it is known that the posterior distribution of the response at any point follows a normal distribution. The normal distribution can be characterized with a mean and variance. The mean and variance of a point $\mathbf{x}$ can be obtained after training a kriging model. The expected prediction error, EPE can be obtained using;

$$\mathrm{EPE}(\mathbf{x}) = \left(f(\mathbf{x}) - \hat{f}(\mathbf{x})\right)^2 + \hat{s}(\mathbf{x}). \tag{6.2}$$

With the use of Equation 6.2, it becomes possible to collect more useful points than the standard variance-based learning functions. This is because EPE takes into account the additional bias term. Although flexible and potent, active learning that takes into account local exploitation only works well in reality with acceptable local exploitation and a sensible trade-off between local exploitation and global exploration. It is discovered that in order to choose new points based on the EPE criterion, the bias and variance at point $\mathbf{x}$ are necessary. In the case where the function $f(\mathbf{x})$ is not known, k-fold cross-validation (CV) can be used to estimate the bias term.

Maximizing Expected Prediction Error (MEPE)

With the EPE, there is need to balance the exploitation and exploration to obtain

maximum results. To this end, Liu *et al.* [147] introduced the maximizing expected prediction error (MEPE) method. This approach introduces a balance factor α, to adjust the EPE criterion. The formulation is given as as;

$$\mathrm{EPE}(\mathbf{x})^{\alpha} = \alpha \left(\mathrm{f}(\mathbf{x}) - \hat{\mathrm{f}}(\mathbf{x})\right)^2 + (1-\alpha)\hat{\mathrm{s}}(\mathbf{x}) \tag{6.3}$$

The MEPE strategy utilizes cross-validation exploitation and variance-based exploration. Leave-one-out cross validation (LOOCV) error at each sample point is used in the estimation of the bias term. The balance factor then adaptively balance the exploitative and exploratory contributions [79]. For more information about the derivation of the α and the use of LOOCV error in the estimation of the bias component, readers are referred to the article on MEPE by Liu *et al.* [147].

Smart Sampling Algorithm (SSA)

Garud *et al.* [87] introduced the use of a distance-based technique to estimate the exploration term. This method is known as smart sampling algorithm (SSA). Just like the MEPE strategy, this approach also uses cross validation in the estimation of the exploitation part. In this strategy, exploration is performed by maximizing the crowding distance metric (CDM) [278]. A point $\mathbf{x}$ with a large value of CDM($\mathbf{x}$) would be located relatively far away from the sample points already incorporated in the dataset. In this strategy, several proposed sample points are evaluated using the CDM equation. The resulting values are sorted in descending order. The sample point with the highest CDM value is then selected as the candidate. Afterwards, the candidate is checked to ensure that it has a defined minimum distance to all existing sample points. A non-clustering parameter, ϵ, is used to achieve this goal. The candidate point is accepted as a new sample if the requirement is met. Otherwise, the process is repeated again until a candidate fulfills the non-clustering requirement. With the SSA strategy, it is easier to implement a batch selection of new points.

Expected Improvement for Global Fit (EIGF)

With the aim of obtaining an accurate estimation over the entire parametric domain, Lam [136] proposed a variant of the expected improvement (EI) function. This method is known as *expected improvement for global fit*. With this strategy, it becomes

possible to obtain a good global model fit. The method uses the geometric and variance features for the exploitation and exploration components, respectively. The expected improvement for global fit (EIGF) approach chooses informative points that have a large expected improvement over the nearest observed points. The refinement criterion is given as;

$$RC_{EIGF}(\mathbf{x}) = \left(\hat{f}(\mathbf{x}) - f(\mathbf{x}^*)\right)^2 + \hat{s}(\mathbf{x}), \tag{6.4}$$

where $f(\mathbf{x}^*)$ is the observed value at the closest neighbor to the point of interest $\mathbf{x}$. The first term grows as the discrepancy between the exact answer at the closest sampling point and the surrogate estimation f rises. In subdomains where the surrogate model has the highest inherent uncertainty, the second term which provides the exploratory sampling feature becomes substantial.

6.1.3 Difference between Bayesian optimization and active learning

In Section 6.1, active learning strategy was introduced. Sections 6.1.1 and 6.1.2 give more explanation into the steps and learning functions of active learning, respectively. Here, Bayesian optimization will be discussed. Then, the key differences between Bayesian optimization and active learning strategies will be discussed.

Bayesian optimization (BO) [124] is a powerful tool for finding the optimum values of objective functions that are expensive to evaluate. It proceeds by maintaining a probabilistic belief about f and designing an inexpensive function called *acquisition function* to determine the next location to evaluate. It can be used when there is no closed-form equation for the objective function, f, but there are observations of the function at sampling values that can be obtained, even if they are noisy. It is especially helpful when these evaluations are expensive, when derivatives are not available, or when the problem at hand is non-convex. When it comes to minimizing the number of necessary function evaluations, BO techniques are among the most effective methods. Much of the effectiveness stems from the ability of the tool to incorporate prior belief about the problem to guide the sampling process. Also the handling of trade-off between exploration and exploitation of the search space makes BO more valuable. It uses the well-known "Bayes' theorem," which is

why it is termed *Bayesian* [34].

With the explanation on BO, it becomes clearer that it is closely related to active learning. The objective of both tools makes it easier to differentiate between them. In active learning, the objective is to select the most informative observations for updating an existing model for future predictions, subject to a defined budget constraint. The objective of BO, on the other hand, is to select and gather the most informative observations for finding the global optimum of an unknown objective function. Lastly, active learning strategies use learning functions while acquisition functions are used in BO to select the next points.

6.1.4 Acquisition function

Acquisition functions refer to functions that can be used to locate the optimum point during Bayesian optimization. In recent times, they have equally been used in many active learning strategies. When the value of the acquisition function is at its possible highest, the maximum point can be selected. A high acquisition functional value could be as a result of large prediction mean or the variance. In some cases, both mean and variance of prediction contribute to the high value achieved with the acquisition function [34]. Some of the commonly used acquisition functions are [85];

Probability of improvement

The probability of improvement (PI) is one of the earliest acquisition function. It is a measure of expected utility. It evaluates a function *f* at the point which is most likely promising. In the context of BO, the point which has the highest PI is selected. PI can be calculated using the error function [75]. The function is expressed as;

$$a_{PI}(x) = \frac{1}{2}\left[1 + erf\left(\frac{f_{min} - \hat{f}(\mathbf{x})}{\hat{s}\sqrt{2}}\right)\right]. \tag{6.5}$$

Expected improvement

With PI as the acquisition function, there is tendency that BO gets stuck in a local optimal. This is because PI focuses on allocating rewards to points which improve upon the current functional minimum without putting size of the improvement into

consideration, which limits the application of PI. The *expected improvement* (EI) is an alternative acquisition function which attempts to find the global optimum by taking the size of improvement of the points into consideration. Jones *et al.* [125] derives the closed-form expression for the estimator of EI. The EI can be computed using the expression;

$$a_{EI}(x) = \left(f_{min} - \hat{f}(\mathbf{x})\right) \Phi\left(\frac{f_{min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\phi\left(\frac{f_{min} - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right), \tag{6.6}$$

where Φ and ϕ represent the cumulative distribution function (CDF) and the probability distribution function (PDF) of the standard normal distribution. The EI equation as shown in Equation 6.6 has two components; the first term represents the *exploitation* of the EI and the second term represents the *exploration* component. The exploitation component can be increased by reducing the mean function $\hat{f}(\mathbf{x})$ while the exploration term can be increased by increasing the variance $\hat{s}(\mathbf{x})$.

Entropy search

The entropy search is an acquisition function which attempts to minimize the uncertainty in the *location* of the optimal value. It evaluates points and selects the points that minimize the entropy of the induced distribution. The nature of the distribution usually does not have a closed-form expression. This makes it necessary to make several approximations when entropy search is used within BO framework.

Upper confidence bound

The upper confidence bound (UCB) is an acquisition function which balances the tradeoff between the exploration and exploitation by introducing a control parameter, β. Equation 6.7 shows the expression of the UCB acquisition function.

$$a_{UCB}(\mathbf{x}; \beta) = \hat{f}(\mathbf{x}) + \beta\sigma(\mathbf{x}). \tag{6.7}$$

As shown in Equation 6.7, the UCB acquisition function has explicit exploitation $(\hat{f}(\mathbf{x}))$ and exploration $(\sigma(\mathbf{x}))$ terms. The BO will favour solutions with high predictive value when the β is small. In the case of large β, the solution will tend to reward the exploration of more regions with the search space.

6.1.5 Importance of stopping criterion and the issues

When active learning techniques are used to improve the quality of the sample data, it is important to know when to stop adding new points to the sample data. When the sample data contain less than required points, there is a tendency that the surrogate model trained with such data performs below the expected performance. When more than enough sample points are used, several issues could occur. This might force several points to be very close to one another in the sample space. This could lead to ill-conditioning of the correlation matrix of the surrogate model. Apart from this, there is the increase in computational cost of a model with more training samples. Hence, defining a *stopping criterion* for active learning strategies is important. The various types of stopping criteria can be grouped into four categories [79]. They are;

Time constraints

In many industrial applications, time to deliver solutions to problem is usually considered right from planning stages. Because of this, time deadlines can be put in place to stop addition of more sample points during active learning. While this constraint is realistic, it does not guarantee that the current sample data are sufficient and will yield the required performance when used for the development of a surrogate model.

Computational constraints

The process of finding the new points with active learning requires some level of computational resources, even though it could be minimal sometimes. Populating the sample data with more points increases the computational requirements, as more points might take more time to train. Because of this, computational constraints could be used as the criterion for stopping an active learning process. These constraints are usually enforced in the form of a maximum number of iterations or final sample sizes. When a maximum number of iterations is defined as the stopping criterion, once it is achieved, the active learning process is stopped and the resulting sample data are used for surrogate model development. In the cases where a final sample size is defined, the process is equally stopped when the size of the sample data reaches the defined final size. Just like the case of defining time constraints,

placing computational constraints does not guarantee good performance of the resulting sample data.

Accuracy target

Typically, the objective of using active learning is to draw informative points that can best represent a defined function. To achieve this, an accuracy target is sometimes put in place. This is usually achieved by using metrics such as the NRMSE or R^2 score. When the defined model accuracy is met, the active learning process is stopped. The major issues with this stopping criterion is that it requires extra data which will be used for model validation in each iteration. There is also the issue of computational cost involved in making model predictions on a set of test data for each iteration. Lastly, this criterion is defined based on case basis and hence, might not be useful in robust applications.

Relative correction between two successive iterations

Towards defining a better stopping criterion, the influence of adding a new point might be evaluated and used in deciding when to stop the active learning process. If there is no significant improvement after adding a new sample point, increasing the sample size would only increase the computational complexity without any real benefits. Different measures of the correction can be considered for this criterion. The variation of the cross-validation error between two successive iterations can be used in defining a stopping criterion [60]. Allen [12] proposed that the leave-one-out estimate of the mean square error should be used as the metric to monitor convergence. This would involve computing the LOOCV error before and after the addition of a new point. In each of the cross validation procedures, a kriging model is built on a subset of the sample data and trained. This means that to compute the relative cross validation error, $2N_s + 1$ kriging models will be trained. The number of models to be trained grows with each iteration. This process is costly and might lead to a prohibitive learning time. Towards reducing the learning time, Dubourg *et al.* [60] recommended setting a minimal number of samples ($N_s \geqslant 30$) before computing the cross validation errors in each iteration. The authors believed that this would help against ineffective and wasteful evaluation of a costly procedure. However, the assumptions of the authors could be misleading as sometimes, a function

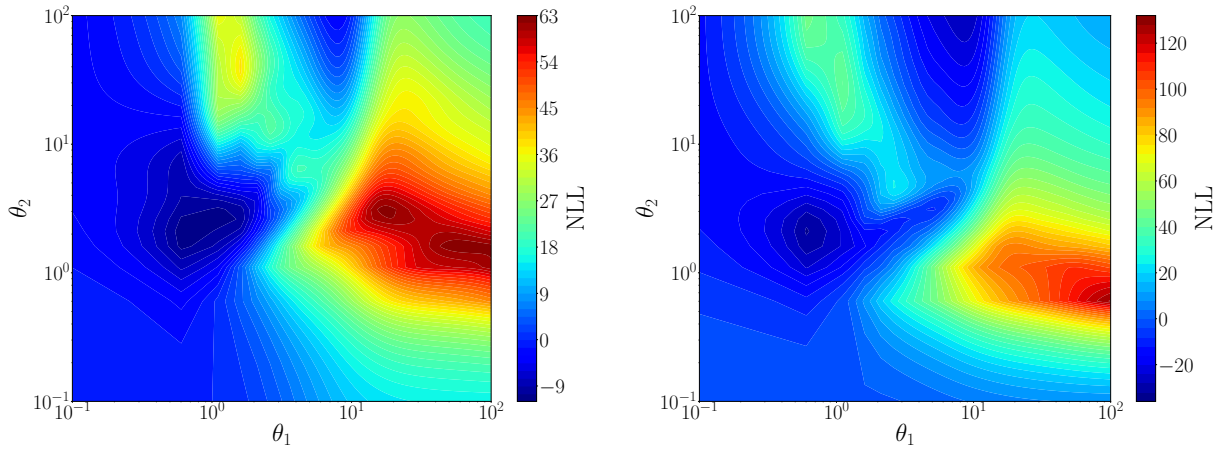
might require less number of iterations.

6.2 Multiple stopping criteria for a more robust active learning strategy

The importance of defining the appropriate stopping criterion in an active learning strategy has been elucidated in Section 6.1.5. The various ways of defining these stopping criterion were also given. For each of the different categories of defining the stopping criterion, the limitations were highlighted. Key of the issues is the need for robustness. The stopping criterion for different problems needs to be adjusted for the different cases. This makes it case-dependent and hence, it becomes difficult to make the active learning process generalized and automated. In some other types of stopping criterion, more data are needed to validate the stopping point, which could increase the computational complexity and cost without any guarantees that the model accuracy will improve. Hence, it becomes important to have robust and multiple criteria for active learning strategies.

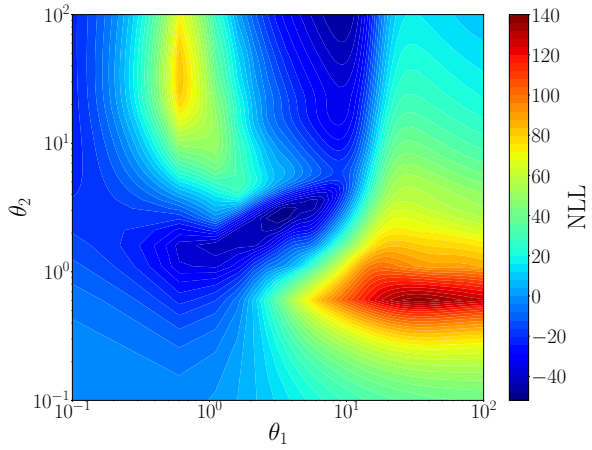
Towards defining effective and robust stopping criteria, we start by looking into the likelihood profiles and try to get enough intuition which can help us with our goal. We propose using multimodal function such as the camel back function. With the function, we will look into the likelihood profile in the hyperparameter space using different sample sizes. Because the end goal is to build models that have high accuracy, we will also look into the contour plot showing the model performance of the hyperparameter space. In the end, we will link the likelihood to the model accuracy, and use the information obtained to suggest necessary strategies that can help us define a stopping criterion.

We draw 20, 30, 40 and 50 sample points using LHS. With each point, we obtain their corresponding function value. We then define an hyperparameter space $\theta \in [10^{-1}, 10^{2}]$ for both axes. From this hyperparameter space, we draw 40 000 points with 200 points in each hyperparameter axis. Using the Gaussian kernel as the covariance function, we evaluate the NLL estimate of each point and show the responses using a contour plot. Figure 6.2 shows the likelihood contour plots using 20, 30, 40 and 50 sample sizes. For the model accuracy study, we evaluate the performance of the model at different hyperparameter values using 40 000 fixed test points. We use the NRMSE here to estimate the model accu-

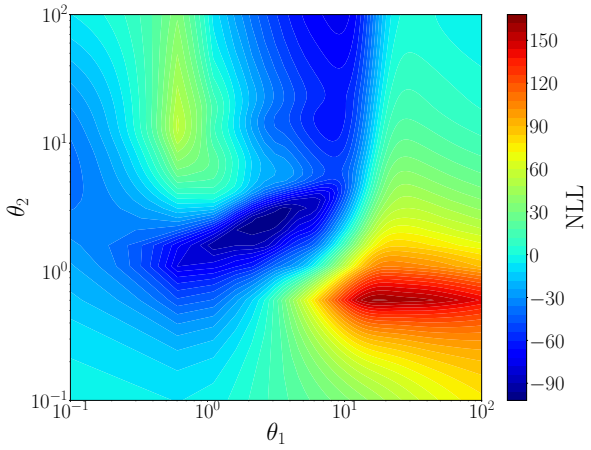


(a) $N_s = 20$

(b) $N_s = 30$



(c) $N_s = 40$

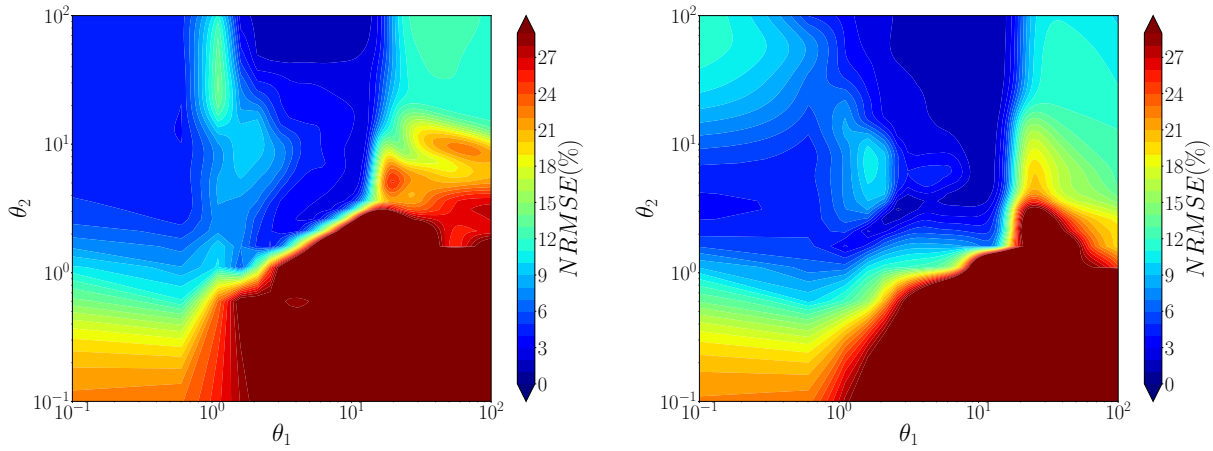


(d) $N_s = 50$

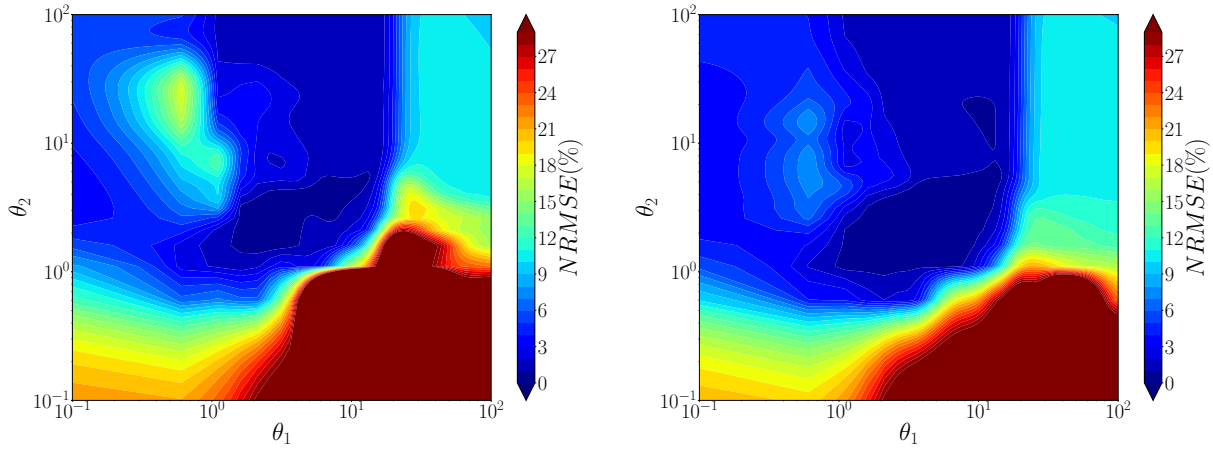
Figure 6.2: Likelihood contour plots for the camel back function with various sample sizes.

racy at each point. We present the outcome of the experiment for the various sample sizes using contour plots as shown in Figure 6.3. For better interpretation of the model accuracy, we limit the error on the contour plots to 30%. For the likelihood and model accuracy contour plots, the colour bar stands for the NLL estimate and the NRMSE, respectively. For all the contour plots, θ_1 and θ_2 are plotted on the x- and y- axes, respectively.

It is important to mention that maximizing the log-likelihood estimate is the same with minimizing the NLL estimate. Hence, we will use the terms interchangeably. With the likelihood contour profiles shown in Figure 6.2, the possible likelihood estimate tends to notably increase with the increase in sample size. With a sample size of 20, a very small likelihood is achievable within the hyperparameter space. However, with increase in the



(a) $N_s = 20$ (b) $N_s = 30$



(c) $N_s = 40$ (d) $N_s = 50$

Figure 6.3: Model accuracy contour plots for the camel back function with various sample sizes.

sample size to 30 points, the previous optima region shrunk and there is an emergence of a new optimal region in the contour profile. With further increase in sample size to 40 points, the previous new region became more prominent, and a slight shift in the maximum likelihood region to the right. Following another increment in the sample size to 50 points, the disappearance of multiple optimum regions is observed. With 50 sample points, it becomes easier to maximize the likelihood function. The instability of the optimum region especially with smaller sample sizes show the need for more samples. Also, when the optimum region is sufficiently developed, it could translate to the sufficiency of existing sample points, and could signify the need to not add more sample points.

Now, it is important to understand how this affects the model accuracy. With a sample

size of 20, the error contour shows a promise region in the upper part of the contour. With an increase in the sample size to 30, a shift in the promise region is observed. With more increase in the sample points to 40, an entirely new promise region is developed. The expansion of the promise region is observed with further increase in the sample size to 50 points. This could also translate to the establishment of a well-sampled function and the need to stop adding more points.

By comparing the likelihood and model accuracy contours, we draw conclusion that when there are insufficient sample points, the hyperparameter optimization becomes more sensitive to the starting points and hence, could greatly affect model performance. We also conclude that we can estimate the performance of a model by investigating the likelihood profile. While increase in the possible likelihood might not necessarily lead to better performance, it is important that the likelihood profile is sufficiently developed. This means that the likelihood profile becomes stable and free from several optimum regions.

We seek to propose an active learning strategy that uses the maximum likelihood information to define a stopping criterion. We hope by using the likelihood estimate, it becomes possible to make the stopping criterion robust and extensible to many problems. We propose capturing the relative change in MLE before and after adding a new point. This can be done using Equation 6.8,

$$\text{Absolute change in likelihood, } e_t = \frac{|LL_{t+1} - LL_t|}{LL_t}, \tag{6.8}$$

where t is the current iteration. LL_t and LL_{t+1} are the maximum log-likelihood estimate before and after the addition of a new point in t. It is equally important to define a threshold ϵ that will be monitored. We also proposed that 5% or 0.05 relative error be used as the stopping criterion. Because it is possible that this value is reached even when the function profile is not adequately learned, we propose that the values of few successive iterations be considered in defining convergence of the learning process. This can be achieved by capturing the last n_e set of likelihood changes. We make it necessary that these values are all less than the threshold ϵ and are in decreasing order, that is, $e_{t+1} < e_t$ for convergence to occur.

We propose using a weighted expected prediction error (w-EPE) as the learning function of our active learning strategy. To keep it as simple as possible, we use equal weight

of 0.5 for both the exploitation and exploration parts of EPE. It is common that sometimes the learning functions give new points that are similar to existing points in the sample data. When several points in the sample data are close to one another, it can lead to serious numerical issues when such data are used for modeling. To avoid this occurrence, we propose evaluating the quality of the new points using EI. In the case where the value of the EI is zero, such a point is rejected. When this happens, we propose repeating this process as much as possible to find a new point and evaluate the expected improvement. We set an upper limit c_{max} to the number of possible trials. In each trial, a new starting point is selected for the optimization of the defined learning function. In the case where c_{max} is reached, the active learning process is stopped. It is assumed that no more addition to the sample data can have any significant improvement. We denote this multi-criteria based active learning strategy as AL_{MC}. We summarize our proposed method in Algorithm 8. Figure 6.4 shows the flowchart of the proposed active learning strategy. In summary, the four stopping criteria used are:

Likelihood-based criterion: At each iteration t, the log-likelihood estimate, LL_t is compared to the next value, LL_{t+1}. For convergence to be reached, it is needed that the absolute change in the log-likelihood estimate be less than or equal to a defined threshold value, ϵ. For this criterion, Equation 6.9 must be satisfied.

$$e_t = \frac{|LL_t - LL_{t-1}|}{LL_{t-1}} \leqslant \epsilon \tag{6.9}$$

Continuous successive iteration: When the likelihood-based criterion is met, successive absolute change in likelihood for few last iterations, n_e, are compared. For convergence to be reached, it becomes mandatory that a decrease in the absolute change in likelihood occurs. This is to ensure progressive improvement for n_e iterations. Hence, Equation 6.10 must be satisfied.

$$e_t > e_{t+1} \tag{6.10}$$

Maximum number of stalls criteria: We define *stall* as a situation where the candidate point would not improve the function being learned. This is a situation where the EI value of the candidate point is zero. Whenever this happens, a new candidate

point is selected by maximizing the learning function with different initial values. The initial values are always selected within the defined bounds. The number of stalls, c, is incremented by one. In a case where after c_{max} trials, no improvement of any sort is seen, the learning process is stopped. Hence, convergence can also be reached if Equation 6.11 is met. It is important to mention that c_{max} is not the maximum number of points added. Rather, it is the number of permissible trials in each iteration.

$$c > c_{max} \tag{6.11}$$

Computational resource limitation: This criterion is set to prevent the active learning strategy to continue in a infinite loop. It is enforced as a limitation to the number of possible iterations. The maximum number of iteration, T_{max} is usually set to a reasonably high value. For this criteria, Equation 6.12 has to be met.

$$t > T_{max} \tag{6.12}$$

6.3 Test cases

To evaluate the converging ability and the robustness of our proposed method in active learning strategies, it is important to test using several cases. It is desirable to also assess the accuracy of models trained with the samples drawn with the proposed method. In order to achieve this, it is necessary to use algebraic functions. We propose using common multimodal functions for this purpose. We will use the Himmelblau, camel back and Branin functions since we have some understanding about their convergence. The previous results are presented in Section 3.5.1. The magnitude of multimodality varies in different functions. Some functions such as Ackley function [180, 41] tend to have several peaks in the profile. The Ackley function can be evaluated using Equation 6.13. In this chapter, we set the values of a, b, c to 20, 0.2 and 2π, respectively for the Ackley function. For ease of visualization and interpretation of results, we limit the dimension, N_d of the Ackley function to 2. Figure 6.5 show both the response surface and the contour plot

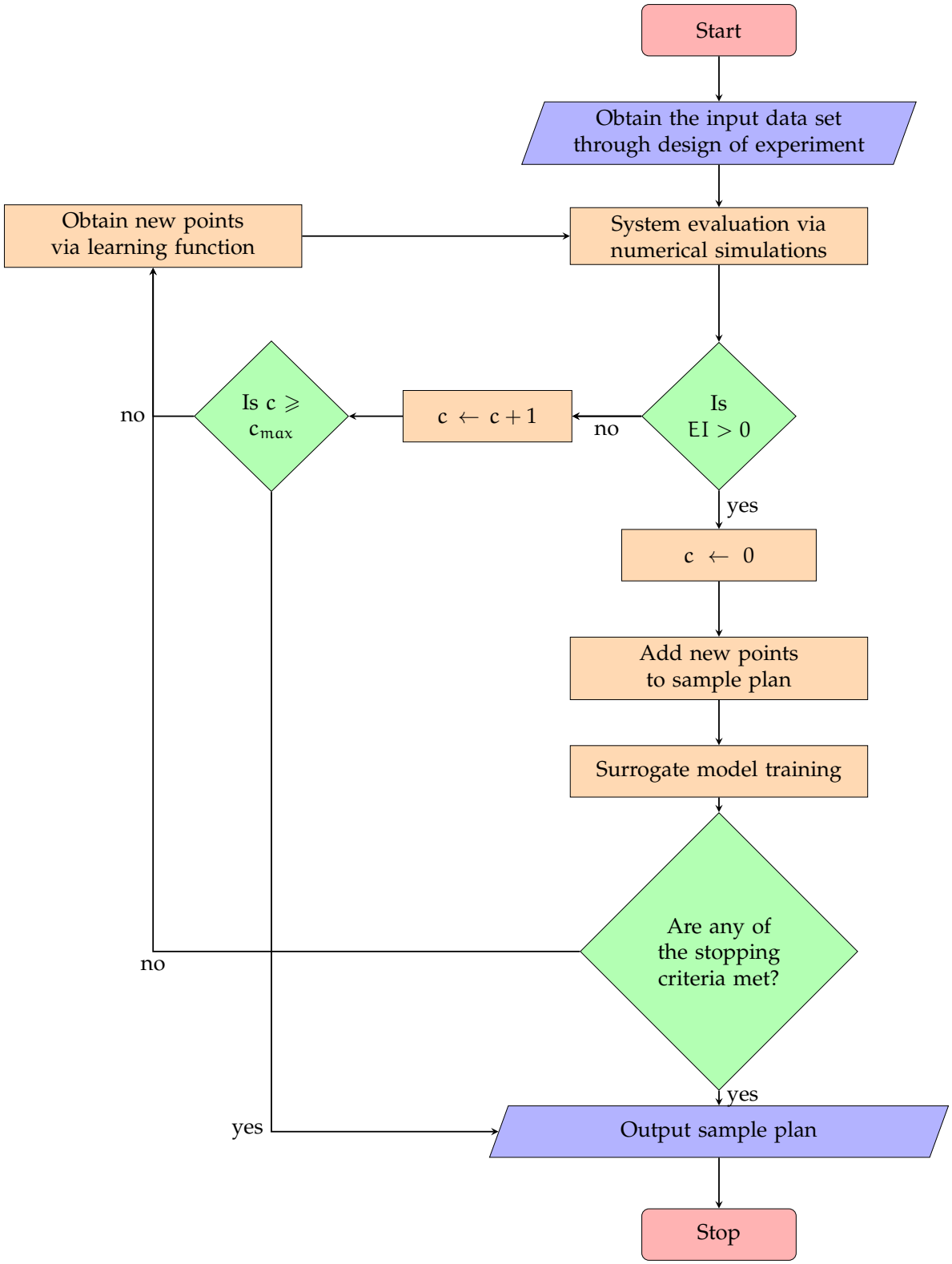


Figure 6.4: Flowchart showing the proposed multiple criteria active learning strategy.

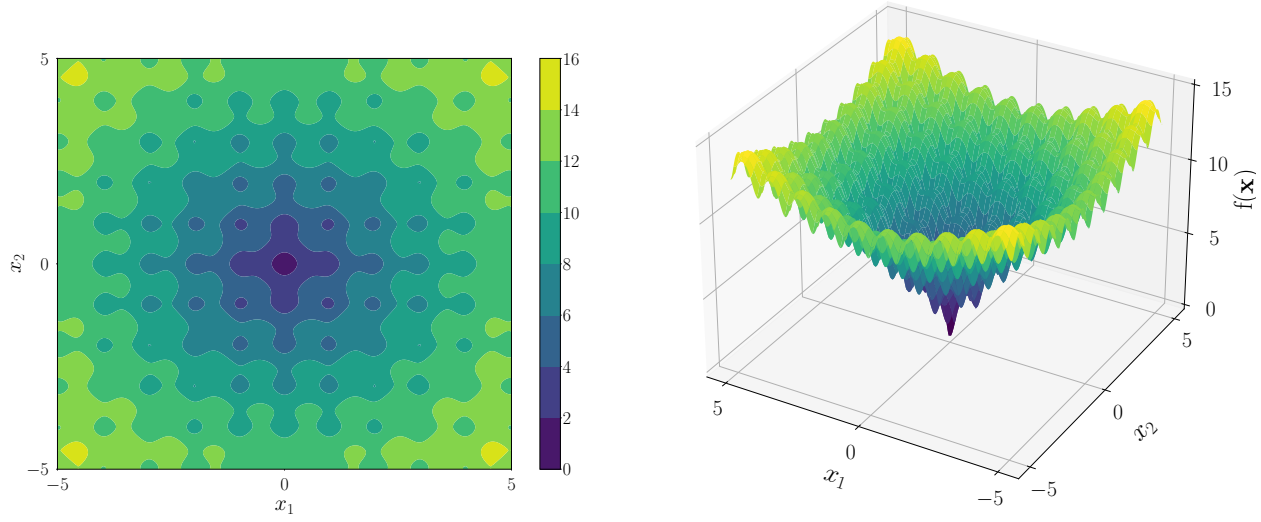
Algorithm 8 Active learning with multiple stopping criteria

Input: Function, $f(.)$, initial sample size, n_i, threshold point, ϵ, maximum iterations, T_{max}, maximum improvement check c_{max}, size of considered likelihood change, n_e
Output: $\mathbf{x}^{opt}$, $\mathbf{y}^{opt}$

Draw initial samples, $\mathbf{x_1}$
Obtain the functional value of the input data, $y_1 = f(\mathbf{x}_1)$
Initialize iteration, $t = 1$
while $t < T_{max}$ **do**
    Calculate GP posterior $q_t(f|S_t)$
    Select starting point within $\mathbf{x}$-space
    Obtain next proposed point by maximizing the learning function: $\mathbf{x}_{t+1}^{proposed} = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, 0.5\left(f(\mathbf{x}) - \hat{f}(\mathbf{x})\right)^2 + 0.5\hat{s}(\mathbf{x})$
    Compute the expected improvement of the proposed point, $EI(\mathbf{x}_{t+1}^{proposed})$
    $c \leftarrow 0$
    while $EI(\mathbf{x}_{t+1}^{proposed}) = 0$ **OR** $c < c_{max}$ **do**
        Reject $\mathbf{x}_{t+1}^{proposed}$
        Select another starting point within $\mathbf{x}$-space
        Obtain next proposed point by maximizing the learning function: $\mathbf{x}_{t+1}^{proposed} = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, 0.5\left(f(\mathbf{x}) - \hat{f}(\mathbf{x})\right)^2 + 0.5\hat{s}(\mathbf{x})$
        $c \leftarrow c + 1$
    end while
    if $c \geqslant c_{max}$ **then**
        break
    else
        Accept $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_{t+1}^{proposed}$
    end if
    Obtain functional value of new point: $\mathbf{y}_{t+1} = f(\mathbf{x}_{t+1})$
    Update dataset: $S_{t+1} \leftarrow S_t \cup \{(\mathbf{x}_{t+1}, \mathbf{y}_{t+1})\}$
    Calculate GP posterior $q_{t+1}(f|S_{t+1})$
    Initialize E = {}
    Compute absolute change in likelihood, e_t
    $E \leftarrow E \cup e_t$
    if $e_t < \epsilon$ **then**
        for $j = t - n_e, t + 1 - n_e, ..., t$ **do**
            if $e_j < e_{j+1}$ **OR** $e_j > \epsilon$ **then**
                break
            end if
        end for
    end if
    $t \leftarrow t + 1$
end while

of a two-dimensional Ackley function. The presence of many peaks in the profile renders such a function difficult to model. For effective modeling of such a function, many sample points, at the "right" locations, will be required. Such functions are desirable for testing active learning methods. Hence, the Ackley function will also be used as a test case.

$$f(\mathbf{x}) = -a\ \exp\left(-b\sqrt{\frac{1}{N_d}\sum_{i=1}^{N_d} x_i^2}\right) - \exp\left(\frac{1}{N_d}\sum_{i=1}^{N_d}\cos(c\ x_i)\right) + \exp(1) + a \tag{6.13}$$



(a) Contour profile

(b) Response surface

Figure 6.5: Contour profile and response surface of the Ackley function.

6.4 Results and Discussion

In this section, the test cases introduced in Section 6.3 will be used for benchmarking studies. To be specific, we present and discuss results showing the convergence behaviour, number of samples added, the learning time and the accuracy of models developed with different methods. We also look into the effect of kernels selection on the performance of models developed from the learned samples.

As a baseline for the benchmark, we use the LOOCV technique proposed by Dubourg *et al.* [60]. In the method, the leave-one-out estimate of the mean square error (MSE) is used to compute the cross validation error of the samples drawn. This is obtained by using Equation 6.14,

$$\mathrm{LOOCV}_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathrm{MSE}_i. \tag{6.14}$$

We then take the square root of Equation 6.14 to obtain an approximation of the error, e. The error of a sample with N_s number of points is obtained using Equation 6.15,

$$e_{N_s} = \sqrt{\mathrm{LOOCV}_{N_s}}. \tag{6.15}$$

We compare the cross validation error of two successive iterations and stop the learning process once the value reaches a defined point as advised by Dubourg *et al.*. Similar to the authors, we introduce a term, α, based on the error estimate to define a stopping point. The stopping criterion term, α, can be obtained by measuring the absolute change in error, as shown below,

$$\alpha = \frac{|e_{N_s+1} - e_{N_s}|}{e_{N_s}}, \tag{6.16}$$

where e_{N_s} and e_{N_s+1} are the approximated error of the sample points before and after the addition of the new point, respectively. For the value of α, we use 0.01 and 0.05 as the cross validation based stopping criterion. We use $\mathrm{AL}_{\alpha=0.01}$ and $\mathrm{AL}_{\alpha=0.05}$ to represent the active learning strategies having the values of α as 0.01 and 0.05, respectively.

With the baseline methods discussed, we set the parameters for our method. Since our test cases are all two-dimensional, we set the maximum number of iteration, T_{max} to 1000. From Algorithm 8, our method considers the last n_e number of absolute likelihood change in defining the stopping point. For our benchmark case, we fix the value of n_e to four. This is to ensure that continuous progress is indeed met before stopping of the learning process is proposed. In the case of the last four likelihood change meeting the stopping criteria, the last three points are discarded. This is to reduce the number of redundant points. For each of the absolute likelihood changes considered, they are expected to be

less than or equal to the threshold, ϵ. We use $\epsilon = 0.05$ as the threshold point. In the case where there is no expected improvement of a proposed new point, our algorithm continues to search the design space for improvement by using a new starting point in every iteration while optimizing the EI function. While this process will indeed improve the exploratory tendencies of our method, it is important to put a limit to the number of times the exploration can be done. We set an upper limit, c_{max} of 30. Hence, in a particular iteration t, our algorithm can repeat the learning process 30 times to pick a point that can improve the function being learned. In the case where c_{max} is reached, we stop the entire learning process as we assume that no further improvement can be made. Lastly, we use the Gaussian kernel in estimating the Gaussian process posteriors.

For the benchmark, we use an initial sample size of 20 for the different methods and test cases. The discussion on the convergence and number are samples added are given in Section 6.4.1. Sections 6.4.2 and 6.4.3 provide explanation on the learning time and model accuracy, respectively.

6.4.1 Convergence and number of samples added (N_s^+)

With the four test cases, we monitor their convergence using our method and the baseline methods. For each of the test functions, the learning process starts from the initial 20 sample points. As more points are added, the changes in likelihood and cross validation error are captured. Figure 6.6 shows the fluctuations in the likelihood estimate of each of the test functions as well as when the learning process is stopped. Figure 6.9 and 6.8 show the change in the error as well as the stopping point with 0.01 and 0.05 α values, respectively.

In the Branin test case, as more points are added initially, a sharp change is observed. With 25 sample points, the change absolute likelihood estimate went below the 0.05 threshold line. However, after 25 sample points, the changes in the likelihood estimate only fluctuated within a very short range. This could mean that after 25 sample points, the active learning algorithm could be stopped because the Branin function has been sufficiently learned. Due to the continuous small fluctuations, the learning continued till 33 sample point mark as shown in Figure 6.6a. Since the convergence happened after the 0.05

threshold mark, this means that further exploration of the Branin function after 33 sample points would not seek to improve the existing sample points. Comparing the convergence here with the previous convergence results for in Branin function as shown in Figure 3.9a, the convergence process is seen to initiate after 25 samples points and the convergence was finally realized with 35 sample points. With the $AL_{\alpha=0.05}$ and $AL_{\alpha=0.01}$ methods, the convergence was observed at 30 and 31 sample points as shown in Figures 6.8a and 6.9a, respectively. Since this happened between the 25 and 35 sample points mark, it could be interpreted as being reasonable. Hence, a good performance is expected when such points are used in modeling. Overall, 13, 10 and 11 new points points are added for the AL_{MC}, $AL_{\alpha=0.05}$ and $AL_{\alpha=0.01}$ methods, respectively.

Considering the camel back function, the AL_{MC} method has a similar behaviour with the learning process of the Branin function. With the camel back function, the threshold point is crossed when the samples reached 29 points. The learning process did not stop at that point because of the slight increase in the likelihood with an additional new point. Further addition of new points brought about continuous fluctuation in the absolute change in likelihood estimate. However, at the 39 sample points mark, convergence is reached as shown in Figure 6.6b. Hence, convergence was reached after the addition of 19 points. Comparing the convergence behaviour of the AL_{MC} method with the previously generated result in Figure 3.9e. The convergence is observed at 40 sample points mark. This shows the precision of the AL_{MC} method in sufficiently learning the camel back function. With both $AL_{\alpha=0.05}$ and $AL_{\alpha=0.01}$ methods, the convergence was reached after the additional of four new points to the sample data. Comparing with the convergence plot in Figure 3.9e, it shows that more sample points will still be required for the camel back function to be adequately learned.

With the Himmelblau function, at the 24 sample point mark, the absolute change in likelihood estimate goes past the threshold line with the AL_{MC} method as shown in Figure 6.6c. However, convergence does not occur until a total of 31 points make up the sample data. In other words, after 11 new points are added to the sample data, c_{max} is reached and the learning process is stopped. In the previously generated result as shown in Figure 3.9c, convergence starts at the 25 sample points mark and a full convergence is reached with a sample point of 30 points. So much similarity is seen between the AL_{MC}

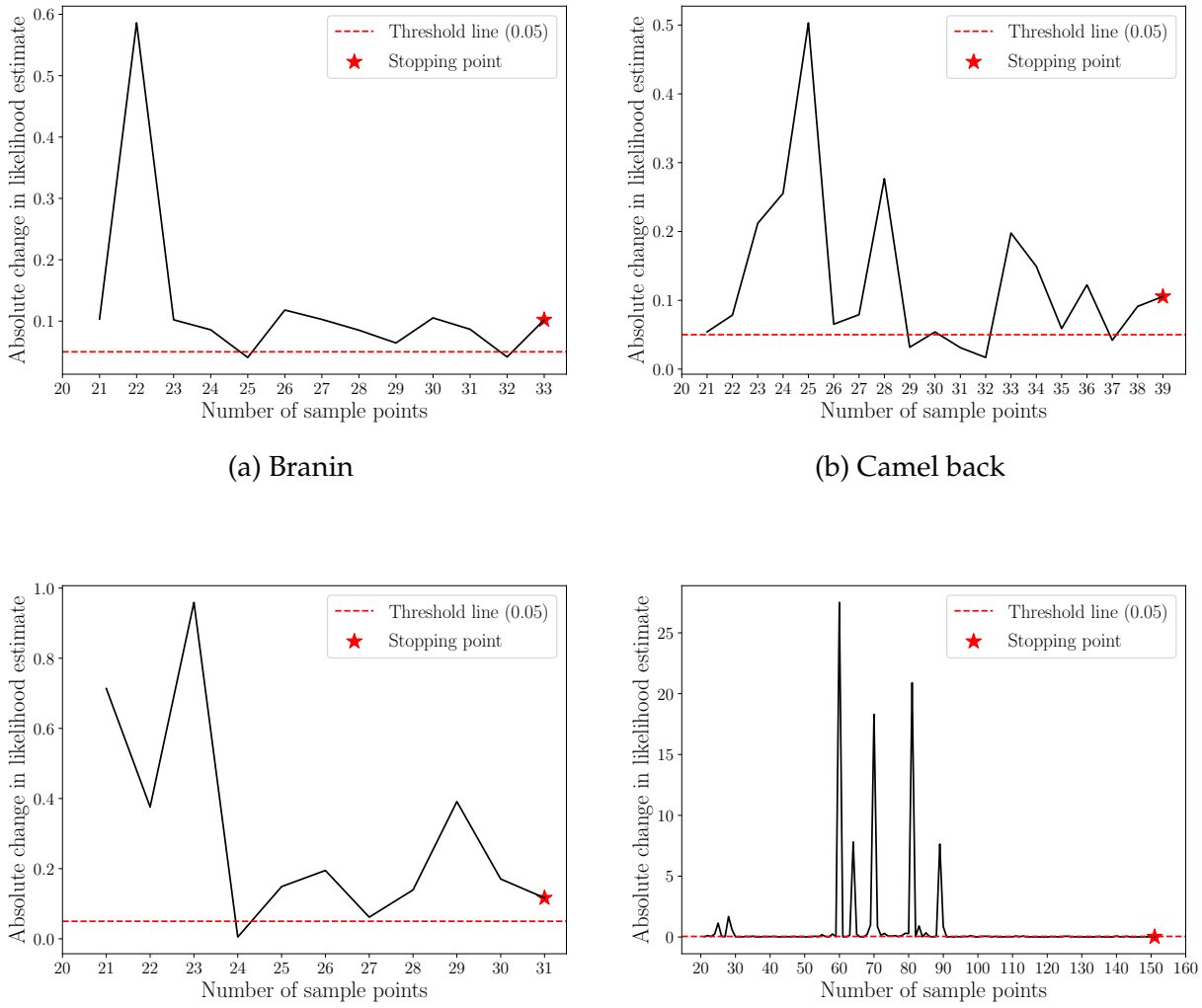
method and the convergence plot in Figure 6.6c. With the $AL_{\alpha=0.05}$ method, convergence is reached after the addition of just two points. However, convergence of the active learning process is reached after the addition of 21 new sample points with the $AL_{\alpha=0.01}$ method. While the number of points added might be sufficient and gives good model performance, it is considered to be overly sufficient. The direct implication of this is increased computational time in using the points in model training. This increase in time comes with little to no increase in modeling accuracy.

Considering the Ackley function, with the AL_{MC} method several points are added to the sample plan before convergence is reached as shown in Figure 6.6d. After the addition of 80 new sample points, several fluctuations below and above the threshold point are observed as shown in Figure 6.6d. The figure shows that the AL_{MC} method struggles with the Ackley function. The undulating profile of the Ackley function as shown in Figure 6.5 makes learning and modeling this function difficult. The presence of several modes posses great challenge to effectively modeling. However, after the addition of 131 new points, convergence is reached. This means that the exact point on convergence will be three point before the stopping point. Hence, the Ackley function can be adequately learned with 148 sample points. With the $AL_{\alpha=0.05}$ method, the active learning process converges after the addition of just eight points. This appears to be lower than expected, especially for a highly multimodal function like Ackley function. With the $AL_{\alpha=0.01}$ method, the absolute change in error moves closer to the threshold line at several times; however, convergence is only reached when 55 points are added. A total of 75 sample points obtained with the $AL_{\alpha=0.01}$ method appear to be more reasonable. The points will be put to a further test to measure the accuracy of models trained with them.

6.4.2 Learning time (L_T)

In as much as it is of good practice to adequately learn a function and be able to identify a stopping point, it is more important to do this as efficiently as possible. During the learning process, we capture the time it takes the methods to reach the stopping point for the test cases. The summary of the time spent by each method with test cases is shown in Table 6.2.

With the Branin function, 13 new sample points are added in approximately 17.99 sec-



(a) Branin

(b) Camel back

(c) Himmelblau

(d) Ackley

Figure 6.6: Plots showing the convergence and changes in likelihood estimate with increase in sample sizes for different functions.

onds with AL_{MC} method. With the $AL_{\alpha=0.05}$ and $AL_{\alpha=0.01}$ methods, a significant learning time is expended in adding 10 and 11 points, respectively.

Considering the Himmelblau function test case, only two new sample points are added when the $AL_{\alpha=0.05}$ method is used. Even with the few points being added by $AL_{\alpha=0.05}$ method, the method has similar computational time with the AL_{MC} method. With the AL_{MC} method, 11 new sample points are added. The $AL_{\alpha=0.01}$ method, on the other hand, adds 21 new points to the initial sample data in 181.58 seconds. That is, the AL_{MC} method converges approximately 8.3 times faster than the $AL_{\alpha=0.01}$ method.

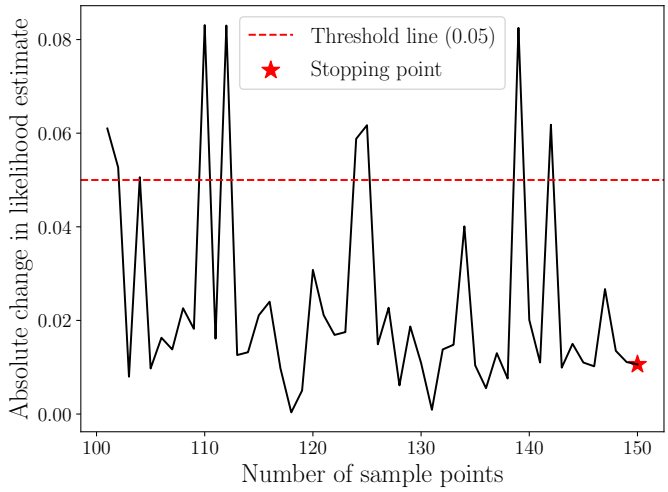


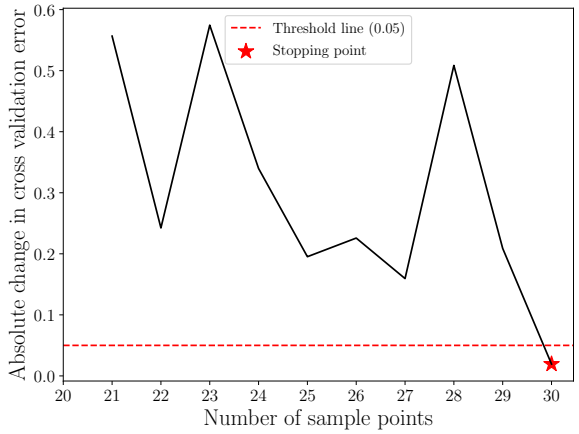
Figure 6.7: Plot showing the convergence and change in likelihood estimate for the Ackley function after 100 samples points.

Table 6.2: Table showing the convergence and model accuracy benchmark for different models using multimodal functions.

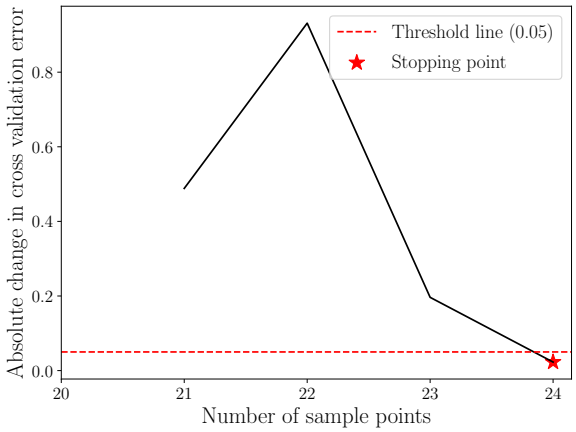
Method	N_s^+	L_T (s)	R^2	N_s^+	L_T (s)	R^2
		Branin			Himmelblau	
AL_{MC}	13	17.99	0.9999	11	21.87	0.9998
$AL_{\alpha=0.01}$	11	102.15	0.9999	21	181.58	0.9999
$AL_{\alpha=0.05}$	10	94.47	0.9994	2	22.33	0.8219
		Camel back			Ackley	
AL_{MC}	19	19.65	0.9977	128	288.05	0.9077
$AL_{\alpha=0.01}$	4	28.62	0.9154	55	15133.75	0.8429
$AL_{\alpha=0.05}$	4	27.04	0.9130	8	1420.70	0.7664

With the camel back function, the $AL_{\alpha=0.01}$ and $AL_{\alpha=0.05}$ stop adding points to the sample data after 28.62 and 27.04 seconds, respectively. Although the AL_{MC} method adds 19 new sample points to the initial sample data, it does so with less computational time of 19.65 seconds.

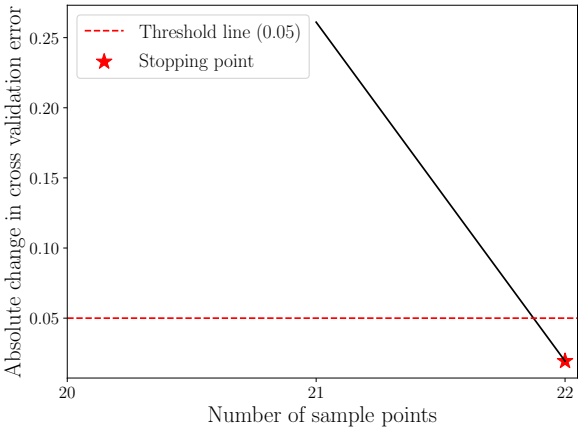
A considerable time is used in learning the Ackley function because of the inherent complexity in the profile. The AL_{MC} method adds extra 128 sample points to the sample plan in 288 seconds, while it takes the $AL_{\alpha=0.05}$ method about 1420.70 seconds to add just eight new points to the initial sample data. In the case of the $AL_{\alpha=0.01}$ method, an



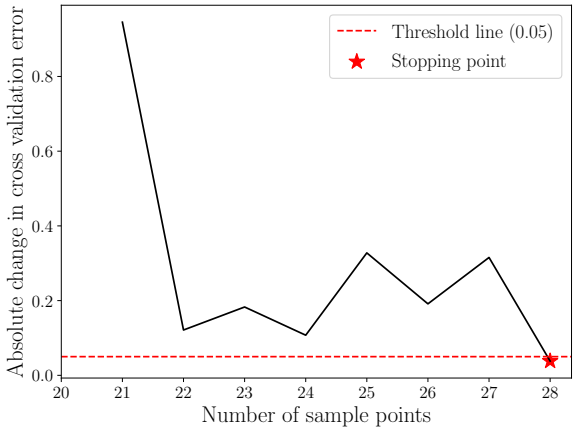
(a) Branin



(b) Camel back



(c) Himmelblau



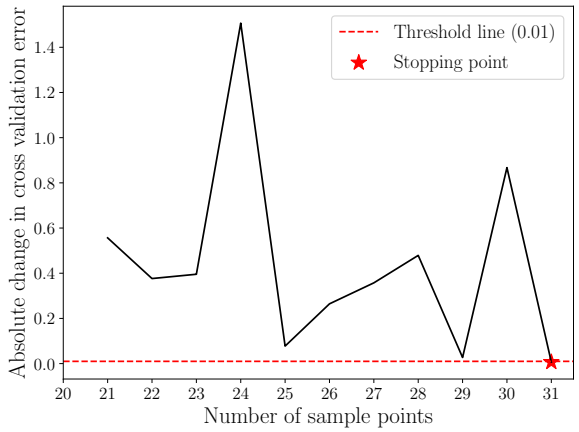
(d) Ackley

Figure 6.8: Plots showing the convergence and changes in cross validation error with increase in sample sizes for different functions ($\alpha = 0.05$).

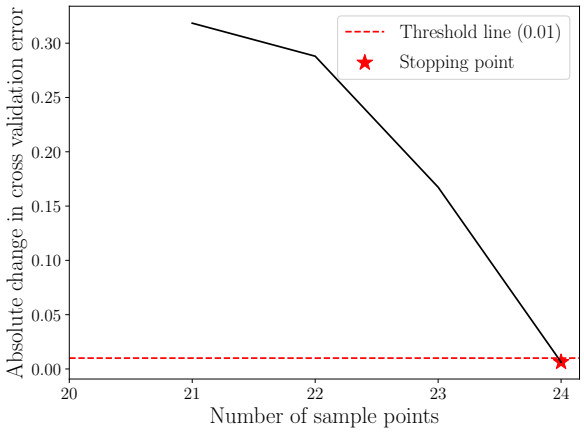
exhaustive 15133.75 seconds are used to add only 55 points to the sample plan.

6.4.3 Model accuracy

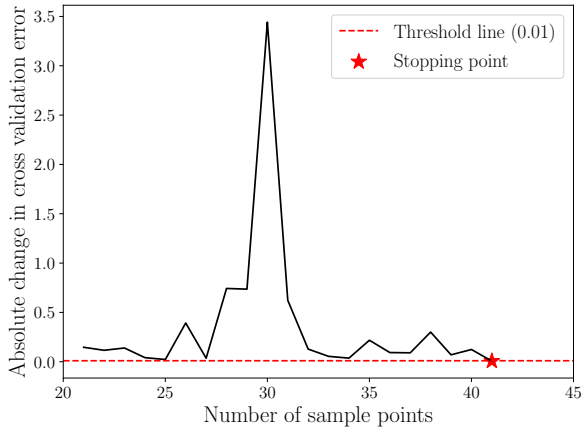
In Sections 6.4.1 and 6.4.2, we discuss the convergence and the learning time of the active learning methods. While they are both important in choosing a method, the accuracy of the models developed after the learning process is of utmost importance. In fact, the model accuracy is usually the end goal of deploying active learning methods. Here, we



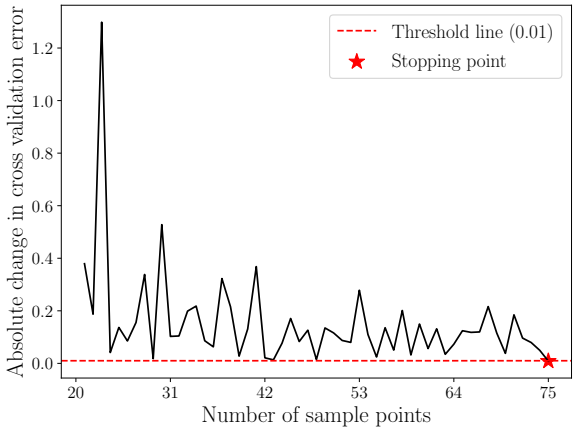
(a) Branin



(b) Camel back



(c) Himmelblau



(d) Ackley

Figure 6.9: Plots showing the convergence and changes in cross validation error with increase in sample sizes for different functions ($\alpha = 0.01$).

study the model accuracy of the learned functions. To make up the validation sets for the different test cases, we draw 500 random points with the input space. We then obtain the corresponding output by using the functions. Then, we use the learned sample data obtained to train a kriging model. The Gaussian kernel is used as the covariance function. We use the trained model to predict the output and compute the R^2 score between the predicted and real outputs. Table 6.2 summarizes the R^2 values for the methods on all the test cases. With the Branin function, the three methods have similar model performance with an almost perfect R^2 score. The $AL_{\alpha=0.01}$ and AL_{MC} methods have similar model

accuracy of about 0.9999 with the Himmelblau function. This shows that the two methods sufficiently learn the Himmelblau function. A reduced R^2 score of 0.8219 is observed with $AL_{\alpha=0.05}$ method for the Himmelblau function. The premature stop in the learning process as a result of quick convergence accounts for the loss in model accuracy. The $AL_{\alpha=0.01}$ and $AL_{\alpha=0.05}$ methods have a similar model performance with approximately 0.91 R^2 score. The AL_{MC} method, on the other hand, records an almost perfect R^2 score of 0.9977. When compared with the cross-validation based methods, the AL_{MC} method outshines. With the more complex Ackley function, none of the three methods is able to draw samples that have nearly perfect R^2 score. However, samples drawn using the AL_{MC} method perform best by achieving a model performance of 0.9077. Sample points drawn using the $AL_{\alpha=0.01}$ and $AL_{\alpha=0.05}$ methods achieve R^2 scored of 0.8429 and 0.7664, respectively.

In summary, the AL_{MC} method in this benchmark shows robustness in terms of consistent good performance associated with the samples drawn using the method. With the method, we do not have to alter the parameters. With all of the test cases, convergence is reached with the least number of samples capable of effectively modeling the functions in the least possible learning time.

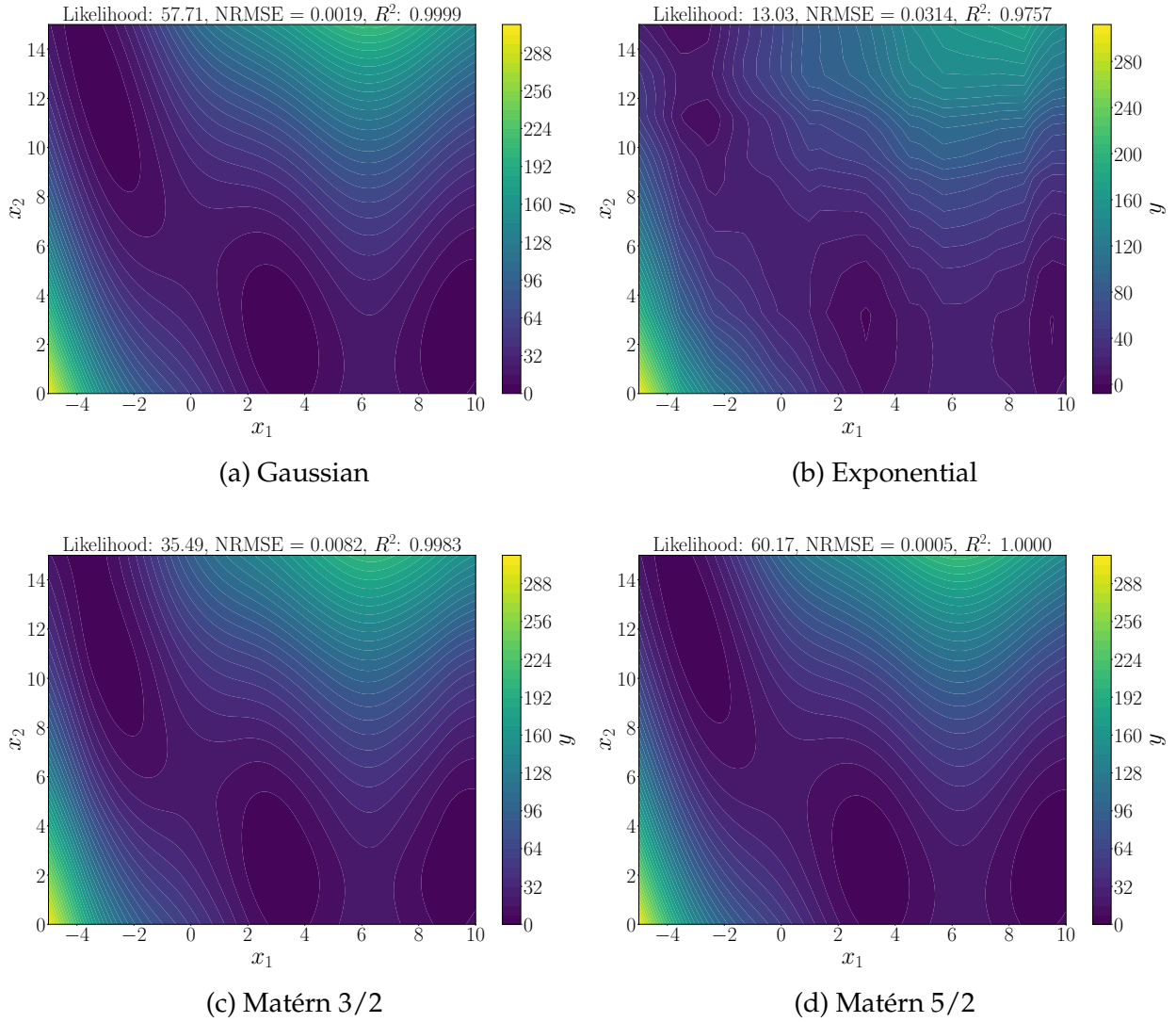
6.4.4 Effect of kernel selections on model accuracy of learned functions

In Section 6.4.3, the modeling performance of the learned data drawn using the different methods were discussed. In the benchmark, the model accuracy of the learned Ackley function was approximately 0.91 R^2 score even at convergence. Here, we explore the influence of kernel selections on the Ackley function as well as other functions presented in the test cases. We use the four commonly used kernels in engineering application.

For each kernel, we draw a 500×500 mesh grid with the bounds fixed to the different input domain. We then predict the output of each point which forms a node in the mesh grid. We then compare the predicted output and true output using the NRMSE and R^2 score. We display the estimated metrics as the title of each contour plot. We also show the MLE obtained after model training.

With the Branin function as shown in Figure 6.10, all kernels have similar performance, with the exception of the exponential kernel model. In the Branin contour generated with

the exponential kernel, a significant distortion in the contour lines is observed. A reduced model performance is also observed with the exponential kernel model as shown in Figure 6.10b. Hence, the Matérn kernels and the Gaussian kernel are deemed suitable for modeling smooth functions like the Branin function.



(a) Gaussian

(b) Exponential

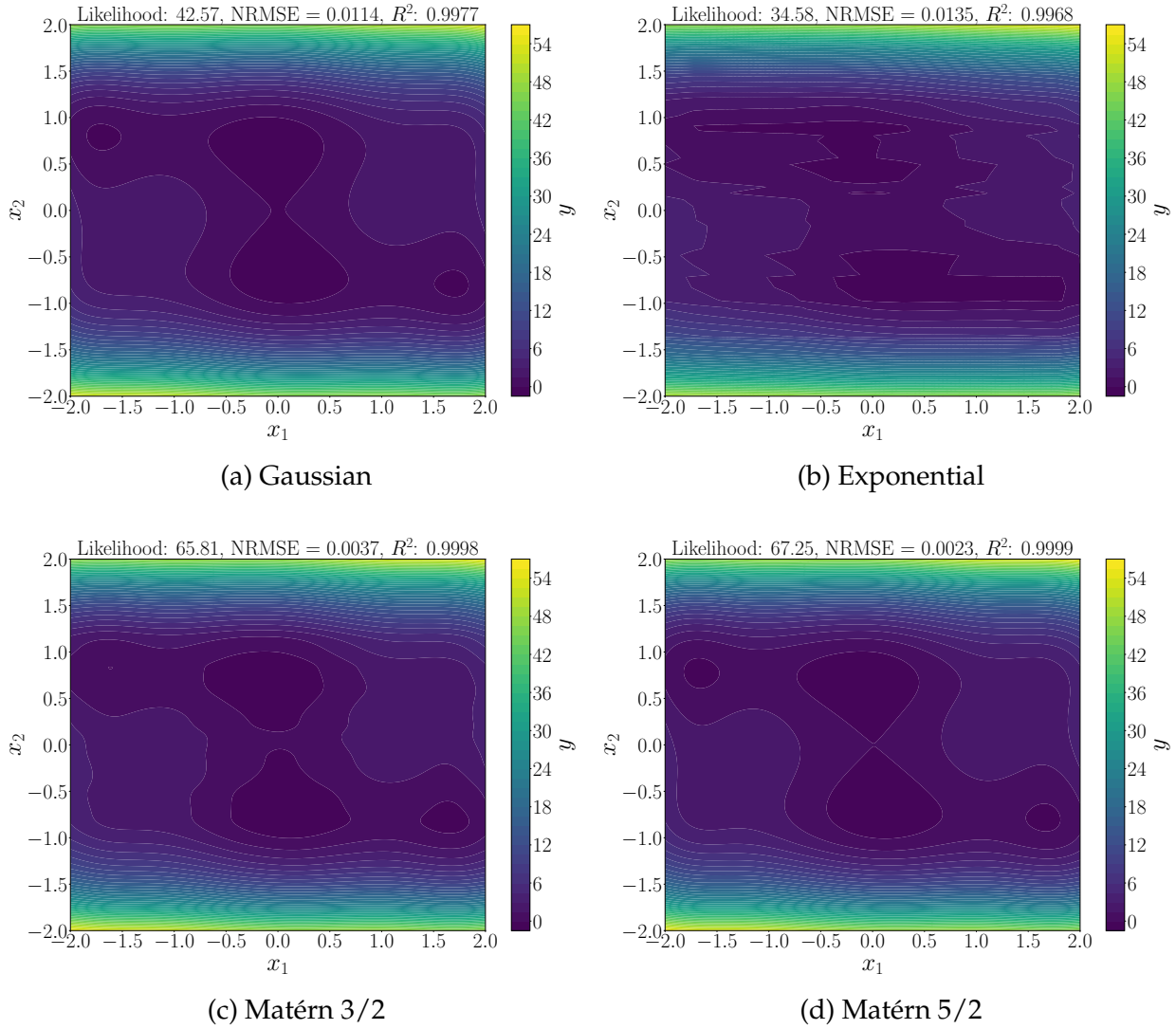
(c) Matérn 3/2

(d) Matérn 5/2

Figure 6.10: Plots showing the model performance of different kernels on the learned Branin function.

The camel back function showcases similar kernel performance when compared to those of the Branin function. The exponential kernel also has some form of alteration in the problem profile. Figure 6.11 shows the performance benchmark for different kernels using the camel back function. The case of the camel back function is similar to the Branin function. All four kernels have very good global model performance with the Gaussian

and Matérn 5/2 having a smoother contour surface. As shown in Figure 6.11b, the exponential kernel has a significant rough contour in spite of a good global accuracy. For the camel back function, it is desirable to use the Gaussian and Matérn 5/2 to obtain good global accuracy as well as a smooth profile.



(a) Gaussian

(b) Exponential

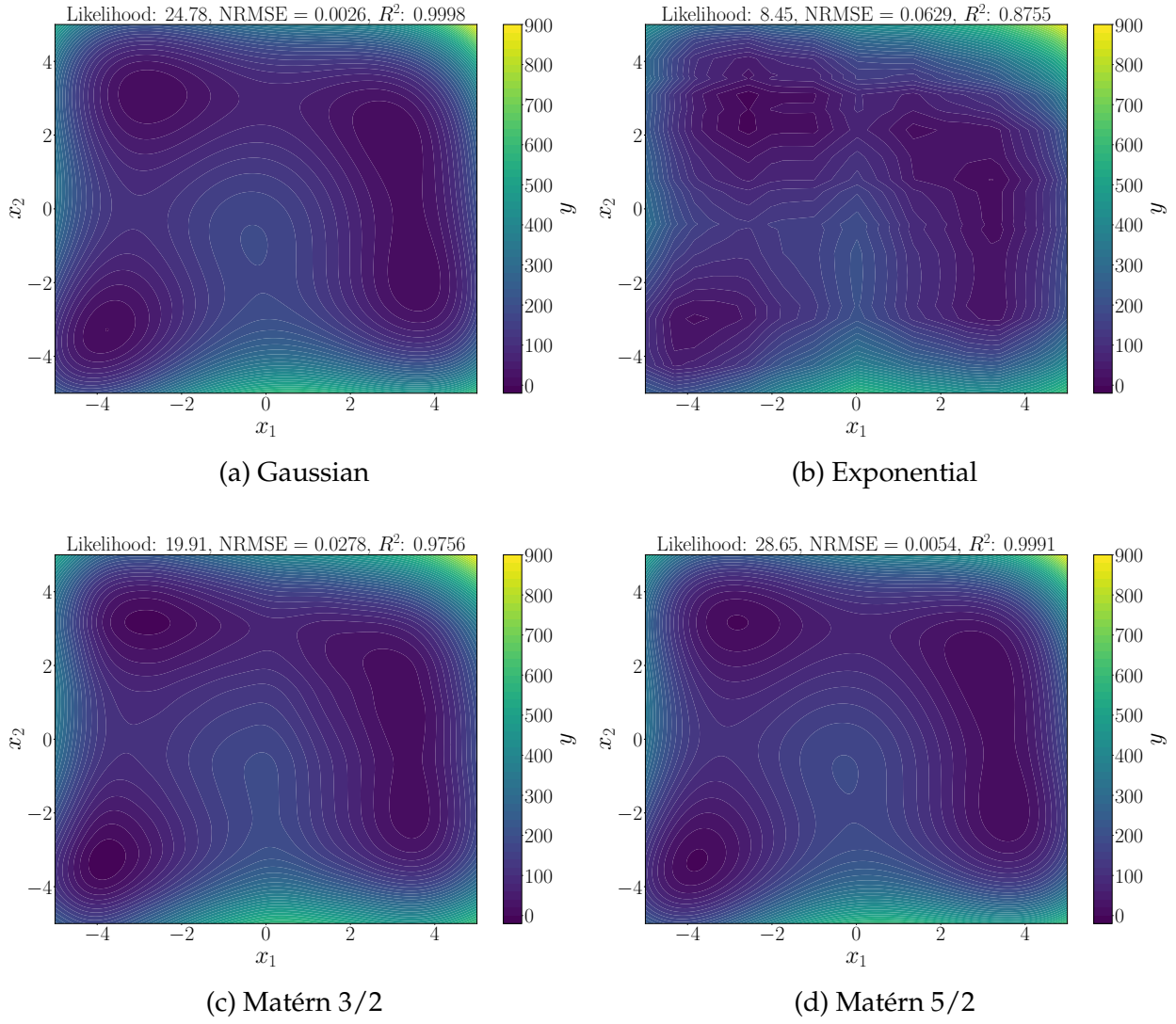
(c) Matérn 3/2

(d) Matérn 5/2

Figure 6.11: Plots showing the model performance of different kernels on the learned camel back function.

The kernel comparison plots with the learned Himmelblau function is shown in Figure 6.12. From the plots, the Gaussian and the Matérn 5/2 kernels adequately modeled the function with great model accuracy. The Matérn 3/2 kernel also modeled the function with a great model accuracy of 0.9756 R^2 score. Overall, apart from having good model accuracy, the Gaussian and both Matérn kernels are able to achieve a smooth contour of

the Himmelblau function. In the case of the exponential kernel as shown in Figure 6.12b, a reduced model accuracy of 0.8755 R^2 score is attained. A significant distortion in the contour is also observed with the exponential kernel. This shows the unsuitability of the exponential kernel for modeling the Himmelblau function.



(a) Gaussian

(b) Exponential

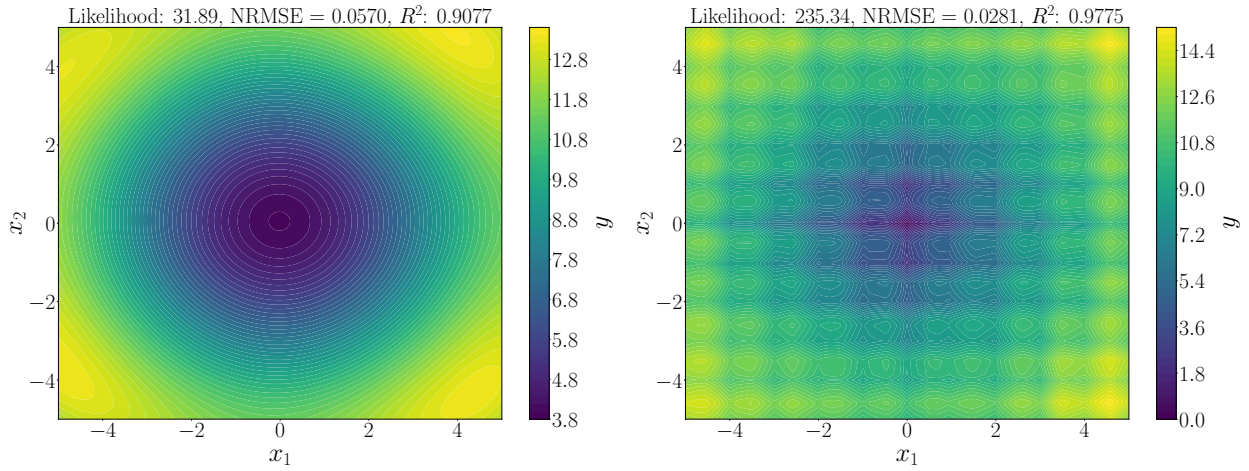
(c) Matérn 3/2

(d) Matérn 5/2

Figure 6.12: Plots showing the model performance of different kernels on the learned Himmelblau function.

The Ackley function is a much difficult problem to tackle. It takes our active learning method about 128 steps before convergence was reached. Even after convergence, the modeling accuracy from using a Gaussian kernel on the learned function to build a surrogate records a subpar performance of approximately 0.91 R^2 score. This is shown in Table 6.2. With the use of other kernels in building the surrogate, an interesting result

is seen. Figure 6.13 shows the contour plots and the model performance of the different kernels with the learned Ackley sample data. With other test cases (Branin, camel back and Himmelblau) , the Matérn 5/2 kernels have great performance on the learned functions. This is not the case here. In fact, the model performance plunged further with the Matérn 5/2 kernel. In spite of the relatively poor modeling performance of the Gaussian and Matérn 5/2 kernels, noticeable improvement in model capabilities is observed with the models trained with exponential and Matérn 3/2 kernels.



(a) Gaussian

(b) Exponential

(c) Matérn 3/2

(d) Matérn 5/2

Figure 6.13: Plots showing the model performance of different kernels on the learned Ackley function.

6.5 Summary

In this chapter, we gave sufficient background on active learning strategies and highlighted the steps involved in using the strategy. We gave explanations on some of the commonly used learning functions in active learning. Due to the similarities between Bayesian optimization and active learning, we explained the differences between them. The common acquisition functions used within the BO framework were briefly explained. We proceeded by highlighting the importance of stopping criteria in active learning. The four main ways of defining stopping criteria were given, and their limitations were mentioned. Due to the limitations of existing methods, we proposed that the likelihood information be exploited in defining a robust and precise stopping criterion. While we used the Gaussian process to obtain the likelihood information, the method can be easily extended to any Bayesian-based method. A quick study on the changes that happen to the likelihood estimate and model accuracy within the hyperparameter space was used as the basis of the proposed method. First, we ensured that the new points added to the existing sample data actually improved the *richness* of the sample data. We used the EI acquisition function to evaluate the quality of the proposed points. In the case where the points were redundant and did not add to the value of the sample plan, such points were rejected. Another new point was then proposed using the learning function, but with a different starting point. The new starting point was selected at random and used in the maximization of the learning function. In the case of rejection occurring in a small number of times, the learning process was stopped. This marked the first stopping criterion of our proposed method. In each iteration of the proposed active learning strategy, the likelihood estimate is monitored. The second stopping criterion was defined at the point where there was little to no improvement in the likelihood estimate with further addition of sample points. The last set of changes in the likelihood were captured and used in deciding the stopping point. The third criterion was based on the total number of iterations. This was put in place to prevent the computationally exhaustive learning of the proposed method.

The proposed method was then benchmarked against two cross-validation-based techniques using four multimodal functions. The parameters being benchmarked included the number of new samples added, learning time, and the accuracy of the model trained with the learned functions. In all the benchmarks, our method proved to be robust, as it

reached convergence easily without the need of adjusting the parameters. Also, the convergence rate of our method was promising. In all the cases, the learning process was stopped with reasonable sample sizes, which was in agreement with past results. The learning time used by our multi-criteria-based model was less compared to other methods. On the other hand, with the cross-validation method, the computational time it took to build and evaluate the cross-validation error limits the applicability of the method for high dimensional problems. Even with low-dimensional problems, there was the issue of robustness with the cross-validation-based approach. With the cross validation method from literature, there was a need to use a reasonable value of α, which was not always known. On the other hand, relying on the use of a maximum number of iterations for high-dimensional problems, as advised by Dubourg *et al.* [60], would lead to the addition of more points than required. In rare cases, the points would be insufficient.

From the convergence plots in both our method and the cross-validation-based methods from literature, the presence of peaks showed the danger of stopping an active learning method based on the number of final samples. Perhaps, the learning process might be stopped after a much distanced point had just been added to the sample data. This will lead to poorer performance of the model developed with such sample data after the new point addition. Hence, knowing where to stop the learning process is as important as using active learning strategies in the first place.

The kernel study on the learned function data revealed the suitability of using the Gaussian or either of the Matérn kernels for modeling relatively simpler problems. The relatively simpler problems are those with smooth profiles such as the Branin function. With more complex problems such as the Ackley function which is heavily multimodal, the exponential or Matérn 3/2 kernel were best suited, even after optimally learning the function with active learning. For such functions, it is expected that the Gaussian and Matérn 5/2 kernels will have suboptimal performance. Since Matérn 3/2 kernel was shown to have great performance in all the cases, we would recommend it as the first kernel to be considered when building surrogate models after an active learning procedure.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis, we addressed some of the fundamental issues with the use of surrogate models in solving complex problems that are common in engineering applications. In particular, we addressed issues with kernel selections, high-dimensional modeling, and the selection of optimal training samples with active learning strategies.

Considerable progress was made towards understanding and improving kernel selections in surrogate-modeling techniques such as kriging. The research started with understanding multiple kernel methods in Chapter 3. We then proposed the MiKL and MCKL, which used weighted-matrix to select and combine kernels, respectively. In these approaches, the weight matrix was optimized simultaneously with the model hyperparameters. The multiple-kernel methods helped model problems without the need to manually select the kernel for an application. The methods were also effective in improving the performance of models; however, care must be taken when using them. The use of non-performing kernels within the multiple kernel's framework could negatively affect the performance of the model. Another reason for caution is the increased computational training time observed with the multiple kernel models. In this thesis, it became clear that the model training time spontaneously increased when the problem dimension is greater than 12. The big difference in the computational training time of multiple kernel and single kernel models was a result of the many processes it required in building them. In this thesis, we showed that using a different kernel for each variable could lead to significant improvements in model accuracy for both engineering and algebraic problems. However, this can only be used for efficiently modeling low-dimensional problems.

In this thesis, we showed that the starting point in hyperparameter optimization could sometimes affect the end goal, which is the model accuracy of the surrogate model. We discovered that handling model training using different kernels and starting points for

model training could be helpful in obtaining surrogate models with good predictive abilities. To make the process efficient, the hyperparameter optimization can be handled with an optimization framework such as `Optuna`, which is capable of running the different sets in parallel with the focus on selecting the best combination that maximizes the likelihood estimate. Following a deeper look into the characteristic behaviour of common kernels used in engineering communities, we discovered that more than ever, the Gaussian kernel is more predisposed to having sub-optimal results. To tackle this challenge, we proposed pre-training the main model process by using the optimal hyperparameter from a model training process with the general Matérn kernel. The pre-training process helped the hyperparameter to get closer to the optimum. We showed that using this strategy did help to significantly improve the model performance of the Gaussian kernel and the former method helped to effortlessly select the best kernel that produces great model performance.

Regardless of the observed good performance, the methods developed in Chapters 3 and 4 are mostly applicable to low-dimensional problems. When kriging is used to model high-dimensional data, it takes so long for the models to train, which limits their usage in many real-world applications. While existing techniques such as the KPLS method helped with the significant reduction in computational training time, there was notable loss in the model accuracy. Such is the bane of many other model reduction methods. We traced the loss in model accuracy to the change in the original model structures. We proposed the use of feature-selection-based metric and a weight-bias approach. The feature-selection-based metric is used to capture the relationship between input variables and the output while the weight-bias is obtained through maximizing the MLE equation. The resulting model showed a significant reduction in model training time with little or no loss in model accuracy. In some cases, it even improved the accuracy of the model when compared to the conventional methods. Hence, we strongly recommend that researchers who attempt to contribute to the development of model reduction techniques should not only focus on benchmarking their proposed methods by comparing the obtained model accuracy on some set of test cases. We advise that the focus should be on approximating the model structures. In addition, we also recommend that the ellipsoid problem be used to validate model reduction techniques. This is because of the intuitive nature of the problem. With the ellipsoid problem, the input variables have a clear meaning and influence on the hy-

perparameter. The importance of the input variables of this problem increases with the dimension index. This should be reflected in the approximated hyperparameter values.

After the development of an effective model reduction method, the effect of kernels was evaluated on multimodal and high dimensional problems. In the research work, both conventional ordinary kriging and the reduced model were used to study this effect. It was discovered that kernel selection could affect multimodal problems differently. The degree of the multimodality of the problem needs to be taken into consideration. In this thesis, we proposed a test on functions to assess their degree of multimodality in both low and high dimensions. From our results, we recommend that the exponential or Matérn 3/2 kernel be used in modeling high-dimensional and highly-multimodal problems.

In many cases, the efforts in selecting an appropriate kernel would be fruitless if the sample points are not chosen well. Sometimes, a sizeable amount of sample points is required. Other times, the positioning of those points is of utmost importance. With active learning strategies, positioning of the sample points can be effectively done. This is because active learning strategies focus on enriching the sample data by adding informative new points. However, the right amount of points that needs to be added is usually unknown because it differs on a case-by-case basis. In this thesis, we proposed a multi-criteria algorithm to help identify where to stop adding new points. We showed that by keeping track of the likelihood estimate, we can define a good stopping point for active learning strategies. The algorithm proposed showed that it was robust, precise, and with good convergence when benchmarked using several test cases. We recommend that cross-validation based methods should be avoided within the active learning framework. This is due to the enormous computational requirement of the approach. We also recommend that the actual size of improvement of new points be estimated before adding to the sample plan. By doing so, over-reliance on the learning function to select informative point is avoided.

In closing, we demonstrated that we have been able to tackle challenges in kernel selection, hyperparameter approximation, and optimal sample selection in both low- and high-dimensional problems with various levels of complexities.

7.2 Future work

In the future, we hope to further improve on the work done in this thesis as well as explore new areas. The specific areas of improvement are discussed below;

Kernel selections in Gaussian process classification In this thesis, kernel selection is sufficiently explored in kriging interpolation technique. The knowledge obtained can be easily extended to Gaussian process regression (GPR). Afterwards, we hope it will also be worthwhile to extensively study how kernels affect Gaussian process classification (GPC) models. When this is done, we hope to develop algorithms to help with the selection of good and viable kernels for various classification task with minimum efforts.

Improved Gaussian process for heterogeneous noisy data In many real-world applications, the amount of noise in data is usually beyond the control of engineers and researchers. To aid ease of modeling, data with little to no noise are used for research. This is a quite reasonable practice because highly noisy data are difficult to properly model. However, this practice can sufficiently limit the types of problems that can be tackled. Matters become worse when noisy data from different sources are to be modeled. In the future, we will like to address such issues. Perhaps, reformulating Gaussian process to efficiently estimate the noise variance in each problem dimension and effectively handling them before they add up to large amount, that become really difficult to handle. This is usually due to the heavy distortion (deviation) of the true functional value.

Surrogate modeling for large datasets Since the 90s, there has been increasing focus on the use of Gaussian process as the choice of surrogate model. Due to the numerical instability that might arise when several training points are required, Gaussian process is usually limited to applications where small and medium-sized data are needed. With large datasets, other modeling techniques such as neural networks would be more suitable. With the recent introduction of deep Gaussian process (DGP) [40], it has been possible to extend Gaussian process to much larger datasets. However, due to the amount of Gaussian process-based models within the layers, a

considerable amount of training time would be required. In the future, we will assess the use of DGP for building surrogates as well as provide efficient and effective alternatives. What comes to mind is to develop an algorithm where an informative subset of a large dataset is drawn and used for model training. The model hyperparameters obtained from the subset can be used to approximate the hyperparameter for the large dataset. The bulk of the work here is to be able to draw a representative subset of the dataset. The active learning strategy with robust and precise stopping criteria developed in Chapter 6 is developed to achieve this purpose. We hope to stabilize the model development process and benchmark with existing state-of-the-art techniques. Our goal remains to develop a surrogate modeling technique that is capable of handling large dataset with lesser training time and great model performance.

Improved active learning techniques In the adaptive sampling work done in this thesis, a single new point is added in each iteration. This means that many steps will be required for very complex problems, which is not desirable. The first improvement will be to implement a batch update for new points to be added to the robust sampling plan framework. This will require fewer steps and faster convergence. The second point of improvement will be the dynamic balancing of exploration and exploitation within our framework. When this is done, we expect to see a notable improvement in the performance of our active learning method. The addition of *richer*—that is, more informative—new points will also lead to faster convergence to meet the prescribed stopping criteria. We will start by integrating the current state-of-the-art strategies for balancing, which are the maximizing expected prediction error (MEPE) [147] and the smart sampling algorithm (SSA) [87]. Currently, ten times the problem dimension is advised to be used as the initial sample size for adaptive sampling strategies. While it is reasonable to increase the sample size with increase in problem dimension for a defined N_d-dimensional function, two different functions with the same dimension might have very different level of complexities. Because of this, it is ideal that different initial sample sizes be used for them, to minimize the number of updates required in the more complex function. Motivated to tackle this challenge, we will also work on predicting an effective initial size for

adaptive sampling strategies, since the initial size used might have serious effect on model performance.

APPENDIX A

COMMONLY USED BENCHMARKING FUNCTIONS

A.1 Haupt Function

The Haupt function is a multimodal function expressed in two dimensions. In most cases, it is evaluated in the bounded domain between 0 and 4 (that is, $x_i \in [0, 4]$). Equation A.1 gives the analytical form of the Haupt function. The response surface plot of the function is shown in Figure A.1.

$$f = x_1 \sin(4x_1) + 1.1x_2 \sin(2x_2) \tag{A.1}$$

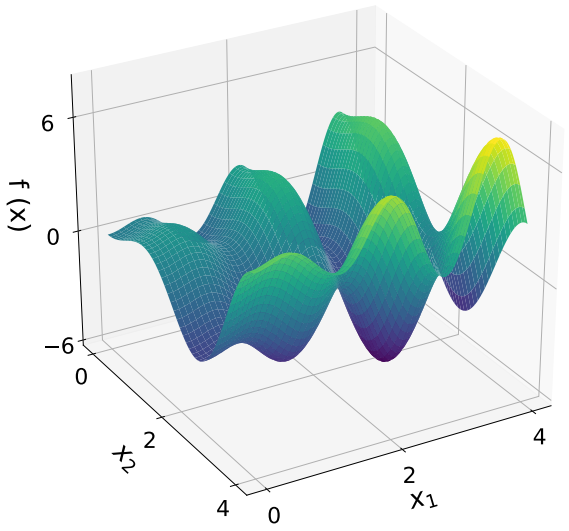


Figure A.1: Haupt function profile

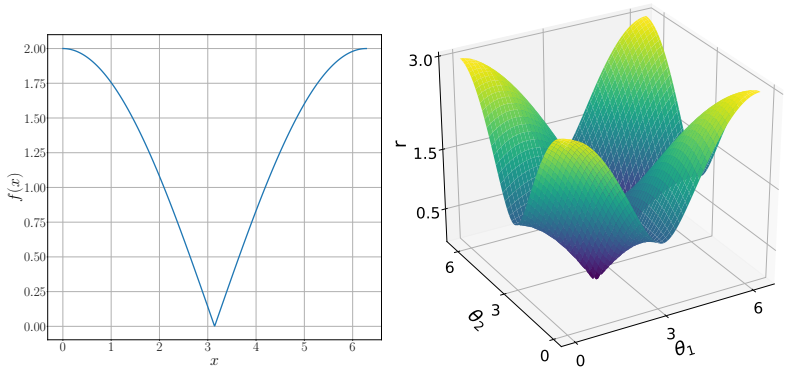
A.2 Robot Arm Function

For the purpose of this research, the robot arm function was modified to take on the form of an N_d-dimensional problem. This was done notwithstanding the typical interval of the

inputs of the function being angle($\theta \in [0, 2\pi]$) and length($L \in [0, 1]$) for each arm segment. We take the length of the arm to be fixed at 1.0, and vary the angle of each arm segment. The function output is left to be the distance to the tip of the robot arm, denoted by (r).

The one-dimensional and two-dimensional plots of the output with respect to the inputs are shown in Figure A.2(a) and (b) respectively. We observe that the function profile clearly shows a non-linear and symmetric behavior. Equation A.2 gives the explicit relationship between the variables of interest in the robot arm function.

$$r = \sqrt{\left(\sum_{i=1}^{n+1} L_i \cos\left(\sum_{j=1}^{i} \theta_j\right)\right)^2 + \left(\sum_{i=1}^{n+1} L_i \sin\left(\sum_{j=1}^{i} \theta_j\right)\right)^2} \tag{A.2}$$



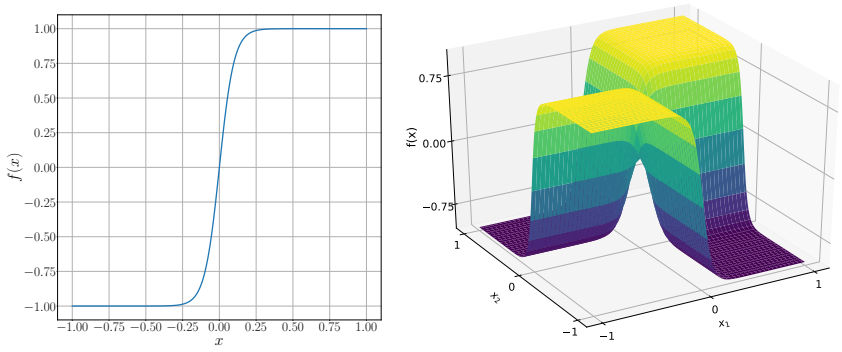
(a) 1D (b) 2D

Figure A.2: Robot arm function profile.

A.3 Tensor Product Hyperbolic Tangent Function

The tensor product hyperbolic tangent function has the property of being able to fit in multiple dimensions of n. It exhibits a unit step profile in one dimension. Its inputs x ranges between intervals: $x \in [-1, 1]$ and step function abruptness is regulated by parameter a. In this study, we keep the value of the abruptness parameter at 10. Figure A.3(a) and (b) illustrate the function profile in one and two dimensions respectively. Equation A.3 expresses the function profile as;

$$f = \prod_{i=1}^{n} \tanh(\alpha x_i) \tag{A.3}$$



(a) 1D (b) 2D

Figure A.3: Tensor product hyperbolic tangent function.

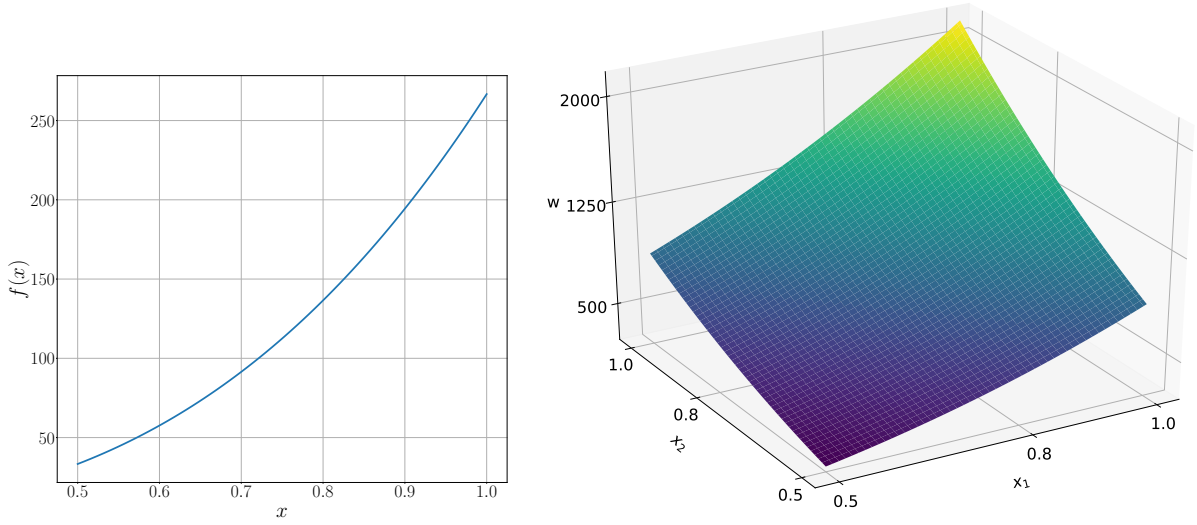
A.4 Cantilever Beam Problem

This is a linear, n-dimensional function. Its inputs are: length ($l \in [0.5, 1.0]$), width ($b \in [0.01, 0.05]$), height ($h \in [0.30, 0.65]$), Young's modulus E of individual elements and applied force at the tip P. The function output is the tip deflection (w). Figure A.4(a) and (b) illustrate the one and two dimensional plot of the function respectively. Equation A.4 expresses the function profile as;

$$w = \frac{P}{3E} \sum_{i=1}^{n} \left[\frac{12}{b_i h_i^3} \left(\left(\sum_{j=i}^{n} l_j \right)^3 - \left(\sum_{j=i+1}^{n} l_j \right)^3 \right) \right] \tag{A.4}$$

A.5 Branin Function

This is also known as the Branin-Hoo function. It is a two dimensional analytical function that is continuous and non-convex. It is commonly evaluated in the domain bounded as:



(a) 1D (b) 2D

Figure A.4: Cantilever beam function profile

$x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$. Equation A.5 expresses the function in its analytical form, while its equivalent response surface plot is shown in Figure A.5.

$$f(x_1, x_2) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10 \tag{A.5}$$

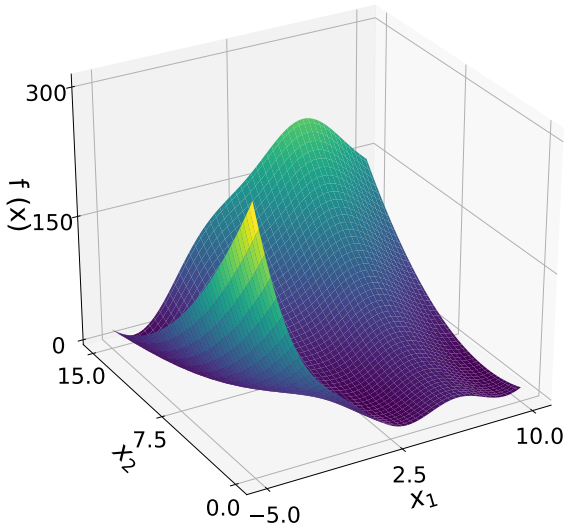


Figure A.5: Branin function profile

A.6 Multimodal 2D Function

This is a function with an interesting profile. It has one local minima and one local maxima. Also, it has one global minima and one global maxima. The function is usually evaluated within the input domain bounded by: $x_i \in [-2, 2]$. Its analytical form is expressed in Equation A.6, and Figure A.6 illustrates the associated response surface plot.

$$f = x_1 x_2 \sin x_1 + \frac{x_1^2}{10} + x_1 - 1.5x_2 \tag{A.6}$$

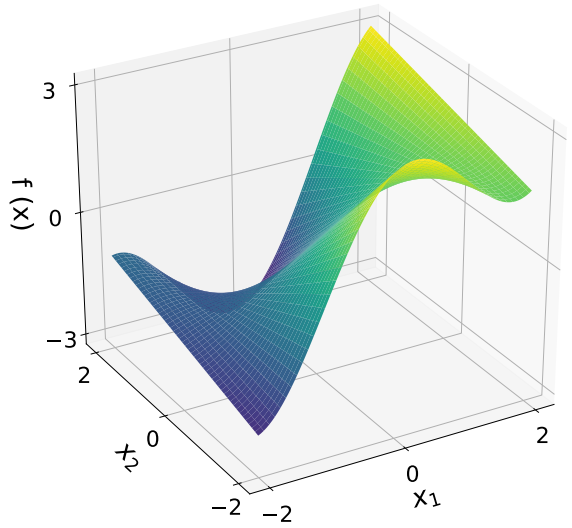


Figure A.6: Multimodal function profile

A.7 Camel back Function

This is explicitly known as the six hump camel function. It is simply known as the camel function and serves very well as a two dimensional analytical function. Its inputs are commonly defined on the domain: $x_1 \in [-3, 3]$ and $x_2 \in [-2, 2]$. The corresponding response surface plot of this function is illustrated in Figure A.7. The function's analytical form is given by Equation A.7:

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \tag{A.7}$$

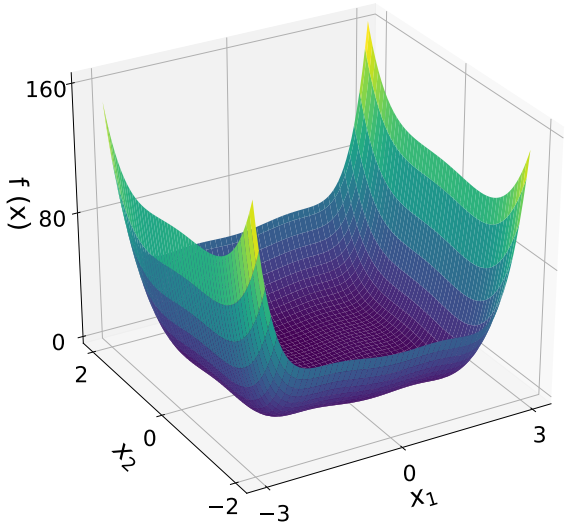


Figure A.7: Camel back function profile

A.8 Rosenbrock Function

The Rosenbrock function is also known as the Banana or Valley function. It is an analytical function with two dimensions. It is unimodal, and its global minimum lies along a parabolic, narrow valley. The Rosenbrock function is commonly evaluated in a domain bounded by: $x_i \in [-5, 10]$. Its analytical form is expressed in Equation A.8, and Figure A.8 illustrates the associated response surface plot.

$$f = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \tag{A.8}$$

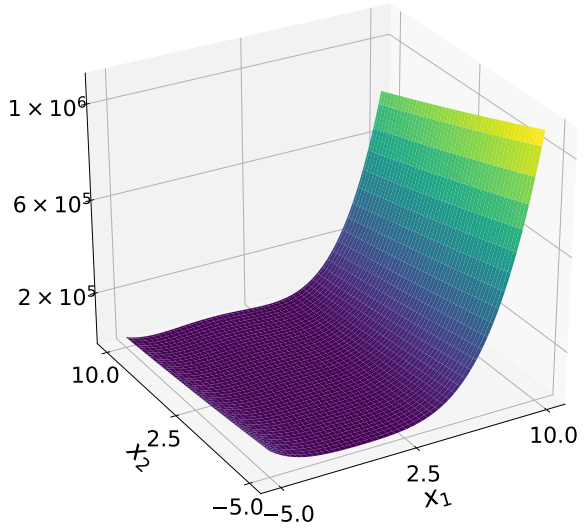


Figure A.8: Rosenbrock function profile

A.9 Hosaki Function

The Hosaki function is a common two dimensional function which is used for benchmarking. It has a shape similar to the hull of a ship. The function is usually evaluated within the input domain bounded by: $x_i \in [0, 5]$. Its analytical form is expressed in Equation A.9, and Figure A.9 illustrates the associated response surface plot.

$$f = \left(1 - 8x_1 + 7x_1^2 - \frac{7}{3}x_1^3 + \frac{1}{4}x_1^4\right) x_2^2 e^{-x_2} \tag{A.9}$$

A.10 Himmelblau function

This is a multimodal two-dimensional function with input defined on $x_i \in [-6, 6]$. It is known to have four local minimum and one local maximum. It is commonly used to evaluate the performance of optimization methods. Figure A.10 shows the function profile. The function can be expressed mathematically using Equation A.10.

$$f = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{A.10}$$

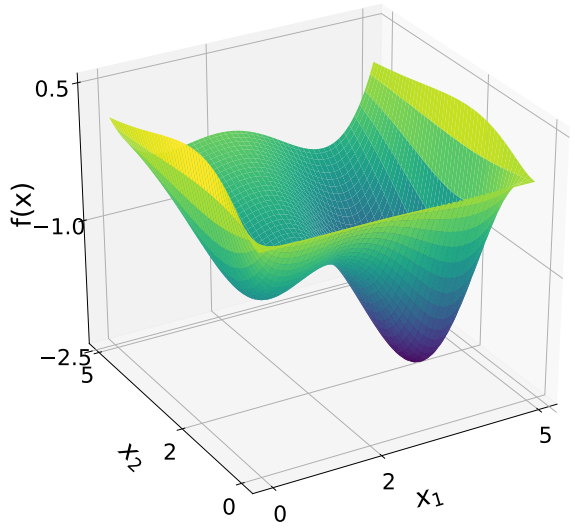


Figure A.9: Hosaki function profile

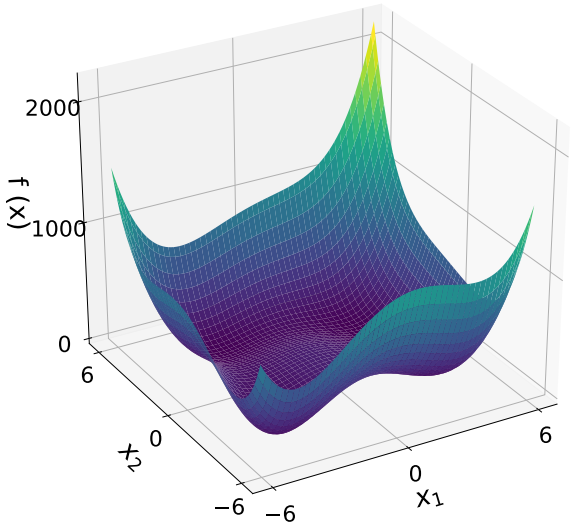


Figure A.10: Himmelblau function profile

A.11 Function with contrasting profiles

This is a one dimensional function with input defined on $x \in [0, 2]$, and exhibits multiple profiles over this interval. Figure A.11 clearly shows that the function profile is flat at smaller values, $x \in [0, 0.5]$ and exhibits oscillation of increasingly higher magnitudes over $x \in [0.5, 2]$. The mathematical expression of the function is given as,

$$f(x) = (6x-2)^2 \sin(12x-4). \tag{A.11}$$

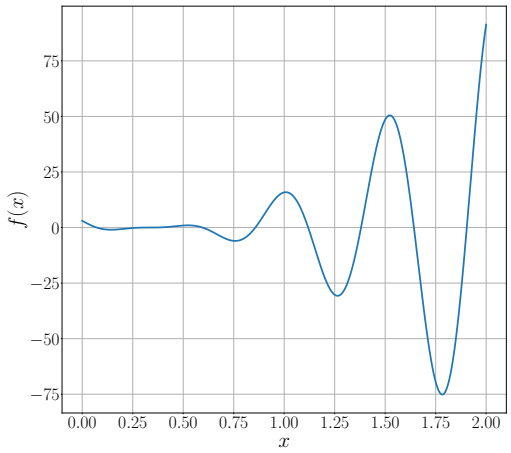


Figure A.11: Function with two distinct profiles

A.12 Constrained Function

This function is one-dimensional and has constraint in its profile. The functional output is non-negative, hence, cannot be less than 0. The function is evaluated with the input domain: $x \in [0, 1]$. The mathematical expression of the function is given in Equation A.12. The profile of the function is shown in Figure A.12.

$$f(x) = \frac{1}{1+(10x)^4} + \frac{1}{2}e^{-100\left(x-\frac{1}{2}\right)^2} \tag{A.12}$$

A.13 Symmetrically-constrained Function

This is a one dimensional algebraic function with a profile that is both symmetric and constrained. It is usually evaluated in the domain: $x \in [0, 1]$. The mathematical expression of the function is given in Equation A.13. The profile of the function is shown in Figure A.13.

$$f(x) = \frac{1}{100} + \frac{5}{8}(2x-1)^4 \left((2x-1)^2 + 4\sin^2 5\pi x\right) \tag{A.13}$$

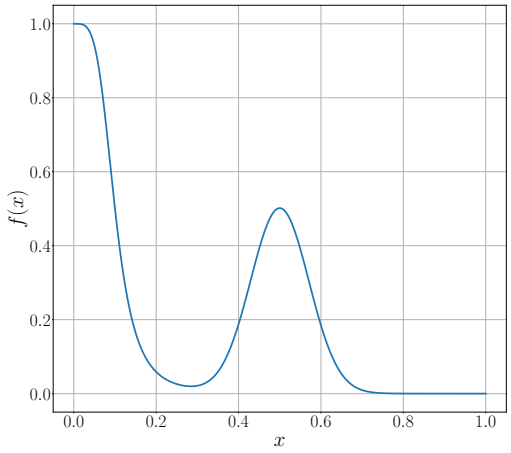


Figure A.12: Function with constrained profile

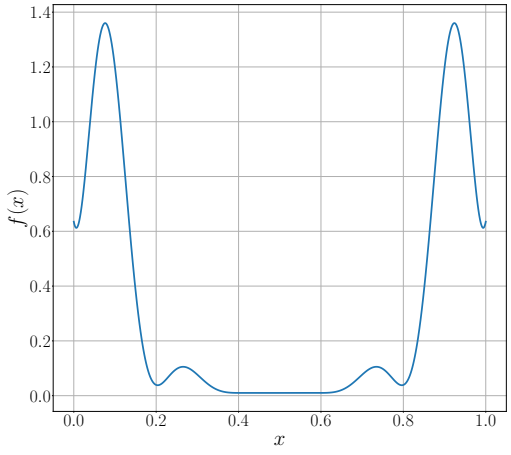


Figure A.13: Function with symmetrically-constrained profile

REFERENCES

[1] Cohn D A, Ghahramani Z, and Jordan M I. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[2] A B Abdessalem, N Dervilis, D J Wagg, and K Worden. Automatic kernel selection for gaussian processes regression with approximate bayesian computation and sequential monte carlo. *Front Built Environ*, 2017.

[3] Erdem Acar and Masoud Rais-Rohani. Ensemble of metamodels with optimized weight factors. *Structural and Multidisciplinary Optimization*, 37(3):279–294, 2009.

[4] A Afkanpour, C Szepesvári, and M Bowling. Alignment based kernel learning with a continuous set of base kernels. *Machine learning*, 91(3):305–324, 2013.

[5] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[6] T Agrawal. Optuna and automl. In *Hyperparameter Optimization in Machine Learning*, pages 109–129. Springer, 2021.

[7] MYM Ahmed and Ning Qin. Surrogate-based aerodynamic design optimization: Use of surrogates in aerodynamic design optimization. In *International Conference on Aerospace Sciences and Aviation Technology*, volume 13, pages 1–26. The Military Technical College, 2009.

[8] K Aho, D Derryberry, and T Peterson. Model selection for ecologists: the worldviews of aic and bic. *Ecology*, pages 631–636, 2014.

[9] T Akiba, S Sano, T Yanase, T Ohta, and M Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[10] N Alexandrov, E Nielsen, R Lewis, and W Anderson. First-order model management with variable-fidelity physics applied to multi-element airfoil optimization. In *8th Symposium on Multidisciplinary Analysis and Optimization*, page 4886, 2000.

[11] S Ali and K A Smith-Miles. A meta-learning approach to automatic kernel selection for support vector machines. *Neurocomputing*, pages 173–186, 2006.

[12] David M Allen. *The prediction sum of squares as a criterion for selecting predictor variables*. University of Kentucky, 1971.

[13] Hassane Allouche, Noura Ghanou, and Khalid Tigma. Detecting discontinuity points from spectral data with the quotient-difference (qd) algorithm. *Journal of Computational and Applied Mathematics*, 236(9):2406–2424, 2012.

[14] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W Hogg, and Michael O'Neil. Fast direct methods for gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):252–265, 2015.

[15] Virgil L Anderson and Robert A McLean. *Design of experiments: a realistic approach*. CRC Press, 1974.

[16] Ioannis Andrianakis, Ian R Vernon, Nicky McCreesh, Trevelyan J McKinley, Jeremy E Oakley, Rebecca N Nsubuga, Michael Goldstein, and Richard G White. Bayesian history matching of complex infectious disease models using emulation: a tutorial and a case study on hiv in uganda. *PLoS computational biology*, 11(1):e1003968, 2015.

[17] B Ankenman, B L Nelson, and J Staum. Stochastic kriging for simulation metamodeling. *Operations Research*, 58(2):371–382, 2010.

[18] Bruce Ankenman, Barry L Nelson, and Jeremy Staum. Stochastic kriging for simulation metamodeling. In *2008 Winter Simulation Conference*, pages 362–370. IEEE, 2008.

[19] András Antos, Varun Grover, and Csaba Szepesvári. Active learning in heteroscedastic noise. *Theoretical Computer Science*, 411(29-30):2712–2728, 2010.

[20] Athanasios C Antoulas. *Approximation of large-scale dynamical systems*. SIAM, 2005.

[21] Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

[22] F Bachoc. Cross validation and maximum likelihood estimation of hyper-parameters of gaussian processes with model misspecification. *Comput Stat Data Anal*, 66:55–69, 2013.

[23] Gregory A Banyay, Michael D Shields, and John C Brigham. Efficient global sensitivity analysis for flow-induced vibration of a nuclear reactor assembly using kriging surrogates. *Nuclear Engineering and Design*, 341:1–15, 2019.

[24] Mohamed Bennasar, Yulia Hicks, and Rossitza Setchi. Feature selection using joint mutual information maximisation. *Expert Systems with Applications*, 42(22):8520–8532, 2015.

[25] D Bettebghor, N Bartoli, S Grihon, J Morlier, and M Samuelides. Surrogate modeling approximation using a mixture of experts based on em joint estimation. *Structural and Multidisciplinary Optimization*, 43(2):243–259, 2011.

[26] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, New York, NY, softcover reprint of the original 1st edition 2006 (corrected at 8th printing 2009) edition, 2016.

[27] S Biswas, S Chakraborty, S Chandra, and I Ghosh. Kriging-based approach for estimation of vehicular speed and passenger car units on an urban arterial. *Journal of Transportation Engineering, Part A: Systems*, 143(3):04016013, 2017.

[28] Alexander Blaessle. `constNMPy`: A python package for constrained nelder-mead optimization, 2017.

[29] M A Bouhlel, N Bartoli, A Otsmane, and J Morlier. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, pages 935–952, 2016.

[30] M A Bouhlel, N Bartoli, A Otsmane, and J Morlier. Efficient global optimization for high dimensional constrained problems by using the kriging models combined with the partial least squares method. *Engineering Optimization*, 2018.

[31] M A Bouhlel and J R R A Martins. Gradient-enhanced kriging for high-dimensional problems, 2017.

[32] Hamparsum Bozdogan. Akaike's information criterion and recent developments in information complexity. *Journal of mathematical psychology*, 44(1):62–91, 2000.

[33] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[34] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[35] J M Buhmann. Simbad: Emergence of pattern similarity. In *Similarity-Based Pattern Analysis and Recognition*, pages 45–64. Springer London, 2013.

[36] M D Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics, 2003.

[37] C J C Burges. A tutorial on support vector machines for pattern recognition, 1998.

[38] K P Burnham and D R Anderson. Multimodel inference: understanding aic and bic in model selection. *Sociology Methods Research*, pages 261–304, 2004.

[39] Dizza Bursztyn and David M Steinberg. Comparison of designs for computer experiments. *Journal of statistical planning and inference*, 136(3):1103–1119, 2006.

[40] Damianou A C and Lawrence N D. Deep gaussian processes. In *Proceedings of the sixteenth international conference on artificial intelligence and statistics, AISTATS 2013*, pages 207–215, 2013.

[41] Wei Cai, Li Yang, and Yu Yu. Solution of ackley function based on particle swarm optimization algorithm. In *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pages 563–566. IEEE, 2020.

[42] Victoria CP Chen, Kwok-Leung Tsui, Russell R Barton, and Janet K Allen. A review of design and modeling in computer experiments. *Handbook in Statistics: Statistics in Industry*, 2002.

[43] X Chen, B Ankenman, and B L Nelson. Enhancing stochastic kriging metamodels with gradient estimators. *Operations Research*, 61(2):512–528, 2013.

[44] Jean-Paul Chilès and Nicolas Desassis. Fifty years of kriging. In *Handbook of mathematical geosciences*, pages 589–612. Springer, Cham, 2018.

[45] Tinkle Chugh, Alma Rahat, Vanessa Volz, and Martin Zaefferer. Towards better integration of surrogate models and optimizers. In *High-Performance Simulation-Based Optimization*, pages 137–163. Springer, 2020.

[46] H-S Chung and J Alonso. Using gradients to construct cokriging approximation models for high-dimensional design optimization problems. In *40th AIAA Aerospace Sciences Meeting & Exhibit*, page 317, 2002.

[47] Thomas M Cioppa and Thomas W Lucas. Efficient nearly orthogonal and space-filling latin hypercubes. *Technometrics*, 49(1):45–55, 2007.

[48] Joseph C Collins and III. Latin hypercube sampling in sensitivity analysis, 1994.

[49] C Cortes and V Vapnik. Support-vector networks. *Machine Learning*, pages 273–297, 1995.

[50] T M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, pages 326–334, 1965.

[51] Noel Cressie. The origins of kriging. 22:239–252, 1990.

[52] Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

[53] C Currin, T Mitchell, M Morris, and D Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

[54] Veronica Czitrom. One-factor-at-a-time versus designed experiments. *The American Statistician*, 53(2):126–131, 1999.

[55] Jouke de Baar, Stephen Roberts, Richard Dwight, and Benoit Mallol. Uncertainty quantification for a sailing yacht hull, using multi-fidelity kriging. *Computers & Fluids*, 123:185–201, 2015.

[56] Jouke HS de Baar, Richard P Dwight, and Hester Bijl. Improvements to gradient-enhanced kriging using a bayesian interpretation. *International Journal for Uncertainty Quantification*, 4(3), 2014.

[57] K Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, pages 311–338, 2000.

[58] Dirk Deschrijver, Karel Crombecq, Huu Minh Nguyen, and Tom Dhaene. Adaptive sampling algorithm for macromodeling of parameterized s-parameter responses. *IEEE Transactions on Microwave Theory and Techniques*, 59(1):39–45, 2010.

[59] P Domingos. A few useful things to know about machine learning. *Communications of the ACM*, pages 78–87, 2012.

[60] Vincent Dubourg, Bruno Sudret, and Franois Deheeger. Metamodel-based importance sampling for structural reliability analysis. *Probabilistic Engineering Mechanics*, 33:47–57, 2013.

[61] Olivier Dubrule. Cross validation of kriging in a unique neighborhood. *Journal of the International Association for Mathematical Geology*, 15(6):687–699, 1983.

[62] D Duvenaud. *Automatic model construction with Gaussian Processes*. PhD thesis, 2014.

[63] David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174. PMLR, 2013.

[64] Richard Dwight and Zhong-Hua Han. Efficient uncertainty quantification using gradient-enhanced kriging. In *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 17th AIAA/ASME/AHS Adaptive Structures Conference 11th AIAA No*, page 2276, 2009.

[65] J Eason and S Cremaschi. Adaptive sequential sampling for surrogate model generation with artificial neural networks. *Computers and Chemical Engineering*, 68:220–232, 2014.

[66] Michael Eldred and Daniel Dunlavy. Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models. In *11th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 7117, 2006.

[67] MS Eldred, H Agarwal, VM Perez, SF Wojtkiewicz Jr, and JE Renaud. Investigation of reliability method formulations in dakota/uq. *Structure and Infrastructure Engineering*, 3(3):199–213, 2007.

[68] Ugur Ergul and Gokhan Bilgin. Mck-elm: multiple composite kernel extreme learning machine for hyperspectral images. *Neural Computing and Applications*, pages 1–11, 2019.

[69] GM Fadel, MF Riley, and JM Barthelemy. Two point exponential approximation method for structural optimization. *Structural optimization*, 2(2):117–124, 1990.

[70] Jianqing Fan, Jinchi Lv, and Lei Qi. Sparse high-dimensional models in economics. *Annu. Rev. Econ.*, 3(1):291–317, 2011.

[71] Kai-Tai Fang, Runze Li, and Agus Sudjianto. *Design and modeling for computer experiments*. Chapman and Hall/CRC, 2005.

[72] Benjamin Fischer, Nico Gorbach, Stefan Bauer, Yatao Bian, and Joachim M Buhmann. Model selection for gaussian process regression by approximation set coding. *arXiv preprint arXiv:1610.00907*, 2016.

[73] Ronald Aylmer Fisher et al. The design of experiments. *The design of experiments.*, (2nd Ed), 1937.

[74] François Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine learning research*, 5(9), 2004.

[75] Alexander Forrester, András Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[76] Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

[77] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[78] Chongbo Fu, Peng Wang, Liang Zhao, and Xinjing Wang. A distance correlation-based kriging modeling method for high-dimensional problems. *Knowledge-Based Systems*, 206:106356, 2020.

[79] Jan N Fuhg, Amélie Fau, and Udo Nackenhorst. State-of-the-art and comparative review of adaptive sampling methods for kriging. *Archives of Computational Methods in Engineering*, 28(4):2689–2747, 2021.

[80] Kenji Fukumizu and Chenlei Leng. Gradient-based kernel dimension reduction for regression. *Journal of the American Statistical Association*, 109(505):359–370, 2014.

[81] Journel A G and Rossi M E. When do we need a trend model in kriging? *Mathematical Geology*, pages 715–739, 1989.

[82] Kronberger G and Kommenda M. Evolution of covariance functions for gaussian process regression using genetic programming, 2013.

[83] Shawn E Gano, John E Renaud, Jay D Martin, and Timothy W Simpson. Update strategies for kriging models used in variable fidelity optimization. *Structural and Multidisciplinary Optimization*, 32(4):287–298, jul 2006.

[84] Martha Gardner, Vinay Ramanath, Premnath Ayyalasomayajula, and Angela Patterson. From small x to large x: Assessment of space-filling criteria for the design and analysis of computer experiments. In *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 14th AIAA/ASME/AHS Adaptive Structures Conference 7th*, page 1714, 2006.

[85] Roman Garnett. Bayesian methods in machine learning, 2015.

[86] Sushant S Garud, Iftekhar A Karimi, and Markus Kraft. Design of computer experiments: A review. *Computers & Chemical Engineering*, 106:71–95, 2017.

[87] Sushant Suhas Garud, Iftekhar A Karimi, and Markus Kraft. Smart sampling algorithm for surrogate model development. *Computers & Chemical Engineering*, 96:103–114, 2017.

[88] Ruiquan Ge, Manli Zhou, Youxi Luo, Qinghan Meng, Guoqin Mai, Dongli Ma, Guoqing Wang, and Fengfeng Zhou. Mctwo: a two-step feature selection algorithm based on maximal information coefficient. *BMC bioinformatics*, 17(1):1–14, 2016.

[89] Yu Ge, Junjun Shi, Yaohui Li, and Jingfang Shen. An efficient kriging modeling method based on multidimensional scaling for high-dimensional problems. *Algorithms*, 15(1):3, 2021.

[90] KC Giannakoglou, DI Papadimitriou, and IC Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. *Computer methods in applied mechanics and engineering*, 195(44-47):6312–6329, 2006.

[91] David Ginsbourger, Céline Helbert, and Laurent Carraro. Discrete mixtures of kernels for kriging-based optimization. *Quality and Reliability Engineering International*, 24(6):681–691, 2008.

[92] M Giselle Fernández-Godino, Chanyoung Park, Nam H Kim, and Raphael T Haftka. Issues in deciding whether to use multifidelity surrogates. *AIAA Journal*, 57(5):2039–2054, 2019.

[93] Anthony Giunta, Steven Wojtkiewicz, and Michael Eldred. Overview of modern design of experiments methods for computational simulations. In *41st Aerospace Sciences Meeting and Exhibit*, page 649, 2003.

[94] José L Godoy, Jorge R Vega, and Jacinto L Marchetti. Relationships between pca and pls-regression. *Chemometrics and Intelligent Laboratory Systems*, 130:182–191, 2014.

[95] Tushar Goel, Raphael T Haftka, Wei Shyy, and Nestor V Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, sep 2006.

[96] Tushar Goel, Raphael T Haftka, Wei Shyy, and Nestor V Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, sep 2006.

[97] M Gönen and E Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[98] Mehmet Gönen and Adam A Margolin. Localized data fusion for kernel k-means clustering with application to cancer biology. *Advances in Neural Information Processing Systems*, 27:1305–1313, 2014.

[99] D Gong, C Wei, L Wang, L Feng, and L Wang. Adaptive methods for center choosing of radial basis function interpolation: A review. pages 573–580. Information Computing and Applications, 2010.

[100] R Gramacy. lagp: large-scale spatial modeling via local approximate gaussian processes in r. *Journal of Statistical Software*, 72(1):1–46, 2016.

[101] Robert B Gramacy. Surrogates: Gaussian process modeling, design, and optimization for the applied sciences. 2020.

[102] Eric James Grimme. *Krylov projection methods for model reduction*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[103] Serkan Gugercin and Athanasios C Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766, 2004.

[104] Braham H, Ben Jemaa S, Sayrac B, Fort G, and Moulines E. Low complexity spatial interpolation for cellular coverage analysis. In *12th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt) IEEE*, pages 188–195, 2014.

[105] J H Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. 2:84–90, 1960.

[106] Min Han, Weijie Ren, and Xiaoxin Liu. Joint mutual information-based input variable selection for multivariate time series modeling. *Engineering Applications of Artificial Intelligence*, 37:250–257, 2015.

[107] Zhong-Hua Han, Yu Zhang, Chen-Xing Song, and Ke-Shi Zhang. Weighted gradient-enhanced kriging for high-dimensional surrogate modeling and design optimization. *Aiaa Journal*, 55(12):4330–4346, 2017.

[108] H Hang and I Steinwart. Optimal learning with anisotropic gaussian svms, 2018.

[109] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[110] T Hofmann, B Scholkopf, and A J Smola. A review of kernel methods in machine learning, 2006.

[111] Scott A Hollingsworth and Ron O Dror. Molecular dynamics simulation for all. *Neuron*, 99(6):1129–1143, 2018.

[112] Daniel Homola. Mifs: Parallelized mutual information based feature selection module. `https://https://github.com/danielhomola/mifs`, 2016.

[113] JC Hoskins and DM Himmelblau. Process control via artificial neural networks and reinforcement learning. *Computers & chemical engineering*, 16(4):241–251, 1992.

[114] Deng Huang, Theodore T Allen, William I Notz, and Ning Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 34(3):441–466, 2006.

[115] John T Hwang and Joaquim RRA Martins. A fast-prediction surrogate model for large datasets. *Aerospace Science and Technology*, 75:74–87, 2018.

[116] Hensman J, Fusi N, , and Lawrence N D. Gaussian processes for big data. In *Proceedings of the twenty-ninth international conference on uncertainty in artificial intelligence, Bellevue*, 2013.

[117] R A Jacobs, M I Jordan, S J Nowlan, and G E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[118] Jan Dirk Jansen and Louis J Durlofsky. Use of reduced-order models in well control optimization. *Optimization and Engineering*, 18(1):105–132, 2017.

[119] Shinkyu Jeong, Mitsuhiro Murayama, and Kazuomi Yamamoto. Efficient optimization design method using kriging model. *Journal of aircraft*, 42(2):413–420, 2005.

[120] Jing Jiang. Information extraction from text. In *Mining text data*, pages 11–41. Springer, 2012.

[121] Ping Jiang, Qi Zhou, and Xinyu Shao. *Surrogate model-based engineering design and optimization*. Springer, 2020.

[122] Ruichen Jin, Wei Chen, and Timothy W Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and multidisciplinary optimization*, 23(1):1–13, 2001.

[123] Iain M Johnstone and D Michael Titterington. Statistical challenges of high-dimensional data, 2009.

[124] Donald R Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[125] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[126] V Roshan Joseph, Ying Hung, and Agus Sudjianto. Blind kriging: A new method for developing metamodels. 2008.

[127] M Kanagawa, P Hennig, D Sejdinovic, and B K Sriperumbudur. Gaussian processes and kernel methods:a review on connections and equivalences, 2018.

[128] Andy Keane and Prasanth Nair. *Computational approaches for aerospace design: the pursuit of excellence*. John Wiley & Sons, 2005.

[129] Pierric Kersaudy, Bruno Sudret, Nadège Varsier, Odile Picon, and Joe Wiart. A new surrogate modeling technique combining kriging and polynomial chaos expansions–application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics*, 286:103–117, 2015.

[130] Jack PC Kleijnen. Kriging metamodeling in simulation: A review. *European journal of operational research*, 192(3):707–716, 2009.

[131] Jack PC Kleijnen, Susan M Sanchez, Thomas W Lucas, and Thomas M Cioppa. State-of-the-art review: a user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3):263–289, 2005.

[132] JR Koehler and AB Owen. Computer experiments. handbook of statistics. *Elsevier Science*, pages 261–308, 1996.

[133] D Kraft. A software package for sequential quadratic programming, 1988.

[134] D Kraft. Algorithm 733: Tomp-fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.

[135] Jan Kremer, Kim Steenstrup Pedersen, and Christian Igel. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4):313–326, 2014.

[136] Chen Quin Lam. *Sequential adaptive designs in computer experiments for response surface model fit*. PhD thesis, The Ohio State University, 2008.

[137] XB Lam, YS Kim, AD Hoang, and CW Park. Coupled aerostructural design optimization using the kriging model and integrated multiobjective optimization algorithm. *Journal of Optimization Theory and Applications*, 142(3):533–556, 2009.

[138] Luc Laurent, Rodolphe Le Riche, Bruno Soulier, and Pierre-Alain Boucard. An overview of gradient-enhanced metamodels with applications. *Archives of Computational Methods in Engineering*, 26(1):61–106, 2019.

[139] Spiro Lekoudis. Computational fluid dynamics - navy perspective, 1993.

[140] Robert Lewis and Stephen Nash. A multigrid approach to the optimization of systems governed by differential equations. In *8th symposium on multidisciplinary analysis and optimization*, page 4890, 2000.

[141] Runze Li, Wei Zhong, and Liping Zhu. Feature screening via distance correlation learning. *Journal of the American Statistical Association*, 107(499):1129–1139, 2012.

[142] Min Liang and Denis Marcotte. A class of non-stationary covariance functions with compact support. *Stochastic environmental research and risk assessment*, 30(3):973–987, 2016.

[143] R P Liem, C A Mader, and E Lee. Aerostructural design optimization of a 100-passenger regional jet with surrogate-based mission analysis. In *2013 Aviation technology, intergration, and operations conference*, 2013.

[144] R P Liem, C A Mader, and J R R A Martins. Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis. *Aerospace Science and Technology*, 43:126–151, 2015.

[145] Rhea Patricia Liem. *Multimission Fuel-Burn Minimization in Aircraft Design: A Surrogate-Modeling Approach*. PhD thesis, University of Toronto (Canada), 2015.

[146] Rhea Patricia Liem, Kehinde Sikirulai Oyetunde, Pramudita Satria Palar, and Koji Shimoyama. Kriging with mixed kernel (mk) for complex aerospace problems. In *Sixteenth International Conference on Flow Dynamics*, 2019.

[147] H Liu, J Cai, and Y Ong. An adaptive sampling approach for kriging metamodeling by maximizing expected prediction error. *Computers and Chemical Engineering*, 106:171–182, 2017.

[148] Haitao Liu, Shengli Xu, Ying Ma, Xudong Chen, and Xiaofang Wang. An adaptive bayesian sequential sampling approach for global metamodeling. *Journal of Mechanical Design*, 138(1):011404, 2016.

[149] Jun Liu, Zhonghua Han, and Wenping Song. Comparison of infill sampling criteria in kriging-based aerodynamic optimization. In *28th congress of the international council of the aeronautical sciences*, pages 23–28, 2012.

[150] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE transactions on visualization and computer graphics*, 23(3):1249–1268, 2016.

[151] X Liu, L Wang, X Zhu, M Li, E Zhu, T Liu, L Liu, Y Dou, and J Yin. Absent multiple kernel learning algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1303–1316, 2019.

[152] J R Lloyd, D Duvenaud, R James R, Grosse, J Tenenbaum, and Z Ghahramani. Automatic construction and natural-language description of non-parametric regression models. 2014.

[153] Brian A Lockwood and Mihai Anitescu. Gradient-enhanced universal kriging for uncertainty propagation. *Nuclear Science and Engineering*, 170(2):168–195, 2012.

[154] Søren Nymand Lophaven, Hans Bruun Nielsen, and Jacob Søndergaard. Dace-a matlab kriging toolbox, version 2.0. 2002.

[155] Thomas Lorenzen and Virgil Anderson. *Design of experiments: a no-name approach*. CRC Press, 1993.

[156] Alberto Lovison and Enrico Rigoni. Adaptive sampling with a lipschitz criterion for accurate metamodeling. *Communications in Applied and Industrial Mathematics*, 1(2):110–126, 2011.

[157] David J Lucia, Philip S Beran, and Walter A Silva. Reduced-order modeling: new approaches for computational physics. *Progress in aerospace sciences*, 40(1-2):51–117, 2004.

[158] Marco Antonio Luersen, Rodolphe Le Riche, and Frédéric Guyon. A constrained, globalized, and bounded nelder–mead method for engineering optimization. *Structural and Multidisciplinary Optimization*, 27(1):43–54, 2004.

[159] James F Lynch. Analysis and application of adaptive sampling. *Journal of Computer and System Sciences*, 66(1):2–19, 2003.

[160] Y Lyu and R P Liem. Flight performance analysis with data-driven mission parameterization: mapping flight operational data to aircraft performance analysis. *Transportation Engineering*, 2020.

[161] Yolanda Mack, Tushar Goel, Wei Shyy, and Raphael Haftka. Surrogate model-based optimization framework: a case study in aerospace design. In *Evolutionary computation in dynamic and uncertain environments*, pages 323–342. Springer, 2007.

[162] Zuzana Majdisova and Vaclav Skala. Radial basis function approximations: comparison and applications. *Applied Mathematical Modelling*, 51:728–743, 2017.

[163] Y Mao, H Cao, P Ping, and X Li. Feature selection based on maximum conditional and joint mutual information. *Journal of Computer Applications*, 39(3):734, 2019.

[164] Jay D Martin and Timothy W Simpson. Use of kriging models to approximate deterministic computer models. *AIAA journal*, 43(4):853–863, 2005.

[165] Joaquim RRA Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[166] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

[167] Kevin McBride and Kai Sundmacher. Overview of surrogate modeling in chemical process engineering. *Chemie Ingenieur Technik*, 91(3):228–239, 2019.

[168] Dale McDonald, Walter Grantham, Wayne Tabor, and Michael Murphy. Response surface model development for global/local optimization using radial basis functions. In *8th Symposium on Multidisciplinary Analysis and Optimization*, page 4776, 2000.

[169] M. Meckesheimer, A. J. Booker, R. R. Barton, and T. W. Simpson. Computationally inexpensive metamodel assessment strategies. *AIAA Journal*, 40(10), 2002.

[170] E Meeds and S Osindero. An alternative infinite mixture of gaussian process experts. *Advances in Neural Information Processing Systems*, 18:883–890, 2006.

[171] Ali Mehmani, Souma Chowdhury, and Achille Messac. Predictive quantification of surrogate model fidelity based on modal variations with sample density. *Structural and Multidisciplinary Optimization*, 52(2):353–373, 2015.

[172] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat no 98th8468)*, pages 41–48. IEEE, 1999.

[173] Hossein Mohammadi. *Kriging-based black-box global optimization: analysis and new algorithms*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2016.

[174] Douglas C Montgomery. *Design and analysis of experiments*. John wiley & sons, 2017.

[175] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[176] G Moreaux. Compactly supported radial covariance functions. *Journal of Geodesy*, 82(7):431–443, 2008.

[177] M D Morris, T J Mitchell, and D Ylvisaker. Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics*, pages 243–255, 1993.

[178] Amir Mosavi and Atieh Vaezipour. Reactive search optimization; application to multiobjective optimization problems. *Applied Mathematics*, 3(10A):1572–1582, 2012.

[179] Irene Moser. Hooke-jeeves revisited. In *2009 IEEE Congress on Evolutionary Computation*, pages 2670–2676. IEEE, 2009.

[180] Saeed Motiian and Hamid Soltanian-Zadeh. Improved particle swarm optimization and applications to hidden markov model and ackley function. In *2011 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings*, pages 1–4. IEEE, 2011.

[181] Maliki Moustapha, Jean-Marc Bourinet, Benoît Guillaume, and Bruno Sudret. Comparative study of kriging and support vector regression for structural engineering applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 4(2), 2018.

[182] Tanmoy Mukhopadhyay, S Chakraborty, S Dey, S Adhikari, and Rajib Chowdhury. A critical assessment of kriging model variants for high-fidelity uncertainty quantification in dynamics of composite shells. *Archives of Computational Methods in Engineering*, 24(3):495–518, 2017.

[183] Evan Murray, Jae Hun Cho, Daniel Goodwin, Taeyun Ku, Justin Swaney, Sung-Yon Kim, Heejin Choi, Young-Gyun Park, Jeong-Yoon Park, Austin Hubbert, et al. Simple, scalable proteomic imaging for high-dimensional profiling of intact systems. *Cell*, 163(6):1500–1514, 2015.

[184] Durrande N, Ginsbourger D, and Roustant O. Additive covariance kernels for high-dimensional gaussian process modeling. *Ann Fac Sci Toulouse Math*, 21(3):481–499, 2012.

[185] L Nechak, F Gillot, S Besset, and J-J Sinou. Sensitivity analysis and kriging based models for robust stability analysis of brake systems. *Mechanics Research Communications*, 69:136–145, 2015.

[186] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[187] Oliver Nelles. Nonlinear dynamic system identification. In *Nonlinear System Identification*, pages 547–577. Springer, 2001.

[188] Huu Minh Nguyen, Ivo Couckuyt, Luc Knockaert, Tom Dhaene, Dirk Gorissen, and Yvan Saeys. An alternative approach to avoid overfitting for surrogate models. In *Proceedings of the 2011 winter simulation conference (WSC)*, pages 2760–2771. IEEE, 2011.

[189] Jeremy Oakley. Estimating percentiles of uncertain computer code outputs. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 53(1):83–93, 2004.

[190] Kehinde Sikirulai Oyetunde and Rhea P Liem. Navigating kernel selections in kernel-based methods: The issues and possible solutions. In *AIAA SCITECH 2022 Forum*, page 0507, 2022.

[191] P S Palar, L R Zuhal, and K Shimoyama. Gaussian process surrogate model with composite kernel learning for engineering design. *AIAA Journal*, 2020.

[192] Pramudita S Palar and Koji Shimoyama. Multi-fidelity uncertainty analysis in cfd using hierarchical kriging. In *35th AIAA applied aerodynamics conference*, page 3261, 2017.

[193] Pramudita S Palar and Koji Shimoyama. Kriging with composite kernel learning for surrogate modeling in computer experiments. In *AIAA Scitech 2019 Forum*, page 2209, 2019.

[194] Pramudita S Palar, Kemas Zakaria, Lavi Rizki Zuhal, Koji Shimoyama, and Rhea P Liem. Gaussian processes and support vector regression for uncertainty quantification in aerodynamics. In *AIAA Scitech 2021 Forum*, page 0181, 2021.

[195] Pramudita Satria Palar, Rhea Patricia Liem, Lavi Rizki Zuhal, and Koji Shimoyama. On the use of surrogate models in engineering design optimization and exploration: The key issues. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1592–1602, 2019.

[196] Pramudita Satria Palar and Koji Shimoyama. Ensemble of kriging with multiple kernel functions for engineering design optimization. In *International Conference on Bioinspired Methods and Their Applications*, pages 211–222. Springer, 2018.

[197] Pramudita Satria Palar and Koji Shimoyama. Efficient global optimization with ensemble and selection of kernel functions for engineering design. *Structural and Multidisciplinary Optimization*, 59(1):93–116, 2019.

[198] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[199] Fernando Pérez-Cruz and Olivier Bousquet. Kernel methods and their potential use in signal processing. *IEEE Signal Processing Magazine*, 21(3):57–65, 2004.

[200] Victor Picheny, Tobias Wagner, and David Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and multidisciplinary optimization*, 48(3):607–626, 2013.

[201] K E Pilario, M Shafiee, Y Cao, and S-H Yang. A review of kernel methods for feature extraction in nonlinear process monitoring. *Processes*, 2020.

[202] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[203] Luc Pronzato and Werner G Müller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, 2012.

[204] Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

[205] Milos Radovic, Mohamed Ghalwash, Nenad Filipovic, and Zoran Obradovic. Minimum redundancy maximum relevance feature selection approach for temporal gene expression data. *BMC bioinformatics*, 18(1):1–14, 2017.

[206] Dennis C Rapaport and Dennis C Rapaport Rapaport. *The art of molecular dynamics simulation*. Cambridge university press, 2004.

[207] C E Rasmussen and Z Ghahramani. Infinite mixtures of gaussian process experts. *Advances in Neural Information Processing Systems*, 2002.

[208] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. 2006.

[209] Sami Remes, Markus Heinonen, and Samuel Kaski. Non-stationary spectral kernels. *Advances in neural information processing systems*, 30, 2017.

[210] Theresa Robinson, Michael Eldred, Karen Willcox, and Robert Haimes. Strategies for multifidelity optimization with variable dimensional hierarchical models. In *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 14th AIAA/ASME/AHS Adaptive Structures Conference 7th*, page 1819, 2006.

[211] Theresa Dawn Robinson. *Surrogate-based optimization using multifidelity models with variable parameterization*. PhD thesis, Massachusetts Institute of Technology, 2007.

[212] Christopher Roy. Review of discretization error estimators in scientific computing. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 126, 2010.

[213] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. 4:409–423, 1989.

[214] S Sakata, F Ashida, and M Zako. Structural optimization using kriging approximation. *Computer methods in applied mechanics and engineering*, 192(7-8):923–939, 2003.

[215] S Sakata, F Ashida, and M Zako. An efficient algorithm for kriging approximations and optimization with large-scale sampling data. *Computer Methods in Applied Mechanics and Engineering*, 193:385–404, 2004.

[216] R Salakhutdinov and G E Hinton. Using deep belief nets to learn covariance kernels for gaussian processes. In *NIPS*, volume 7, pages 1249–1256. Citeseer, 2007.

[217] M B Salem and L Tomaso. Automatic selection for general surrogate models. *Structural and Multidisciplinary Optimization*, 2018.

[218] Thomas J Santner, Brian J Williams, and William I Notz. *The design and analysis of computer experiments*, volume 1. Springer, 2003.

[219] Roland Schobi, Bruno Sudret, and Joe Wiart. Polynomial-chaos-based kriging. *International Journal for Uncertainty Quantification*, 5(2), 2015.

[220] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[221] Andreas Sedlmeier, Michael Kölle, Robert Müller, Leo Baudrexel, and Claudia Linnhoff-Popien. Quantifying multimodality in world models. *arXiv preprint arXiv:2112.07263*, 2021.

[222] Larry J Segerlind. *Applied finite element analysis*. John Wiley & Sons, 1991.

[223] Burr Settles. Active learning literature survey. 2009.

[224] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[225] Yun-Wei Shang and Yu-Huang Qiu. A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126, 2006.

[226] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[227] F Simpson, I Davies, V Lalchand, A Vullo, N Durrande, and C Rasmussen. Kernel identification through transformers. *arXiv preprint arXiv:2106 08185*, 2021.

[228] T W Simpson, T M Mauery, J J Korte, and F Mistree. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA journal*, 39(12):2233–2241, 2001.

[229] Timothy Simpson, Vasilli Toropov, Vladimir Balabanov, and Felipe Viana. Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come-or not. In *12th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 5802, 2008.

[230] Timothy W Simpson, Andrew J Booker, Dipankar Ghosh, Anthony A Giunta, Patrick N Koch, and R-J Yang. Approximation methods in multidisciplinary analysis and optimization: a panel discussion. *Structural and multidisciplinary optimization*, 27(5):302–313, 2004.

[231] Timothy W Simpson, Dennis KJ Lin, and Wei Chen. Sampling strategies for computer experiments: design and analysis. *International Journal of Reliability and applications*, 2(3):209–240, 2001.

[232] A Smola and B Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, pages 199–222, 2004.

[233] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

[234] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[235] Jaroslaw Sobieszczanski-Sobieski and Raphael T Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23, 1997.

[236] Ilya M Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, 1976.

[237] Ilya Meerovich Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[238] Xueguan Song, Liye Lv, Wei Sun, and Jie Zhang. A radial basis function-based multi-fidelity surrogate model: exploring correlation between high-fidelity and low-fidelity models. *Structural and Multidisciplinary Optimization*, 60(3):965–981, 2019.

[239] Danny C Sorensen and AC Antoulas. The sylvester equation and approximate balanced reduction. *Linear algebra and its applications*, 351:671–700, 2002.

[240] Jeremy Staum. Better simulation metamodeling: The why, what, and how of stochastic kriging. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 119–133. IEEE, 2009.

[241] M L Stein. Interpolation of spatial data: Some theory for kriging. *Springer Science and Business Media*, page 176, 1999.

[242] Anna Stief, Ruomu Tan, Yi Cao, James R Ottewill, Nina F Thornhill, and Jerzy Baranowski. A heterogeneous benchmark dataset for data analytics: Multiphase flow facility case study. *Journal of Process Control*, 79:41–55, 2019.

[243] Stephen M Stigler. Gergonne's 1815 paper on the design and analysis of polynomial regression experiments. *Historia Mathematica*, 1(4):431–439, 1974.

[244] Barna Szabó and Ivo Babuška. Finite element analysis: Method, verification and validation. 2021.

[245] Collet T and Pietquin O. Optimism in active learning with gaussian processes. In *International Conference on Neural Information Processing*, pages 152–160, 2015.

[246] Praveen Thokala and Joaquim RRA Martins. Variable-complexity optimization applied to airfoil design. *Engineering optimization*, 39(3):271–286, 2007.

[247] Rohit Tripathy, Ilias Bilionis, and Marcial Gonzalez. Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *Journal of Computational Physics*, 321:191–223, 2016.

[248] Jiyuan Tu, Guan Heng Yeoh, and Chaoqun Liu. *Computational fluid dynamics: a practical approach*. Butterworth-Heinemann, 2018.

[249] Selvakumar Ulaganathan, Ivo Couckuyt, Tom Dhaene, Joris Degroote, and Eric Laermans. Performance study of gradient-enhanced kriging. *Engineering with computers*, 32(1):15–34, 2016.

[250] V Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[251] V Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[252] Felipe AC Viana, Raphael T Haftka, and Valder Steffen. Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. *Structural and Multidisciplinary Optimization*, 39(4):439–457, 2009.

[253] Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *ICCV*, volume 3, page 281, 2003.

[254] V Esposito Vinzi, Wynne W Chin, Jörg Henseler, Huiwen Wang, et al. *Handbook of partial least squares*, volume 201. Springer, 2010.

[255] Eric-Jan Wagenmakers and Simon Farrell. AIC model selection using Akaike weights. *Psychonomic bulletin & review*, 11(1):192–196, 2004.

[256] G Gary Wang and Songqing Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4), 2007.

[257] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Bäck. Time complexity reduction in efficient global optimization using cluster kriging. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 889–896, 2017.

[258] Liping Wang and Ramana V Grandhi. Improved two-point function approximations for design optimization. *AIAA journal*, 33(9):1720–1727, 1995.

[259] Mengmeng Wang, Guojin He, Zhaoming Zhang, Guizhou Wang, Zhengjia Zhang, Xiaojie Cao, Zhijie Wu, and Xiuguo Liu. Comparison of spatial interpolation and regression analysis models for an estimation of monthly near surface air temperature in china. *Remote Sensing*, 9(12):1278, 2017.

[260] Wei Wang, Lei Chen, and Qian Zhang. Outsourcing high-dimensional healthcare data to cloud with personalized privacy preservation. *Computer Networks*, 88:136–148, 2015.

[261] Wenting Wang, Veerabhadran Baladandayuthapani, Jeffrey S Morris, Bradley M Broom, Ganiraju Manyam, and Kim-Anh Do. ibag: integrative bayesian analysis of high-dimensional multiplatform genomics data. *Bioinformatics*, 29(2):149–159, 2013.

[262] Ganesha Weerasinghe, Hongmei Chi, and Yanzhao Cao. Particle swarm optimization simulation via optimal halton sequences. *Procedia Computer Science*, 80:772–781, 2016.

[263] Karen E Willcox, Omar Ghattas, and Patrick Heimbach. The imperative of physics-based modeling and inverse theory in computational science. *Nature Computational Science*, 1(3):166–168, 2021.

[264] A G Wilson and R P Adams. Gaussian process kernels for pattern discovery and extrapolation, 2013.

[265] David Wipf and Srikantan Nagarajan. A new view of automatic relevance determination. *Advances in neural information processing systems*, 20, 2007.

[266] Xin Yan and Xiaogang Su. *Linear regression analysis: theory and computing*. world scientific, 2009.

[267] Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: A primer. *Neuron*, 107(6):1048–1070, 2020.

[268] H Yang and John Moody. Feature selection based on joint mutual information. In *Proceedings of international ICSC symposium on advances in intelligent data analysis*, volume 1999, pages 22–25. Citeseer, 1999.

[269] Xiu Yang, Guzel Tartakovsky, and Alexandre Tartakovsky. Physics-informed kriging: A physics-informed gaussian process regression method for data-model convergence. *arXiv preprint arXiv:1809.03461*, 2018.

[270] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[271] Hanfeng Yin, Hongbing Fang, Guilin Wen, Matthew Gutowski, and Youye Xiao. On the ensemble of metamodels with multiple regional optimized weight factors. *Structural and Multidisciplinary Optimization*, 58(1):245–263, 2018.

[272] Raul Yondo, Esther Andrés, and Eusebio Valero. A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Progress in aerospace sciences*, 96:23–61, 2018.

[273] T Yu and H Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003 05689*, 2020.

[274] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[275] Martin Zaefferer, Jörg Stork, Martina Friese, Andreas Fischbach, Boris Naujoks, and Thomas Bartz-Beielstein. Efficient global optimization for combinatorial problems. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pages 871–878, 2014.

[276] Z Zhai, H Li, and X Wang. An adaptive sampling method for kriging surrogate model with multiple outputs. *Engineering with Computers*, 2020.

[277] J Zhang, S Chowdhury, and J Zhang. Adaptive hybrid surrogate modeling for complex systems. *AIAA Journal*, 51(3):643–656, 2013.

[278] Jie Zhang, Souma Chowdhury, and Achille Messac. An adaptive hybrid surrogate model. *Structural and Multidisciplinary Optimization*, 46(2):223–238, 2012.

[279] Liang Zhao, Peng Wang, Baowei Song, Xinjing Wang, and Huachao Dong. An efficient kriging modeling method for high-dimensional design problems based on maximal information coefficient. *Structural and Multidisciplinary Optimization*, 61(1):39–57, 2020.

[280] D Zimmerman, C Pavlik, A Ruggles, and M P Armstrong. An experimental comparison of ordinary and universal kriging and inverse distance weighting. *Mathematical Geology*, pages 375–390, 1999.

[281] Lu Zou and Xiaowei Zhang. Stochastic kriging for inadequate simulation models. *arXiv preprint arXiv:1802.00677*, 2018.

[282] L R Zuhal, G A Faza, P S Palar, and R P Liem. Fast and adaptive reliability analysis via kriging and partial least squares. In *AIAA SciTech Forum*, 2021.

[283] L R Zuhal, K Zakaria, P S Palar, K Shimoyama, and R P Liem. Gradient-enhanced universal kriging with polynomial chaos as trend function. In *AIAA SciTech Forum*, 2020.

[284] Lavi Rizki Zuhal, Pramudita Satria Palar, and Koji Shimoyama. A comparative study of multi-objective expected improvement for aerodynamic design. *Aerospace Science and Technology*, 91:548–560, 2019.