Laboratório Nacional de Computação Científica

Programa de Pós-Graduação em Modelagem Computacional

Gaussian Process Modeling with Applications to Tumor Growth

João Vitor de Oliveira Silva

Petrópolis, RJ - Brasil

July, 2021

João Vitor de Oliveira Silva

Gaussian Process Modeling with Applications to Tumor Growth

Dissertation submitted to the examining committee in partial fulfillment of the requirements for the degree of Master of Sciences in Computational Modeling.

Laboratório Nacional de Computação Científica

Programa de Pós-Graduação em Modelagem Computacional

Advisor(s): Regina Célia Cerqueira de Almeida e Renato Simões Silva

Petrópolis, RJ - Brasil

July, 2021

Ficha catalográfica elaborada por Patrícia Vieira Silva - CRB7 5822

S586g Silva, João Vitor de Oliveira.
Gaussian process modeling with applications to tumor growth / João Vitor de Oliveira Silva. - Petrópolis, RJ: Laboratório Nacional de Computação Científica, 2021.
108 f.: il.; 30 cm.

Dissertação (Mestrado em Modelagem Computacional) - Laboratório Nacional de Computação Científica, 2021.
Orientadores: Regina Célia Cerqueira de Almeida e Renato Simões Silva.

1. Câncer - Modelos matemáticos. 2. Modelagem mecanística. 3. Modelagem baseada em dados. 4. Teoria bayesiana de decisão estatística. 5. Processos Gaussianos. I. Almeida, Regina Célia Cerqueira de. II. Silva, Renato Simões. III. LNCC/MCTI. IV. Título.

CDD - 616.994

Laboratório Nacional de Computação Científica

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

JOÃO VITOR DE OLIVEIRA SILVA

GAUSSIAN PROCESS MODELING WITH APPLICATIONS TO TUMOR GROWTH

Dissertação submetida ao corpo docente do Laboratório Nacional de Computação Científica como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências em Modelagem Computacional.

Aprovada por:

Profª. Regina Célia Cerqueira de Almeida, D.Sc.
(presidente)

Prof. Antônio Tadeu Azevedo Gomes, D.Sc.

Prof. Pablo Javier Blanco, D.Sc.

Prof. Alvaro Luiz Gayoso de Azeredo Coutinho, D.Sc.

Prof. Helcio Rangel Barreto Orlande, D.Sc.

Documento assinado eletronicamente por **Pablo Javier Blanco**, **Pesquisador Associado**, em 08/09/2021, às 17:03 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Regina Célia Cerqueira de Almeida**, **Pesquisador Titular**, em 08/09/2021, às 16:58 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Antônio Tadeu Azevedo Gomes**, **Tecnologista em Ciência e Tecnologia**, em 08/09/2021, às 17:15 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alvaro Luiz Gayoso de azeredo coutinho (E)**, **Usuário Externo**, em 09/09/2021, às 10:52 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **HELCIO RANGEL BARRETO ORLANDE (E)**, **Usuário Externo**, em 09/09/2021, às 11:22 (horário oficial de Brasília), com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site http://sei.mctic.gov.br/verifica.html, informando o código verificador **8088735** e o código CRC **6F790BEC**.

Referência: Processo nº 01209.000098/2020-83 SEI nº 8088735

Acknowledgements

Firstly, I would like to thank God for the strength and determination that made me able to persist in my studies during my whole life.

I would like to thank my advisors Regina Célia and Renato Silva, for their incredible guidance during these graduate years. Besides helping me during the course of this work, they also encourage me to pursue new challenges in scientific research. I would also like to express my gratitude for their patience, understanding, and empathy with the difficulties that I faced in this pandemic moment in which we find ourselves.

I would like to thank my colleagues and friends from the TMG group for their support. Heber, who immensely helped in the discussions and results of this particular work. Gustavo and Emanuelle, for the great partnership in other works. Anna, for helping me with computational tools and advises in the creation of schemes and diagrams in this work.

I would like to thank the other friends I made over these last years at LNCC. Aline, for being a great and caring housemate during the year I stayed in Petrópolis. The EAMC staff (Ana, Luis, Gustavo, Alonso and Matheus), for the great experience of organizing an academic event. The RPG crew (Italo, Edlaine, Weslley, Haron and Lucas), for all the fun we shared during the sessions.

I would like to thank my professors and now colleagues at the Institute of Computer Science at UFRJ for their support in conciliating my position as a substitute professor and the present work, especially Juliana, João Paixão, and João Carlos. Also, I would like to thank the friends I made in this same institution over the years who still help me to this date, including Thais, André, and Cintia.

I would like to thank my whole family for the unconditional support and love in my whole life. My mother Eunice, my father Luiz, and my brother Pedro. Without you, I would not have been able to accomplish so many things in my life.

Lastly, I would like to thank professor Artur Ziviani. I can still remember the conversation we had in the first day I visited LNCC. Me and everyone else in LNCC miss you dearly.

"All models are wrong, but some are useful."
(George E. P. Box)

Abstract

Mechanistic models have been used widely as important tools for understanding and development in many areas of science and engineering. In particular, tumor growth models have provided a better understanding of how the disease evolves and helped in the development of therapies. This process incurs in the solution of an inverse problem. There are several methods dedicated to solving such problems, one of these is Bayesian inference. The difficulty of performing Bayesian inference or other methods is that, in general, they require repeated model evaluation. This may be computationally prohibitive for complex models, specially multiscale models. There are approximate Bayesian inference techniques, which reduce the overall computational time, although these may still be infeasible if the model is expensive to be evaluated. In order to alleviate the difficulty in this procedure, we propose the use of a surrogate (or metamodel). Our surrogate model is a Gaussian Process, a data-driven model recently used in machine learning and metamodeling context. This work reviews theoretical and practical aspects of Gaussian processes (GP) in a regression problem, discussing how to construct an adequate GP for a particular problem, covering recent developments in this area. We then define a GP surrogate model and combine it with an Approximate Bayesian Computation Markov-Chain Monte Carlo method, the ABC-MCMC method, for solving the inverse problem. We compare the standard and proposed approaches in two tumor growth models. Our results suggest that the proposed strategy is promising in terms of reducing the computational cost.

Keywords: Mechanistic modeling, Data-driven modeling, Gaussian process, Approximate Bayesian computation, Surrogate model, Cancer modeling

Resumo

Modelos mecanicistas têm sido largamente usados como importantes ferramentas para o entendimento e desenvolvimento em muitas áreas da ciência e engenharia. Em particular, modelos do crescimento tumoral têm provido um melhor entendimento sobre como a doença evolui e ajudado no desenvolvimento de terapias. Para uso prático, é fundamental que os paramêtros dos modelos possam ser estimados a partir dos dados experimentais disponíveis. Esse processo acarreta a solução de um problema inverso. Existem diversos métodos dedicados a solução destes problemas, um deles sendo a inferência Bayesiana. A dificuldade de realizar inferência Bayesiana ou outros métodos é que, em geral, estes exigem repetidas execuções do modelo. Isso pode ser computacionalmente proibitivo para modelos complexos, especialmente modelos multiescala. Existem técnicas de inferência Bayesiana aproximada que reduzem o tempo computacional geral, embora ainda possam ser inviáveis se o modelo for caro para ser avaliado. Para amenizar a dificuldade desse procedimento, propomos o uso de um substituto (ou metamodelo). Nosso modelo substituto é um Processo Gaussiano, que é um modelo baseado em dados recentemente usado em aprendizado de máquina e no contexto de metamodelagem. Este trabalho revisa aspectos teóricos e práticos dos processos Gaussianos (GP) em um problema de regressão, discutindo como construir um GP adequado para um problema específico, cobrindo desenvolvimentos recentes nesta área. Em seguida, definimos um modelo GP substituto e o combinamos com um procedimento de computação Bayesiana aproximada usando o método de Markov-Chain Monte Carlo, o método ABC-MCMC, para resolver o problema inverso. Comparamos as abordagens padrão e proposta em dois modelos de crescimento tumoral. Os resultados sugerem que a estratégia proposta é promissora em termos de redução do custo computacional.

Palavras-chave: Modelagem mecanística, Modelagem baseada em dados, Processos Gaussianos, Computação Bayesiana aproximada, Metamodelo, Modelagem em câncer

List of Figures

Figure 1 – Mechanistic modeling workflow. 17
Figure 2 – Data-driven modeling workflow. 17
Figure 3 – Example of a two-dimensional multivariate gaussian probability distribution. 20
Figure 4 – Functions obtained from the defined prior Gaussian process. 25
Figure 5 – Functions obtained from the posterior Gaussian process (with 1, 3, and 6 observations). 25
Figure 6 – Gaussian process with a dot product kernel. 28
Figure 7 – Gaussian process with a squared exponential kernel. 29
Figure 8 – Gaussian process with a periodic kernel. 31
Figure 9 – Gaussian process with a Matérn kernel 32
Figure 10 – Gaussian processes with distinct sum and product kernels. 35
Figure 11 – Prior samples of Gaussian processes with an anisotropic squared exponential kernel, two inputs 37
Figure 12 – Graphical representation of Deep kernel learning (DKL). 38
Figure 13 – Gaussian process regression with a squared exponential kernel (function 1). 45
Figure 14 – Gaussian process regression with a with a Matérn kernel (function 1). . 46
Figure 15 – Gaussian process regression with a dot product kernel (function 1). . . 46
Figure 16 – Gaussian process regression with a Periodic kernel (function 1). 47
Figure 17 – Gaussian process regression with the Composed kernel (function 1). . . 48
Figure 18 – Gaussian process regression with a Spectral mixture kernel [1 Gaussian, failed attempt] (function 1). 48
Figure 19 – Gaussian process regression with a Spectral mixture kernel [1 Gaussian] (function 1). 49
Figure 20 – Gaussian process regression with a Spectral mixture kernel [2 Gaussians] (function 1). 49
Figure 21 – Gaussian process regression with a Spectral mixture kernel [4 Gaussians] (function 1). 50
Figure 22 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel (function 1). 50
Figure 23 – Gaussian process regression with a deep kernel, using the spectral mixture [4 Gaussians] as base kernel (function 1). 51
Figure 24 – Gaussian process regression with a squared exponential kernel (function 2). 53
Figure 25 – Gaussian process regression with a Matérn kernel (function 2). 53

Figure 26 – Gaussian process regression with a dot product kernel (function 2). . . 54
Figure 27 – Gaussian process regression with a Periodic kernel (function 2). 54
Figure 28 – Gaussian process regression with the Composed kernel (function 2). . . 55
Figure 29 – Gaussian process regression with a Spectral mixture [1 Gaussian] (function 2). 55
Figure 30 – Gaussian process regression with a Spectral mixture [2 Gaussians] (function 2). 56
Figure 31 – Gaussian process regression with a Spectral mixture [4 Gaussians] (function 2). 56
Figure 32 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel (function 2). 57
Figure 33 – Gaussian process regression with a deep kernel, using the spectral mixture [4 Gaussians] as base kernel (function 2). 57
Figure 34 – Gaussian process regression with a squared exponential kernel (function 3). 59
Figure 35 – Gaussian process regression with a Matérn kernel (function 3). 60
Figure 36 – Gaussian process regression with the Composed kernel (function 3). . . 60
Figure 37 – Gaussian process regression with a Spectral mixture [1 Gaussian] (function 3). 61
Figure 38 – Gaussian process regression with a Spectral mixture [2 Gaussians] (function 3). 61
Figure 39 – Gaussian process regression with a Spectral mixture [4 Gaussians] (function 3). 62
Figure 40 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel. 62
Figure 41 – Gaussian process regression with a deep kernel, using the spectral mixture [4 Gaussians] as base kernel. 63
Figure 42 – Gaussian process regression with a squared exponential kernel (function 4). 64
Figure 43 – Gaussian process regression with a Matérn kernel (function 4). 65
Figure 44 – Gaussian process regression with a dot product kernel (function 4). . . 65
Figure 45 – Gaussian process regression with the Composed kernel (function 4). . . 66
Figure 46 – Gaussian process regression with a Spectral mixture kernel [1 Gaussian] (function 4). 66
Figure 47 – Gaussian process regression with a Spectral mixture kernel [4 Gaussians] (function 4). 67
Figure 48 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel (function 4). 67

Figure 49 – Gaussian process regression with a deep kernel, using the spectral mixture [4 Gaussians] as base kernel (function 4). 68
Figure 50 – Two dimensional Alpine function with $x_1, x_2 \in [0, 10]$. This function has multiple maximum and minimum points, in particular the global minimum at $\mathbf{x}_{\text{opt}} = [7.917, 7.917]$. Sampled training set inputs, used in the process of learning this function, are the blue points present in the yellow plane. 70
Figure 51 – Gaussian process regression with an anisotropic squared exponential kernel (function 5). 70
Figure 52 – Gaussian process regression with an anisotropic Matérn kernel (function 5). 71
Figure 53 – Gaussian process regression with a Spectral mixture kernel [1 Gaussian] (function 5). 71
Figure 54 – Gaussian process regression with a Spectral mixture kernel [4 Gaussians] (function 5). 72
Figure 55 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel (function 5). 72
Figure 56 – Gaussian process regression with a deep kernel, using the Spectral mixture [4 Gaussians] as base kernel (function 5). 73
Figure 57 – Gaussian process regression with a squared exponential kernel and assuming outputs are independent (function 6). 74
Figure 58 – Gaussian process regression with a squared exponential kernel combined with a Multitask kernel (function 6). 75
Figure 59 – Gaussian process regression with a Spectral mixture kernel [1 Gaussian] and assuming outputs are independent (function 6). 76
Figure 60 – Gaussian process regression with a Spectral mixture kernel [1 Gaussian] combined with a Multitask kernel (function 6). 77
Figure 61 – Gaussian process regression with a Spectral mixture kernel [4 Gaussians] and assuming outputs are independent (function 6). 78
Figure 62 – Gaussian process regression with a Spectral mixture kernel [4 Gaussians] combined with a multitask kernel (function 6). 79
Figure 63 – Outline of the GP surrogate construction. 88
Figure 64 – Outline of how the GP surrogate produces QoI estimates at empirical times. 88
Figure 65 – Posterior distributions for the parameters κ (carrying capacity) and r (maximum growth rate). 92

Figure 66 – Verhulst model predictions with the MAP estimates returned from ABC-MCMC and ABC-MCMC with GP surrogate approaches. The results are in accordance, with less than 2% difference in the carrying capacity. 93
Figure 67 – Overview of the two scale model. 94
Figure 68 – Snapshots of the model along 4 days. From left to right: simulations at $t = 1, 2, 3$, and 4 days. 95
Figure 69 – Calibrated parameters $\bar{\alpha}_P$ (proliferation rate) and $\bar{\alpha}_D$ (apoptosis rate) using ABC-MCMC with GP surrogate with different number of samples. 97
Figure 70 – Posterior distributions for the calibrated parameters obtained with ABC-MCMC and GP surrogate with 100 (left) and 1000 (right) samples. 97
Figure 71 – Posterior distributions for the calibrated parameters obtained with ABC-MCMC method without using the GP surrogate. 97
Figure 72 – Simulated live and dead cell confluences, using the MAP estimates obtained with the ABC-MCMC method with GP surrogate with 100 and 1000 samples. Red and green dots indicate the hypothetical dead and live experimental confluences, shown together with their corresponding error bars. 98

List of Tables

Table 1 – Kernels used in the following examples, assuming unidimensional inputs, for which we use $r = |x - x'|$ 43

Table 2 – Hyperparameter tuning process learning rate and iterations values for the $f_1(x)$, $f_2(x)$, and $f_4(x)$ function learning problems. We may notice that more complex kernels require more computations and are more sensitive to variations in the hyperparameters. 45

Table 3 – Hyperparameter tuning process learning rate and iterations values for the $f_5(\mathbf{x})$ function learning problem. As in the previous examples, kernel functions with higher complexity required more robust choices for learning rate and iterations. 69

Table 4 – Hyperparameter tuning process learning rate and iterations values for the $\mathbf{f}_6(x)$ function learning problem. Even though the Multitask kernel introduced more hyperparameters to be estimated, the same set up was sucessful for both cases. 74

Table 5 – Bayesian inference set up for the Verhulst model parameters. The distance function is weighted by the empirical data standard deviations $\hat{\sigma}_1 \dots \hat{\sigma}_n$. 91

Table 6 – Maximum *a posteriori* estimate using ABC-MCMC and ABC-MCMC with GPR for the Verhulst model, in appropriate units (a.u.). 91

Table 7 – Fixed parameters for the two-scale model. 96

Contents

1 Introduction **15**

2 Gaussian Process Regression **19**

2.1 Introduction 19

2.2 Preliminaries: multivariate Gaussian distribution 20

2.3 Gaussian processes 22

2.3.1 Introduction 22

2.3.2 Gaussian process regression 23

2.3.2.1 A simple example 24

2.3.3 Kernel functions 26

2.3.3.1 Dot product kernel 26

2.3.3.2 Squared exponential kernel 27

2.3.3.3 Periodic kernel 30

2.3.3.4 Matérn Kernel 30

2.3.4 Hyperparameter tuning 31

2.3.5 Advanced kernels 33

2.3.5.1 Sum/Product kernels 34

2.3.5.2 Anisotropic kernels 34

2.3.5.3 Spectral Mixture kernel 36

2.3.5.4 Deep kernel learning 38

2.3.6 Multi-output Gaussian Processes 39

2.3.6.1 Independent outputs 39

2.3.6.2 Correlated outputs 40

2.4 Discussion 41

3 Exploring Gaussian Process Regression **42**

3.1 $f_1(x) = \sin(4x)$ 43

3.2 $f_2(x) = \sin(4x) - 0.12x^3$ 51

3.3 f_3 (step function) 58

3.4 $f_4(x)$ (CO_2 dataset) 63

3.5 $f_5(\mathbf{x}) = -\sqrt{x_1}\sqrt{x_2}\sin(x_1)\sin(x_2)$ 68

3.6 $\mathbf{f}_6(x) = [3\cos(x),\ \ 2\cos(x+0.3)]$ 73

3.7 Discussion 79

4 Gaussian Process as a Surrogate Model **81**

4.1 Introduction 81

4.1.1 Bayesian inference 81

4.1.2 Approximate Bayesian Computation 83

4.1.3 Surrogate model 85

4.2 Regression and surrogate model description 86
4.3 Surrogate model: construction and prediction 87
4.4 Discussion 88
5 GP Surrogate Models in Tumor Growth 90
5.1 Verhulst model 90
5.2 Multiscale model (two-scale) 93
5.3 Discussion 99
6 Conclusion 100

Bibliography 102

1 Introduction

It is of no surprise that scientific computing is used as a tool for understanding phenomena of distinct areas of knowledge. Problems from engineering, biology, finances, social sciences, etc., may be formulated and solved with a combination of theory and experiment. These problems are commonly represented by mathematical models, which are solved with the use of computational methods (Velten, 2009; Heath, 2018; Gustafsson, 2018). These models are formulated based on hypothesized relations between observed quantities. In general, these are of a form known as input-output systems, transforming a given input into its corresponding output. With this output, it is possible to obtain a quantity of interest (QoI), which is essentially the answer to the proposed problem. Examples of such mathematical models include algebraic equations, differential equations, Markov chains, and statistical models (Giordano; Fox; Horton, 2013; Heath, 2018; Velten, 2009).

Evaluating such models is fundamental, although it is also important that the models reflect the available empirical data. Model fitting (also known as parameter inference, model calibration, model training, or inverse problem) is a task also performed in order to provide model outputs in accordance with data (Zhang; Yin; Wang, 2020; Hansen; Pereyra; Scherer, 2013). There are different approaches to perform this task, some of which incorporate uncertainties arising from the experiments and the model itself (Gamerman; Lopes, 2006; Kruschke, 2014; Collis et al., 2017). It is worth mentioning that they all have a common characteristic: they require multiple evaluations of the model in the process. Therefore, model fitting can be even more challenging than computing model simulations.

Depending on the elected model, accomplishing both model evaluation and fitting may not be simple. Complex models are prone to considerable computational time. Nonlinearities, stochasticity, and a large number of parameters are some of the factors which increase model complexity. Moreover, the amount of data may not be sufficient or the data may not be very informative to conduct a successful model fitting. A part of scientific computing is dedicated to avoiding or at least alleviating such difficulties.

In the literature, there are two major types of mathematical models in scientific computing, mechanistic and statistical models (Velten, 2009). Mechanistic (or physics-based) models are developed based on observing a certain phenomenon, making some hypothesis, and proposing a mathematical model which relates the quantities involved in the process. These models portray the understanding of the modeler of a specific process. From a mathematical standpoint, the input-output system is described by a set of parameters, and each of these parameters is related to a certain observed mechanism.

For example, the rate of contagion of a susceptible individual by a infected one is typically a parameter in a epidemic model (Volpatto et al., 2020; Korolev, 2021). Another common parameter present in biological systems is the maximum population size of individuals that can be sustained in a certain environment, usually referred as the carrying capacity (Tsoularis; Wallace, 2002; Murphy; Jaafari; Dobrovolny, 2016). As for physical systems, the value of the spring constant in a mass-spring system is also a parameter. By introducing the values associated with each parameter in the model, it is possible to evaluate the model and obtain the associated QoI. Moreover, it is also possible to identify the influence of each parameter over the obtained QoI, performing a task known as sensitivity analysis (Zhang; Yin; Wang, 2020).

Another type of mathematical models are statistical (or data-driven) models (Velten, 2009; Zhang; Yin; Wang, 2020; Kendall et al., 1999; Peng et al., 2021). Unlike the previous mechanistic models, they have a more general form in which adjusting to a particular scenario of interest is possible by incorporating a sufficient amount of data. On the other hand, it is usually not possible to associate the parameters present in such models with a particular physical or biological factor, which would give a better knowledge of how a certain QoI is obtained.

Independently of the chosen model approach, an important step in the scientific problem is to evaluate the model performance, in other words, determining if the model is actually useful at representing the scenario of interest. This is particularly important, since proposing a model and fitting it to the available data may be a difficult task, for reasons previously mentioned. By incorporating this step before obtaining model simulations, we may describe a simplified workflow for the mechanistic or data-driven approaches, as in Figures 1 and 2, respectively.

The use of a mechanistic modeling approach has been more common in contexts where data availability is not very large, or it is only necessary to describe the phenomenon qualitatively (not necessarily quantitatively). When there is a large amount of available data and it is not necessary (or even possible) to outline the relationship between the quantities in the process, data-driven models are preferred. More recently, hybrid strategies that combine both mechanistic and data-driven approaches have been proposed for complex scientific processes (Alber et al., 2019; Peng et al., 2021).

The Gaussian process is an example of a data-driven model used in a wide variety of applications (Rasmussen; Williams, 2006). It was first introduced by Danie G. Krige as a regression technique, in the problem of inferring the distribution of gold using data samples of boreholes. Hence, the technique of performing Gaussian process regression is also known as kriging (Krige, 1951). The use of Gaussian processes has been common in scientific computing, being a powerful tool in machine learning (Liu et al., 2020). Interestingly, this type of data-driven model has been used in combination of mechanistic models,

specifically as a surrogate model (Chiplunkar; Bosco; Morlier, 2017; Preuss; Toussaint, 2018). The surrogate model is an approximation of a certain model, utilized in contexts which executing such model repeatedly is troublesome. There are lots of scientific problems in which this is the case, for example, in parameter inference tasks.

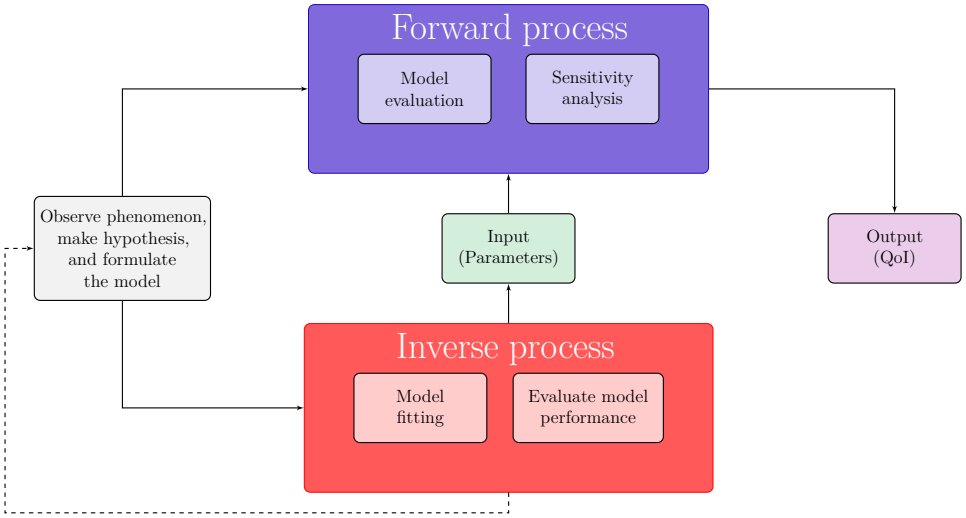


Figure 1 – Mechanistic modeling workflow. The forward process is related to providing input parameters and evaluating the model, obtaining the associated QoI, and also understanding how the input relates to the output with a sensitivity analysis. The inverse process provides the parameters so that the model reflects the available data, although the evaluated model performance may suggest a return to the initial modeling steps.

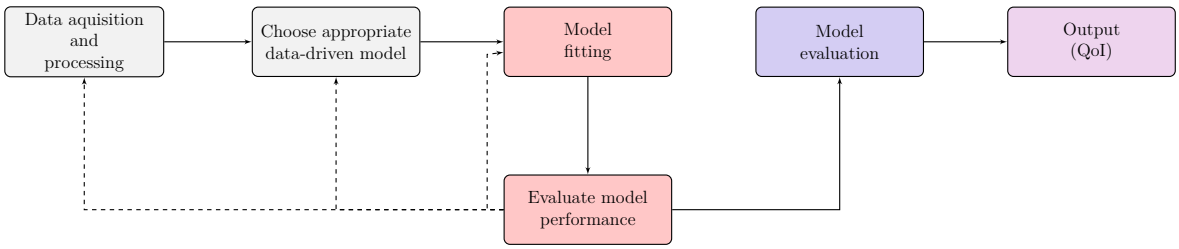


Figure 2 – Data-driven modeling workflow. After acquiring data and choosing a proper data-driven model, it is mandatory to perform a model fitting step. If the model is considered adequate according to a model performance evaluation, it may be used in simulations. Otherwise, it may be necessary to revisit the data acquisition step or to select another type of data-driven model.

This work aims at reviewing the theory and applications of Gaussian processes, as well as its combination with mechanistic models in surrogate modeling. Firstly, we define Gaussian process mathematically, presenting properties and recent advances in the topic (Chapter 2). Then, in order to discuss and provide a better understanding of the method, we apply this technique in distinct regression problems (Chapter 3). We also discuss the problem of performing parameter inference in mechanistic models, in particular, using a

Bayesian approach. After highlighting difficulties in this process over complex models, we discuss the idea of using surrogate models, as well as how to construct and use a Gaussian process surrogate as a tool to alleviate the overall computational burden (Chapter 4). Following, we apply this methodology in two distinct tumor growth models, so as to illustrate the advantages of using a surrogate model (Chapter 5). Lastly, we discuss the results and future works capable of improving the proposed methodology (Chapter 6).

2 Gaussian Process Regression

This chapter starts by introducing regression problems, presents the definition of Gaussian process, and then describes its mathematical properties. The content presented in this chapter is based on the reference (Rasmussen; Williams, 2006).

2.1 Introduction

Regression problems are present in different areas of knowledge such as engineering, physics, chemistry, and biology. This type of problems may be formulated as following:

Let $\{\mathbf{x}^{(j)}, y^{(j)}\}_{j=1}^{n_{\text{train}}}$ be a pair of observations (training set), in which $\mathbf{x}$ denotes an input vector (covariates) and y denotes an output or target (dependent variable), given a new input $\mathbf{x}^*$ obtain its associated output $y(\mathbf{x}^*)$. The inference process is executed by using a prediction model $\hat{y}(\mathbf{x})$, and it is usually assumed that an additive error $\boldsymbol{\epsilon}$ is present:

$$y^{(j)} = \hat{y}(\mathbf{x}^{(j)}) + \epsilon^{(j)}, \quad j = 1, \ldots, n_{\text{train}} \ .$$

Based on assumptions of the prediction model and error, it is possible to learn $y(\mathbf{x}^*)$ with least squares or maximum likelihood estimation (Hastie; Tibshirani; Friedman, 2009). These two classical methods find a single prediction model parameter vector $\boldsymbol{\theta}$ (known as point estimate) which best matches the observations, allowing predictions to be performed afterward. The issue with these approaches is that they do not quantify not quantifying uncertainty in the predicted outcomes directly, requiring additional computations (Wolberg, 2006; Pomerantsev, 1999). Uncertainty is inherently present in the output if data and model uncertainties are taken into account.

Probabilistic inference methods are more robust in the sense that not only a single function is learned. This class of methods learns the most probable function, as well as the uncertainty around it (Duvenaud, 2014).

In this context, Gaussian process regression (GPR) or kriging is a technique that originates from the context of Geostatistics when Danie G. Krige wanted to use samples of boreholes to infer the distribution of gold (Krige, 1951). Recently, it has been used in machine learning problems (Liu et al., 2020), as well as in metamodeling (Chiplunkar; Bosco; Morlier, 2017; Preuss; Toussaint, 2018). Before describing this method, we shall revisit some basic concepts of probability theory.

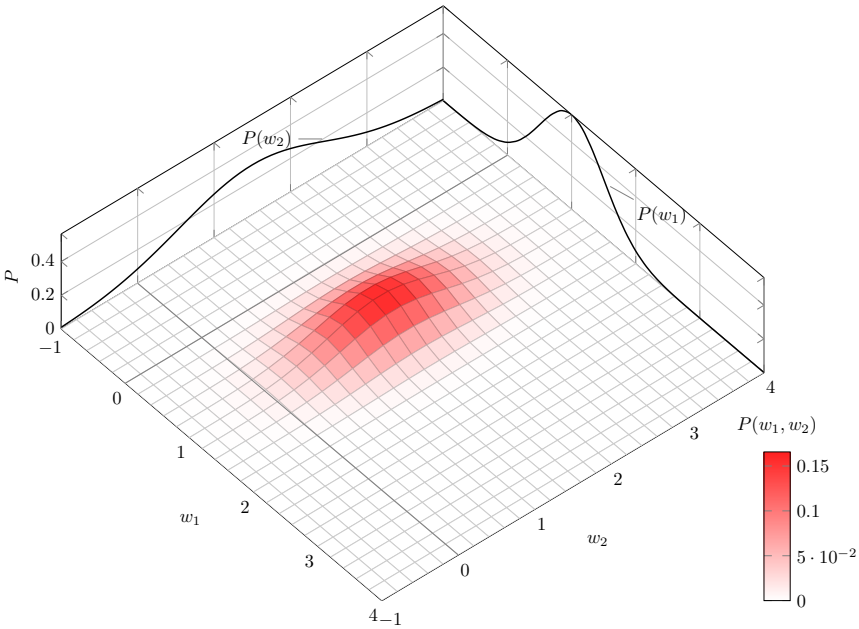


Figure 3 – Example of a two-dimensional multivariate gaussian probability distribution $P(w_1, w_2)$, with mean vector $\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 1.5 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}$. The distributions $P(w_1)$ and $P(w_2)$ represent the marginal distributions with respect to w_1 and w_2, respectively.

2.2 Preliminaries: multivariate Gaussian distribution

A random vector $\mathbf{w} \in \mathbb{R}^d$ is said to follow a multivariate Gaussian (or normal) distribution if every linear combination of its d components has a univariate normal distribution. Analogous to the one dimensional case, such multivariate distribution is defined by a mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and a symmetric positive semi-definite covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. A two-dimensional example is depicted in Figure 3.

This type of probability distribution has some nice properties:

Sum

Let $\mathbf{w}_1 \in \mathbb{R}^d$ and $\mathbf{w}_2 \in \mathbb{R}^d$ be normally distributed random vectors which are independent (here denoted as $\perp\!\!\!\perp$) :

$$\begin{aligned} \mathbf{w}_1 &\sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1), \\ \mathbf{w}_2 &\sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2), \\ \mathbf{w}_1 &\perp\!\!\!\perp \mathbf{w}_2. \end{aligned}$$

Their sum $\mathbf{w}_1 + \mathbf{w}_2$ is also normally distributed:

$$\mathbf{w}_1 + \mathbf{w}_2 \sim \mathcal{N}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \Sigma_1 + \Sigma_2). \tag{2.1}$$

Now, consider $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Suppose we split this vector into two smaller ones $\mathbf{w}_a = [w_1, \ldots, w_r]^T$ and $\mathbf{w}_b = [w_{r+1}, \ldots, w_d]^T$, and similarly for the mean vector and covariance matrix:

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \hdashline \\ \boldsymbol{\mu}_b \end{bmatrix}, \quad \Sigma = \left[\begin{array}{c:c} \Sigma_{aa} & \Sigma_{ab} \\ \hdashline \Sigma_{ba} & \Sigma_{bb} \end{array}\right], \quad \Sigma_{ba} = \Sigma_{ab}^T.$$

With this structure, we define the following properties:

Marginalization

The marginal distributions associated with $\mathbf{w}_a$ and $\mathbf{w}_b$ are also normally distributed (Do, 2008):

$$\mathbf{w}_a \sim \mathcal{N}(\boldsymbol{\mu}_a, \Sigma_a), \quad \mathbf{w}_b \sim \mathcal{N}(\boldsymbol{\mu}_b, \Sigma_b). \tag{2.2}$$

Conditioning

The conditional distributions $\mathbf{w}_a|\mathbf{w}_b$ and $\mathbf{w}_b|\mathbf{w}_a$ are also normally distributed (Do, 2008):

$$\begin{aligned} \mathbf{w}_a|\mathbf{w}_b &\sim \mathcal{N}(\boldsymbol{\mu}_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{w}_b - \boldsymbol{\mu}_b), \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}), \\ \mathbf{w}_b|\mathbf{w}_a &\sim \mathcal{N}(\boldsymbol{\mu}_b + \Sigma_{ba}\Sigma_{aa}^{-1}(\mathbf{w}_a - \boldsymbol{\mu}_a), \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab}). \end{aligned} \tag{2.3}$$

Affine transformation and sampling

Consider $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Let

$$\mathbf{z} = A\mathbf{w} + \mathbf{b},$$

in which $A \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are constant. Then, $\mathbf{z}$ has the following distribution (Tong, 2012):

$$\mathbf{z} \sim \mathcal{N}(A\boldsymbol{\mu} + \mathbf{b}, A\Sigma A^T). \tag{2.4}$$

Assuming the covariance matrix is positive definite, it is possible to determine its Cholesky decomposition. We could generate samples from every normally distributed random vector executing the steps below:

1. Generate a random vector with every entry $[\mathbf{v}_{(sample)}]_i$ being a sample from a unidimensional standard normal distribution. This is exactly a sample of $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$.

2. Find the Cholesky decomposition of the covariance matrix $\Sigma \to SS^T$.

3. Compute $\mathbf{w}_{(sample)} = S\mathbf{v}_{(sample)} + \boldsymbol{\mu}$. By the affine transformation property (2.4), this will be a sample of the desired distribution $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, SS^T)$.

2.3 Gaussian processes

2.3.1 Introduction

We are now ready to introduce the definition of Gaussian processes, and how they can be employed in regression problems. Essentially, a Gaussian process is a probability distribution of functions. It has the mean function being the most probable function, while the covariance function expresses uncertainty and how each input is related.

A collection of random variables $\{h(\mathbf{x}) : \mathbf{x} \in \mathbb{X})\}$, indexed by elements of the set $\mathbb{X}$ (known as index set) is defined as a stochastic process. Gaussian processes are stochastic processes in which every finite subset of random variables forms a multivariate Gaussian distribution.

Analogous to the finite dimensional case, it is characterized by mean $m(\cdot)$ and covariance (or kernel) $k(\cdot, \cdot)$ functions:

$$h(\cdot) \sim GP(m(\cdot), k(\cdot, \cdot)) \ . \tag{2.5}$$

Considering the finite subset $\{h(\mathbf{x}^{(1)}), \ldots, h(\mathbf{x}^{(n_{\text{train}})})\}$, the random vector

$$\mathbf{h} = \begin{bmatrix} h(\mathbf{x}^{(1)}) \\ \vdots \\ h(\mathbf{x}^{(n_{\text{train}})}) \end{bmatrix}$$

has the following multivariate Gaussian distribution:

$$\begin{bmatrix} h(\mathbf{x}^{(1)}) \\ \vdots \\ h(\mathbf{x}^{(n_{\text{train}})}) \end{bmatrix} \sim \mathcal{N}\left(\underbrace{\begin{bmatrix} m(\mathbf{x}^{(1)}) \\ \vdots \\ m(\mathbf{x}^{(n_{\text{train}})}) \end{bmatrix}}_{\mathbf{m}}, \underbrace{\begin{bmatrix} k(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & \cdots & k(\mathbf{x}^{(1)}, \mathbf{x}^{(n_{\text{train}})}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^{(n_{\text{train}})}, \mathbf{x}^{(1)}) & \cdots & k(\mathbf{x}^{(n_{\text{train}})}, \mathbf{x}^{(n_{\text{train}})}) \end{bmatrix}}_{K} \right). \tag{2.6}$$

It is worth noting that covariance functions $k(\cdot,\cdot)$ are positive definite, which in turn means that K will be a positive semi-definite matrix.

2.3.2 Gaussian process regression

As previously stated, GPR is a form of probabilistic inference. In regression problems, this methodology starts by supposing

$$y^{(j)} = \underbrace{h(\mathbf{x}^{(j)})}_{\text{Prediction model}} + \underbrace{\epsilon^{(j)}}_{\text{Noise}}, \quad j = 1, \ldots, n_{\text{train}}, \tag{2.7}$$

where $\epsilon^{(j)}$ represents independent, identically distributed Gaussian noise with zero mean and variance σ_n^2. The process begins with a prediction model being a Gaussian process

$$h(\mathbf{x}) \sim GP(0, k(\mathbf{x}, \mathbf{x}')), \tag{2.8}$$

with $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$. This initial GPR is known as the prior distribution. By definition, the probability distribution of $\mathbf{h} = [h(\mathbf{x}^{(1)}), \ldots, h(\mathbf{x}^{(n_{\text{train}})})]^T$ is:

$$\mathbf{h} \sim \mathcal{N}(\mathbf{0}, K) \ . \tag{2.9}$$

Given a new input $\mathbf{x}^*$, we aim to estimate its associated output $h(\mathbf{x}^*)$. As every subset of random variables follows a multivariate Gaussian distribution, the "augmented" vector

$$\begin{bmatrix} h(\mathbf{x}^{(1)}) \\ \vdots \\ h(\mathbf{x}^{(n_{\text{train}})}) \\ \hdashline \\ h(\mathbf{x}^*) \end{bmatrix} = \begin{bmatrix} \mathbf{h} \\ \hdashline \\ h(\mathbf{x}^*) \end{bmatrix}$$

is also normally distributed:

$$\begin{bmatrix} \mathbf{h} \\ \hdashline \\ h(\mathbf{x}^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{c:c} K & K_*^T \\ \hdashline K_* & K_{**} \end{array}\right]\right) \tag{2.10}$$

where:

- $K \in \mathbb{R}^{n_{\text{train}} \times n_{\text{train}}}, \quad [K]_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}), \qquad i = 1, \ldots, n_{\text{train}} \quad j = 1, \ldots, n_{\text{train}}$,

- $K_* \in \mathbb{R}^{1\times n_{\text{train}}}, \quad [K_*]_i = k(\mathbf{x}^*, \mathbf{x}^{(i)}), \qquad i = 1, \ldots, n_{\text{train}}$,
- $K_{**} = k(\mathbf{x}^*, \mathbf{x}^*)$.

Considering noise in the training set as stated in (2.7), the distribution

$$\begin{bmatrix} \mathbf{y} \\ \hdashline h(\mathbf{x}^*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{c:c} K + \sigma_n^2 \mathrm{I} & K_*^T \\ \hdashline K_* & K_{**} \end{array}\right]\right) \tag{2.11}$$

is also Gaussian.

Finally, prediction is done by using the conditioning expressions (2.3) for a multivariate Gaussian random variable. Essentially, a posterior distribution for the output $h(\mathbf{x}^*)$ is obtained by conditioning to the training set outputs:

$$h(\mathbf{x}^*)|\mathbf{y} \sim \mathcal{N}(K_*(K + \sigma_n^2 \mathrm{I})^{-1}\mathbf{y}, \;\; K_{**} - K_*(K + \sigma_n^2 \mathrm{I})^{-1} K_*^T). \tag{2.12}$$

The mean is the most probable outcome and the variance is an uncertainty measure around it. For the solution of the linear systems in (2.12), it is possible to use the Cholesky factorization of $K + \sigma_n^2 \mathrm{I}$.

2.3.2.1 A simple example

In order to illustrate GPR, we shall start with a regression process over observations of a simple unidimensional function $f : \mathbb{R} \to \mathbb{R}$. At first, a Gaussian process as in (2.8) has to be defined. In turn, this means specifying a kernel function. This choice heavily influences the GPR quality and must be selected based on the regression problem in question. Kernel functions are going to be detailed in the following sections. Here, we suppose the well-known squared exponential kernel is a good choice:

$$k(x, x') = {\sigma_f}^2 \exp\left(-\frac{(x - x')^2}{2{\lambda_f}^2}\right). \tag{2.13}$$

The values σ_f and λ_f are called hyperparameters whose meaning will also be discussed later on.

Setting the kernel hyperparameters $\sigma_f = 1.31$, $\lambda_f = 1.04$ and the noise hyperparameter $\sigma_n = 0.3$, we obtain the prediction model in Figure 4. At first, we notice that $m(x) = 0$ is the mean function of the Gaussian process. Also, we may see the variance is the same for all inputs.

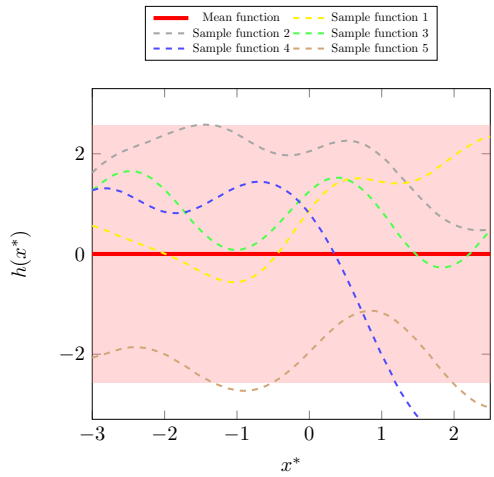
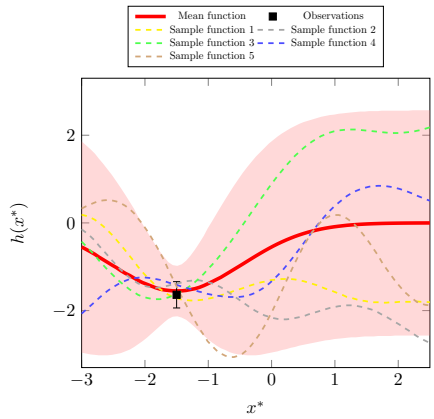
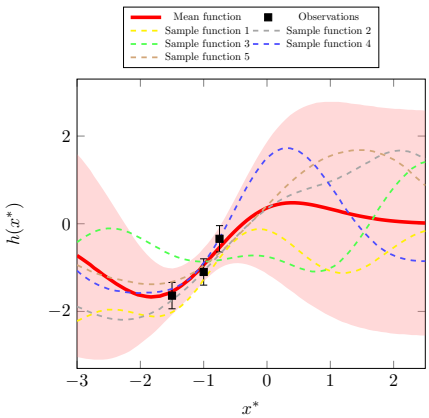


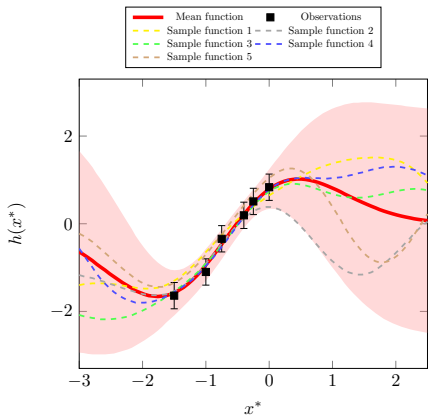
Figure 4 – Functions obtained from the defined prior Gaussian process, using $x^* \in [-3, 2.25]$ as input. The mean function in red is the most probable function evaluated in the input, while the dashed functions are samples from the distribution. The shaded area represents a 95% confidence region (mean plus and minus 2 times the standard deviation for each input).



(a) 1 observation



(b) 3 observations



(c) 6 observations

Figure 5 – Functions obtained from the posterior Gaussian process (with 1, 3, and 6 observations). The shaded area represents a 95% confidence region. It is possible to note that the mean function (in red) now predicts values in accordance with the observations, as well as the variance shrinking around them.

With a single observation $(x^{(1)}, y^{(1)}) = (-1.5, -1.636)$, it is possible to obtain a posterior estimate as shown in Figure 5a. We are able to notice how the mean and variance considerably changes around it. This process may be done with even more observations, as shown in Figures 5b and 5c.

2.3.3 Kernel functions

Kernel functions $k(\mathbf{x}, \mathbf{x}')$ are positive-definite functions which map two distinct inputs $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$ into a real number. It can be shown that these functions represent an inner product in a space $\mathcal{V}$ (feature space):

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}'), \tag{2.14}$$

in which $\phi : \mathbb{X} \to \mathcal{V}$ is known as the feature map. Usually, the function $\phi(\mathbf{x})$ does not need to be explicitly computed, though its existence is guaranteed by the positive definiteness of $k(\mathbf{x}, \mathbf{x}')$ (Mercer's theorem). Being inner products, kernel functions may be interpreted as measures of similarity between inputs (more precisely, similarity between inputs after being transformed by a certain function).

Kernel functions are used in distinct machine learning methods, e.g., support vector machines, principal component analysis, and spectral clustering (Hastie; Tibshirani; Friedman, 2009). In the context of Gaussian processes, they are also defined as covariance functions. As previously mentioned in (2.8), this function represents our prior belief of how the prediction function $h(\mathbf{x})$ is distributed. The posterior distribution (2.12) is heavily influenced by this choice, therefore an inadequate covariance function may result in incorrect estimates. We will start describing well-known kernels in the literature, and afterward, discuss how to calibrate their hyperparameters to match the training set.

2.3.3.1 Dot product kernel

The dot product kernel is a simple function which depends on the dot product (with an offset) of its inputs. Assuming $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$, we can write its expression as:

$$k_{\text{Dot}}(\mathbf{x}, \mathbf{x}') = {\sigma_f}^2 \left(\sum_{j=1}^{d} (x_j - \sigma_0)(x'_j - \sigma_0) \right) = \sigma_f(\mathbf{x} - \sigma_0 \mathbf{1}) \cdot \sigma_f(\mathbf{x}' - \sigma_0 \mathbf{1}), \tag{2.15}$$

in which $\mathbf{1}$ is a d-dimensional vector with all entries equal to 1.

Similar to (2.13), the values σ_f and σ_0 are the associated hyperparameters. The first one (${\sigma_f}^2$) is named signal variance or scale and the other one (σ_0) is called offset. The special case with $\sigma_0 = 0$ is defined as Linear kernel.This kernel is appropriate if the given

inputs are related linearly, which in turn means that $h(\mathbf{x})$ has a distribution only with affine functions. It is interesting to note that a GPR with this kernel is equivalent to the standard Bayesian linear regression (Murray, 2016; Wilson, 2014). For this simple kernel, the feature map ϕ is expressed without difficulties, being $\phi(\mathbf{x}) = \sigma_f(\mathbf{x} - \sigma_0\mathbf{1})$.

We may observe in Figure 6 a unidimensional case graphical representation of the function $k_{\text{Dot}}(x, x')$, as well as samples of the prior distribution if this kernel is selected. Figures 6a and 6b show how the offset σ_0 defines the input with the lowest variance. The scale hyperparameter σ_f performs a scaling to all functions in the distribution, in other words, $h(x)$ becomes $\sigma_f h(x)$ (this is visually depicted by the differences between Figure 6a and Figure 6c).

Assuming that a Gaussian process with only affine functions is appropriate for a general regression problem might be too restrictive. This motivates the study of other kernels.

2.3.3.2 Squared exponential kernel

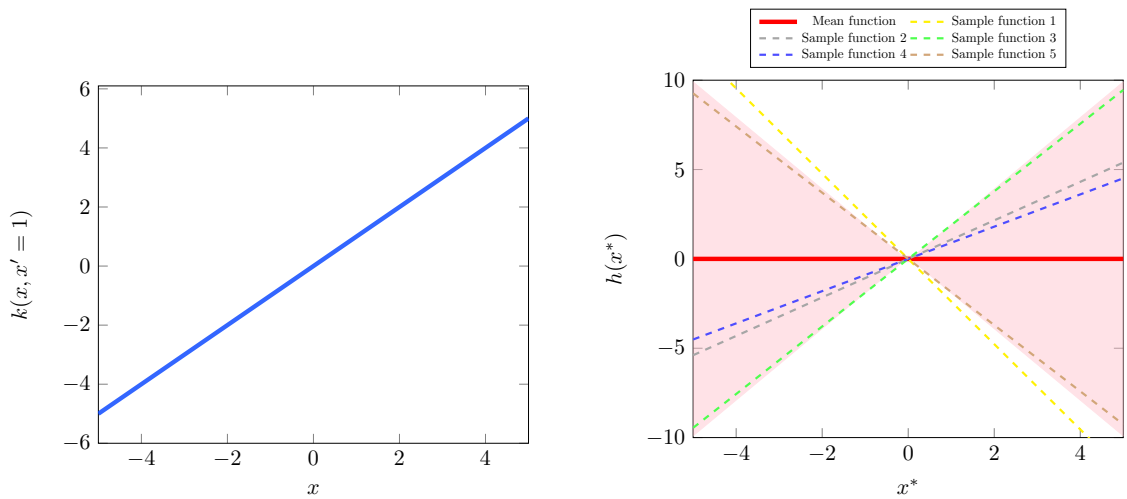
We introduced the squared exponential kernel in our preliminary example of GP regression. In the literature, it is also referred as RBF (radial basis function) kernel.This is one of the most used covariance functions in applications, most likely due its nice properties. An analogous version of (2.13) for multidimensional inputs is

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = {\sigma_f}^2 \exp\left(-\frac{1}{2}\frac{||(\mathbf{x} - \mathbf{x}')||_2^2}{{\lambda_f}^2}\right), \tag{2.16}$$

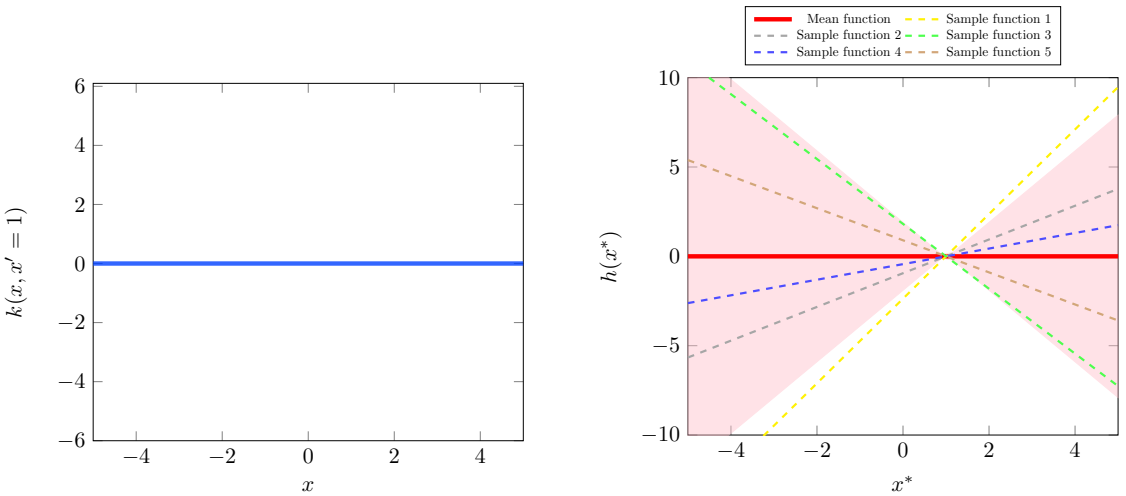
where the hyperparameter λ_f is known as the lengthscale. Unlike the dot product kernel, k_{SE} (2.16) is a function of the difference between two inputs. A kernel with this property is called stationary kernel.

The expression (2.16) suggests that if an input pair is "close", they will be similar. This is a characteristic present in smooth functions, and this is reflected in the corresponding prior having infinitely differentiable functions. Hence, the value λ_f is responsible for controlling this notion of "closeness". From Figure 7a to Figure 7c, we may notice how the functions become "flatter" (similar function values in nearby inputs) as the lengthscale λ_f increases.

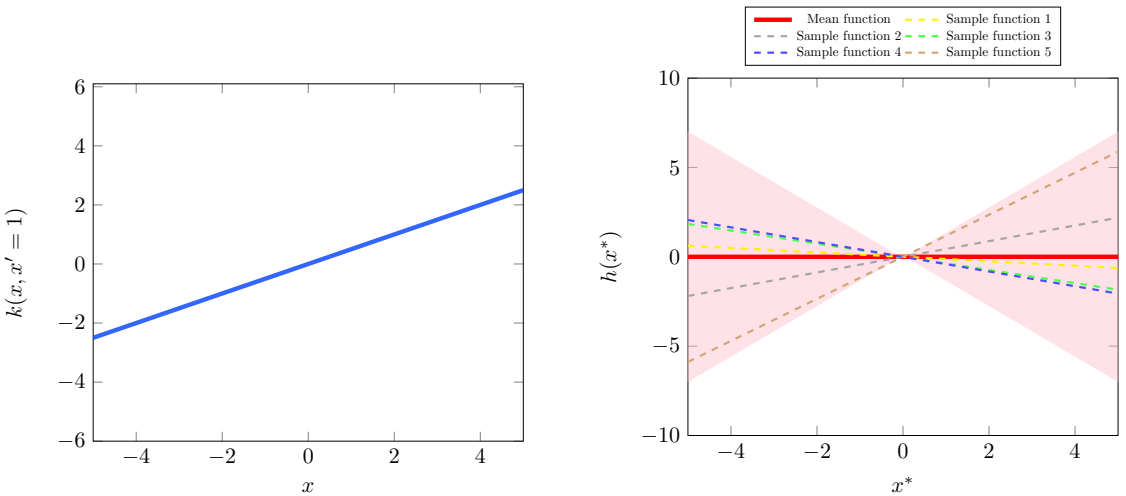
Like polynomials and feedforward neural networks, the squared exponential kernel is a universal approximator for continuous functions on compact subsets of $\mathbb{R}^d$ (Pérez; Quintana, 2006; Hornik, 1991; Micchelli; Xu; Zhang, 2006). In other words, this means that if the function being learned is continuous, the error committed in GPR estimates (for a new input $\mathbf{x}^*$ "in between" the training inputs) will theoretically decrease as more data is used in the training set.



(a) $\sigma_f = 1$, $\sigma_0 = 0$

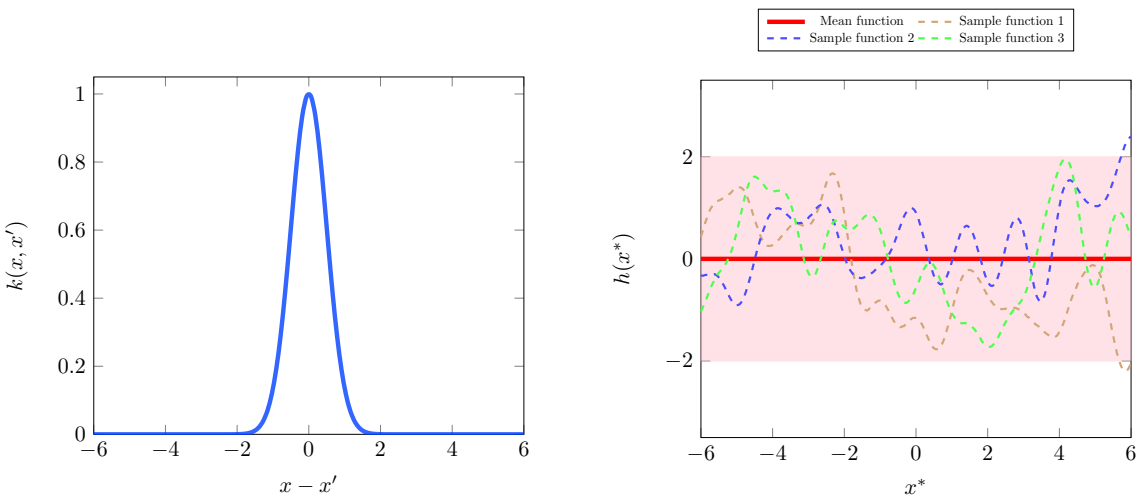


(b) $\sigma_f = 1$, $\sigma_0 = 1$

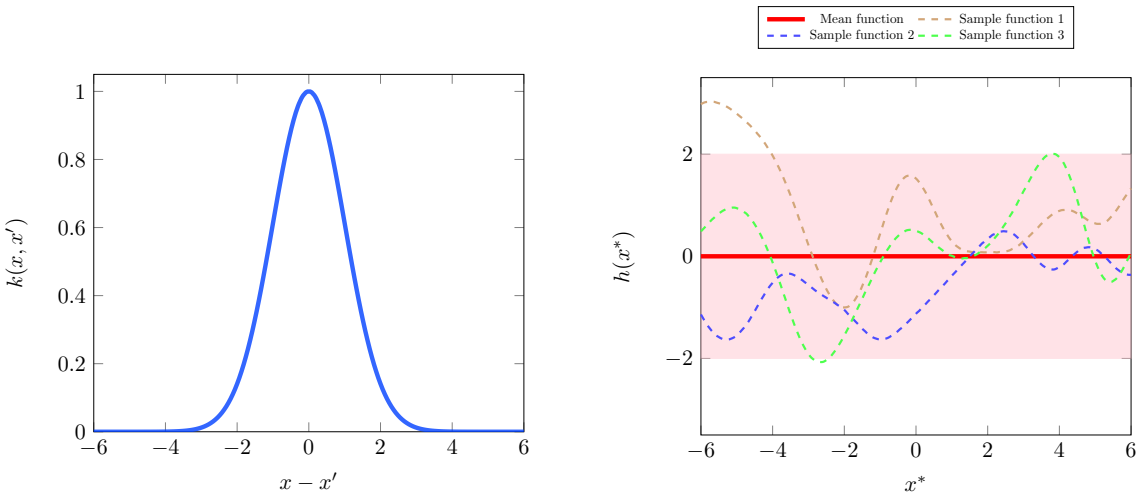


(c) $\sigma_f = 0.5$, $\sigma_0 = 0$

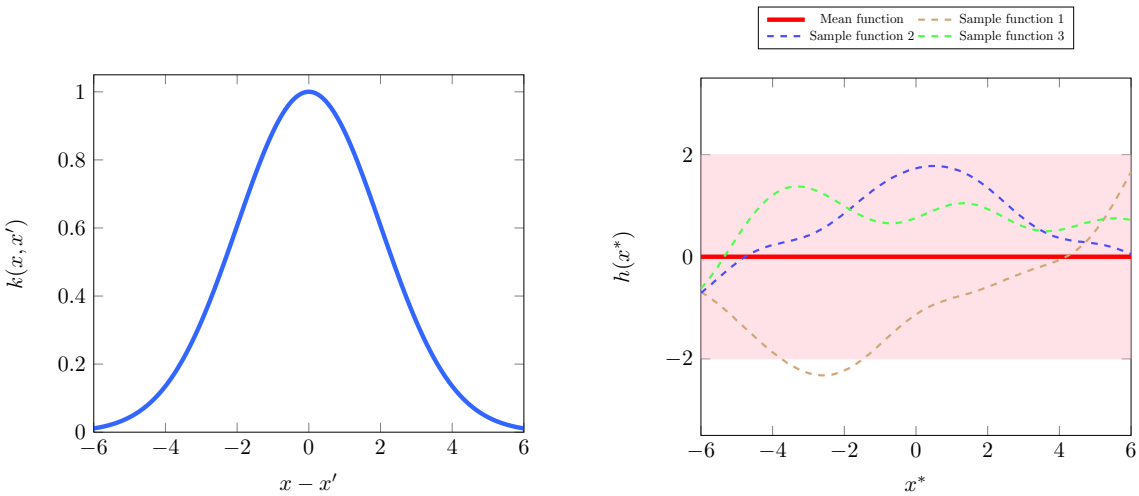
Figure 6 – Gaussian process with a dot product kernel, using $x^* \in [-5, 5]$ and different combinations of σ_f and σ_0. On the left: plot of the kernel function $k(x, x')$ using a fixed $x' = 1$. On the right: prior mean function and samples. The shaded area represents a 95% confidence region. The effects of σ_0 and σ_f are outlined: the former defines the input with the lowest variance (visually the location where all affine functions intersect) and the latter scales all the functions in the Gaussian process.



(a) $\lambda_f = 0.5$



(b) $\lambda_f = 1$



(c) $\lambda_f = 2$

Figure 7 – Gaussian process with a squared exponential kernel, using $x^* \in [-6, 6]$, $\sigma_f = 1$, and different lenghtscales. On the left: plot of the kernel function $k(x, x')$. On the right: prior mean function and samples. The shaded area represents a 95% confidence region. We are able to recognize differences in the Gausssian process due to changes in the lengthscale λ_f. As the lengthscale λ_f increases, the functions tend to become "flatter" (nearby inputs have similar function values).

2.3.3.3 Periodic kernel

As the name suggests, this kernel should be integrated in a Gaussian process when the prediction model is a periodic function. Its derivation for the unidimensional case is based on the two-dimensional version of the squared exponential kernel, after a mapping $\mathbf{u} : \mathbb{R} \to \mathbb{R}^2$, $\mathbf{u}(x) = \left(\cos\left(\frac{2\pi x}{p}\right), \sin\left(\frac{2\pi x}{p}\right)\right)$. The whole argument is outlined in (MacKay, 1998). For multidimensional inputs, it has the following mathematical expression:

$$k_{\text{Per}}(\mathbf{x}, \mathbf{x}') = {\sigma_f}^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi||\mathbf{x}-\mathbf{x}'||_2}{p}\right)}{{\lambda_f}^2}\right). \tag{2.17}$$

Compared to (2.16), it has a new hyperparameter p representing the period length of the covariance function. The effects of this hyperparameter are shown in Figure 8: for lower values of p, highly oscillating functions are more likely to be sampled from the Gaussian process.

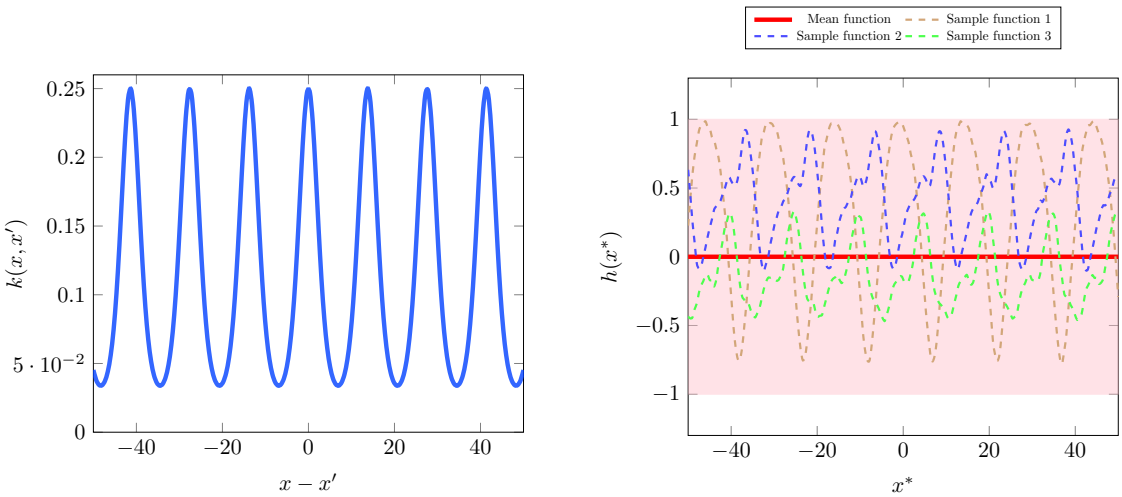
2.3.3.4 Matérn Kernel

The squared exponential kernel is vastly used, although it supposes the function being learned is infinitely differentiable. For real world problems, this might not be the case. In order to alleviate the smoothness assumption, the Matérn kernel is a possible alternative (Stein, 2012). It is a generalization of the squared exponential kernel, with an additional hyperparameter ν. This hyperparameter controls the smoothness of $h(\mathbf{x})$, and by taking the limit of $\nu \to \infty$ the squared exponential kernel is recovered.

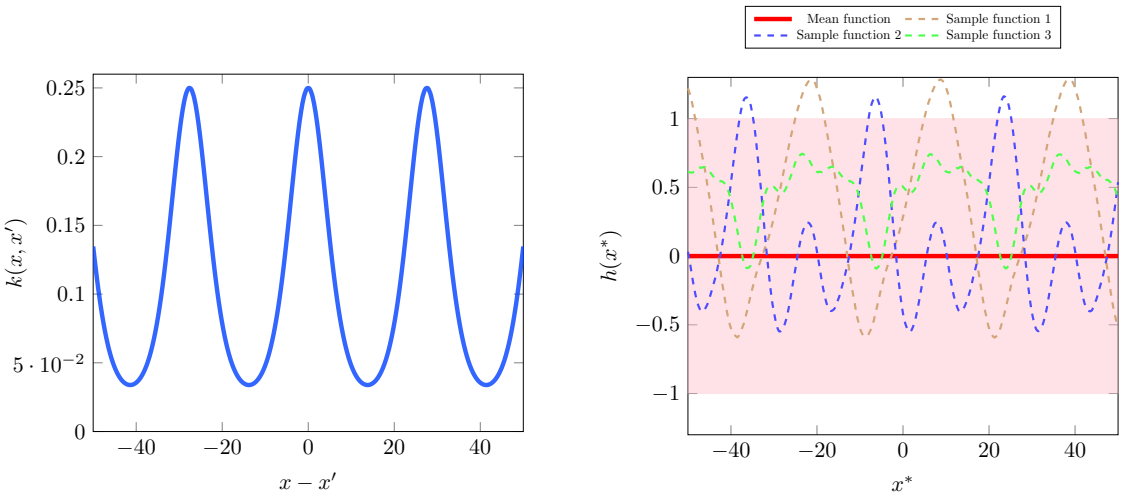
The general expression of this kernel contains modified Bessel functions, which are costly to be evaluated. Fortunately, the case for which the hyperparameter ν is half integer ($\nu = 0.5 + s$, $s \in \mathbb{Z}^+$) has a simpler expression (Abramowitz; Stegun, 1965). Denoting $r = \frac{||\mathbf{x}-\mathbf{x}'||_2}{\lambda_f}$, we have the expression as

$$k_{\text{Matern}}(\mathbf{x}, \mathbf{x}') = {\sigma_f}^2 \exp\left(-\sqrt{2s+1}r\right) \frac{s!}{(2s)!} \sum_{j=0}^{s} \frac{(s+j)!}{j!(s-j)!} \left(2\sqrt{2s+1}r\right)^{s-j}. \tag{2.18}$$

According to (Rasmussen; Williams, 2006), the values $\nu = 1.5$ and $\nu = 2.5$ are the most interesting for GPR. This is due to the fact that for $\nu \geq 3.5$, it is difficult to detect the differences between using this kernel or a squared exponential kernel. Furthermore, the case with $\nu = 0.5$ is also defined as an Ornstein-Uhlenbeck kernel. Illustrative examples of GP with different values of ν are depicted in Figure 9. Of note, sampled functions are smother as ν increases.



(a) $p = 15$



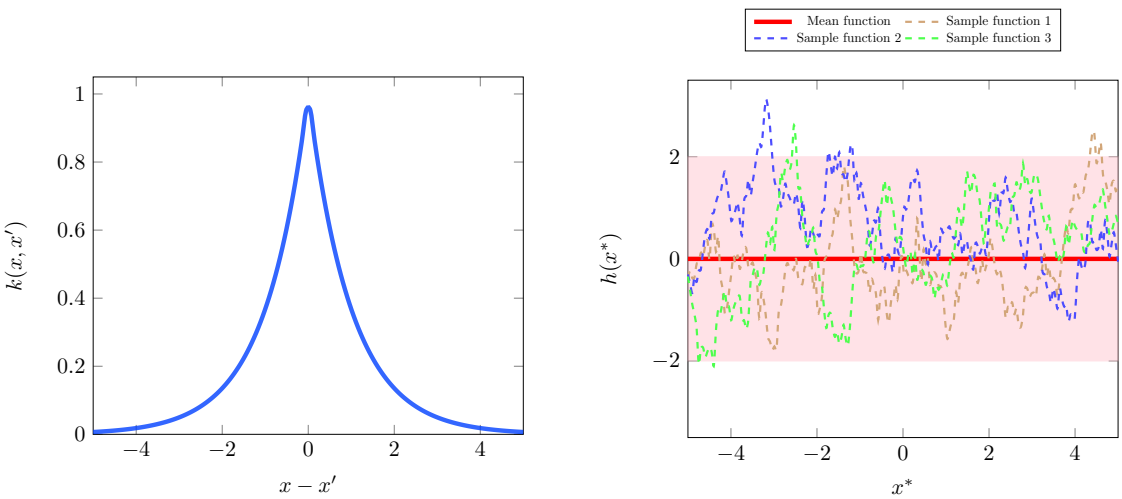
(b) $p = 30$

Figure 8 – Gaussian process with a periodic kernel, using $x^* \in [-50, 50]$, $\sigma_f = 0.5$, and $\lambda_f = 1$. On the left: plot of the kernel function $k(x, x')$. On the right: prior mean function and samples. The shaded area represents a 95% confidence region. Lower values of p increases the probability associated to highly oscillating functions in the distribution.

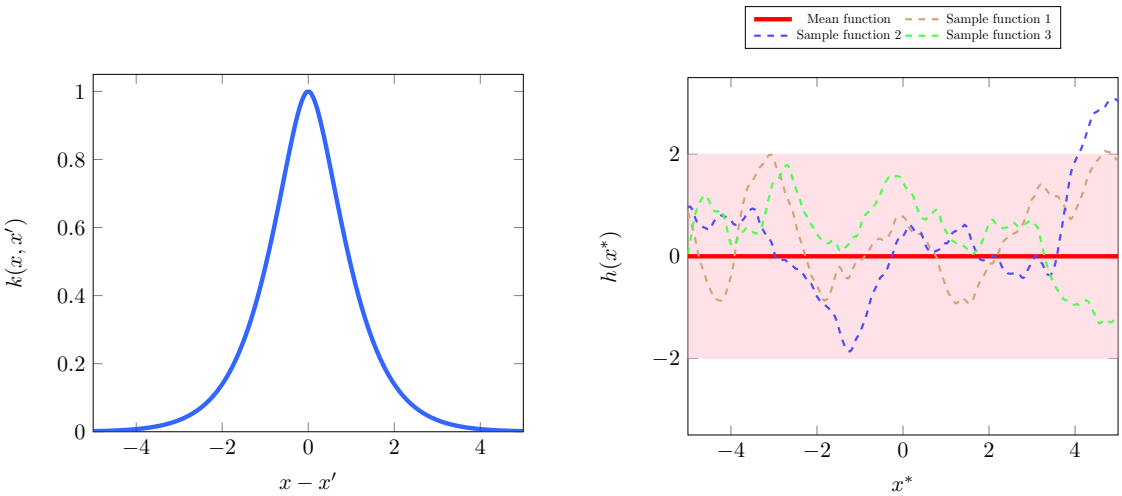
2.3.4 Hyperparameter tuning

As we already noticed, changes in the hyperparameters heavily affect a Gaussian process. This means that an arbitrary choice of hyperparameters may compromise the quality of the posterior estimates. Approaches which select hyperparameters based on the training set are usually employed, the most common one being maximizing the logarithm of the *marginal likelihood* (Wilson, 2014). Here we denote the set of hyperparameters by a vector $\boldsymbol{\psi}$, which may or not include noise hyperparameter σ_n. Of note, if the modeler knows the noise level in advance or knows that overfitting should be avoided, it is possible to set either a fixed value or a range for σ_n.

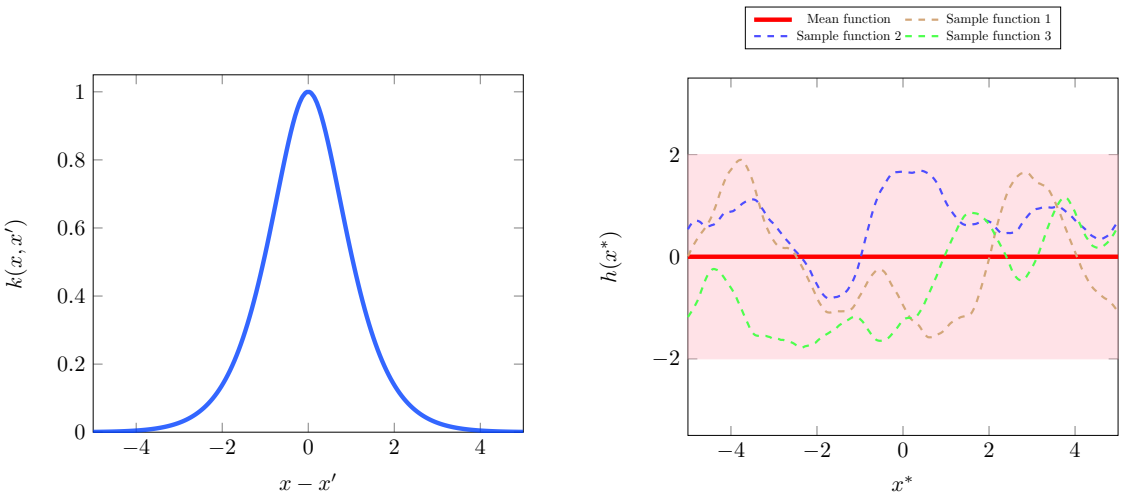
Using $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 \mathrm{I})$ (2.11), the logarithm of the *marginal likelihood* is given



(a) $\nu = 0.5$



(b) $\nu = 1.5$



(c) $\nu = 2.5$

Figure 9 – Gaussian process with a Matérn kernel, using $x^* \in [-5, 5]$, $\sigma_f = 1$, and $\lambda_f = 1$. On the left: plot of the kernel function $k(x, x')$. On the right: prior mean function and samples. The shaded area represents a 95% confidence region. We may observe that as the hyperparameter ν increases, the sampled functions have more smothness.

by:

$$\mathcal{L}(\boldsymbol{\psi}) := \ln(p(\mathbf{y}|\mathbf{x}, \boldsymbol{\psi})) = \underbrace{-\frac{1}{2}\mathbf{y}^T(K_{\boldsymbol{\psi}})^{-1}\mathbf{y}}_{\text{model fit}} - \underbrace{\frac{1}{2}\ln(\det(K_{\boldsymbol{\psi}}))}_{\text{complexity penalty}} - \underbrace{\frac{n_{\text{train}}}{2}\ln(2\pi)}_{\substack{\text{normalization constant} \\ \text{(comes from the} \\ \text{normal distribution)}}}, \tag{2.19}$$

in which $K_{\boldsymbol{\psi}} = K + \sigma_n^2\mathrm{I}$ is the matrix determined by the kernel and hyperparameters. The first term is named model fit, since it is the only one with the presence of training outputs $\mathbf{y}$. The complexity penalty or *Occam's razor* term is related to the kernel function complexity and the level of noise (Wilson, 2014). The hyperparameter vector $\boldsymbol{\psi}$ is obtained by solving

$$\boldsymbol{\psi}^* = \arg\max\left[\,\mathcal{L}(\boldsymbol{\psi})\,\right]. \tag{2.20}$$

The difficulty with this method is that the function expressed in (2.19) is non concave, which in turn means it has multiple local optima. We may also refer to functions with this property as multimodal functions. Well-known local optimization methods are commonly applied in this context (Blum; Riedmiller, 2013; McHutchon; Rasmussen, 2011; Butler et al., 2014). These are iterative methods having the following general form:

$$\boldsymbol{\psi}^{(s+1)} = \boldsymbol{\psi}^{(s)} + \alpha\mathbf{d}^{(s)}, \tag{2.21}$$

in which $\mathbf{d}$ is an ascent direction and α is the learning rate (or step size). How $\mathbf{d}^{(s)}$ is computed at each step s is associated with the local optimizer of choice, while the learning rate is either a fixed or an adaptive value (Li; Arora, 2019). The use of these methods is subject to finding a bad local maximum point. This issue can be alleviated with the use of multiple restarts (Pošík; Huyer, 2012; Dick; Wong; Dann, 2014). The use of global optimization algorithms or even a full Bayesian approach are a possibility, but at the cost of introducing considerable additional computational burden (Petelin; Filipič; Kocijan, 2011; Garbuno-Inigo; DiazDelaO; Zuev, 2016; Lalchand; Rasmussen, 2020).

2.3.5 Advanced kernels

Not every regression problem is well described by a direct use of the previously presented kernels. Using these kernels as a basis, it is possible to build covariance functions encompassing many types of structures. It is worth mentioning that the use of more elaborate kernels in Gaussian processes introduces more operations in its evaluation process, more hyperparameters to be estimated, and more local optima are prone to be present in the likelihood function. We shall visit some of these advanced kernels in the following sections.

2.3.5.1 Sum/Product kernels

A possible way of constructing a kernel is by summing or multiplying two kernels (Duvenaud, 2014), i.e.,

$$\begin{aligned} k_{\text{sum}}(\mathbf{x}, \mathbf{x}') &= k_{\text{A}}(\mathbf{x}, \mathbf{x}') + k_{\text{B}}(\mathbf{x}, \mathbf{x}'), \\ k_{\text{prod}}(\mathbf{x}, \mathbf{x}') &= k_{\text{A}}(\mathbf{x}, \mathbf{x}') \times k_{\text{B}}(\mathbf{x}, \mathbf{x}'). \end{aligned} \tag{2.22}$$

This simple strategy allows learning functions with new structures. For example: if we wanted to perform regression assuming the function being learned is periodic with growing amplitude, we could use a product of a dot product and a periodic kernel in our prior distribution. In Figure 10a, we may observe function samples with this desired profile. In contrast, the sum of these two kernels has another effect, leading to periodic functions with an increasing (or decreasing) trend as depicted in Figure 10b.

Other types of combinations are detailed in (Duvenaud, 2014). An interesting one is the kernel obtained by the product of q dot product kernels, for which a prior of polynomial functions of qth order are placed on the Gaussian process. This resulting kernel is named polynomial kernel in the literature (Rasmussen; Williams, 2006; Duvenaud, 2014).

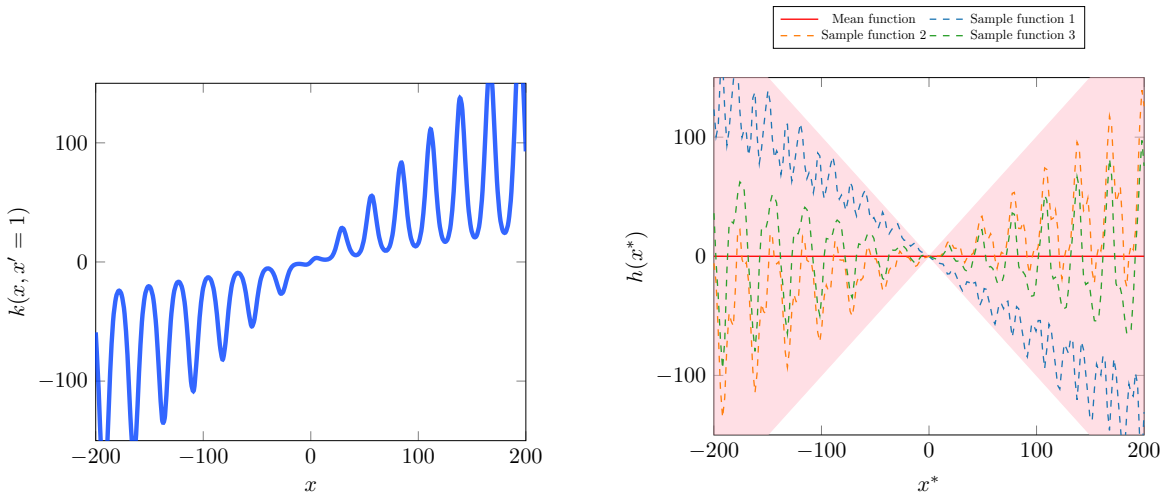
While this approach is powerful and straightforward, there is a drawback. It is difficult, at first, to define an appropriate combination of kernels for a given regression problem. There are strategies to automatically construct a kernel based on the training set and the dimension of the inputs (Duvenaud, 2014), however the additional cost involved in these procedures must be considered.

2.3.5.2 Anisotropic kernels

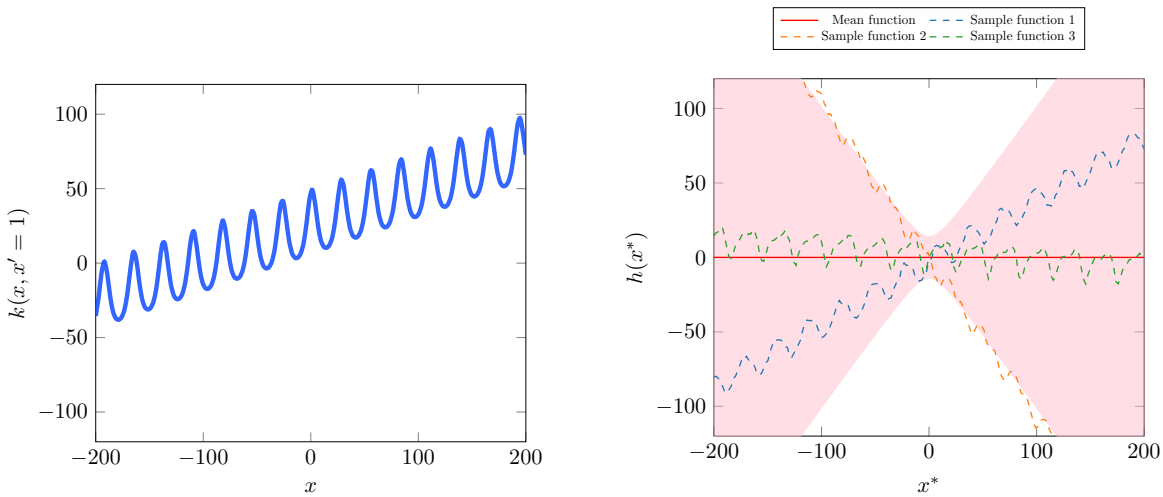
We mentioned in the previous sections that kernels that are functions of the difference between two inputs $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ are named stationary kernels. In particular, the expressions (2.16, 2.18) are functions of the Euclidean distance (normalized by a lengthscale) $r = \frac{||\mathbf{x}-\mathbf{x}'||_2}{\lambda_f}$. These are classified as isotropic kernels, since each input dimension has the same level of influence on the notion of "closeness". For added flexibility in a multidimensional setting, it is possible to introduce other notions of distance which incorporate possible heterogeneities in the input dimensions. A possibility is given by

$$r = \sqrt{(\mathbf{x} - \mathbf{x}') \cdot \Lambda^{-2} \cdot (\mathbf{x} - \mathbf{x}')} = \sqrt{\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{\lambda_i^2}}, \tag{2.23}$$

in which Λ is a diagonal matrix such that $\Lambda_{ii} = \lambda_i$ is the lengthscale associated with the ith dimension. Kernels with this property are known as anisotropic. A prominent example is the anisotropic version of the squared exponential kernel:



(a) $k_{\text{Per}} \times k_{\text{Dot}}, \quad \sigma_f = 0.5, \sigma_0 = 0, \lambda_f = 1, p = 30$



(b) $k_{\text{Per}} + k_{\text{Dot}}, \quad \sigma_f^{(\text{Dot})} = 0.5, \sigma_0 = 0, \sigma_f^{(\text{Per})} = 7, \lambda_f = 1, p = 30$

Figure 10 – Gaussian processes with distinct sum and product kernels. On the left: plot of the kernel function $k(x, x')$. On the right: prior mean function and samples. The shaded area represents a 95% confidence region. A distinct structure is obtained by multiplying or adding two kernels: (10a) the product is associated with periodic functions with growing amplitude, while (10b) the sum is related to periodic functions plus an increasing (or decreasing) trend.

$$k_{\text{AniSE}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i - x_i')^2}{{\lambda_i}^2}\right). \tag{2.24}$$

It is also called SE-ARD (squared exponential automatic relevance determination), as the hyperparameter tuning process selects lenghtscales $\lambda_1, ..., \lambda_d$, and these may be interpreted as how "relevant" a certain dimension is (Duvenaud, 2014). Furthermore, this kernel is also understood as a product of unidimensional squared exponential kernels related to each input dimension (Duvenaud, 2014):

$$\begin{aligned} k_{\text{AniSE}}(\mathbf{x}, \mathbf{x}') &= k(x_1, x_1') \times k(x_2, x_2') \times \cdots \times k(x_d, x_d') \\ &= \sigma_1^2 \exp\left(-\frac{(x_1 - x_1')^2}{2{\lambda_1}^2}\right) \times \sigma_2^2 \exp\left(-\frac{(x_2 - x_2')^2}{2{\lambda_2}^2}\right) \times \cdots \times \sigma_d^2 \exp\left(-\frac{(x_d - x_d')^2}{2{\lambda_d}^2}\right) \\ &= \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i - x_i')^2}{{\lambda_i}^2}\right). \end{aligned} \tag{2.25}$$

The difference between using isotropic (has functions samples without a preeminent direction) and anisotropic kernels (the direction with a lower value lengthscale has a higher "variation") in a Gaussian process is highlighted in Figure 11.

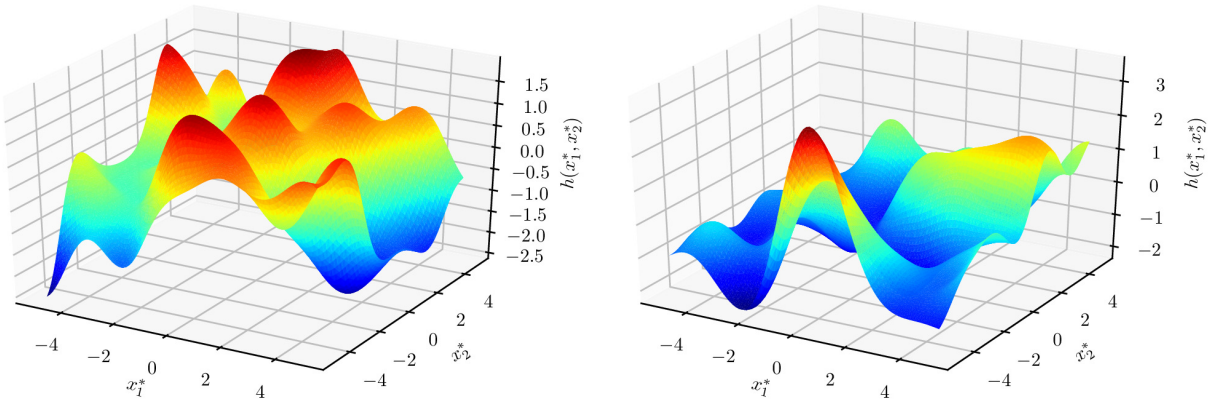
2.3.5.3 Spectral Mixture kernel

With the exception of the dot product kernel, the kernels presented until now are stationary kernels (functions of the difference between two inputs). Following from Bochner's Theorem (Wilson, 2014), by defining a spectral density function as a mixture of q Gaussians on $\mathbb{R}^d$ and taking its Fourier transform, a special type of kernel function known as the spectral mixture kernel is obtained (Wilson, 2014). Denoting $\boldsymbol{\mu}_j = [\mu_j^{(1)}, ..., \mu_j^{(d)}]$ and $V_j = \text{diag}(v_j^{(1)} ..., v_j^{(d)})$ as the jth Gaussian mean vector and variance matrix, respectively, the kernel is expressed as

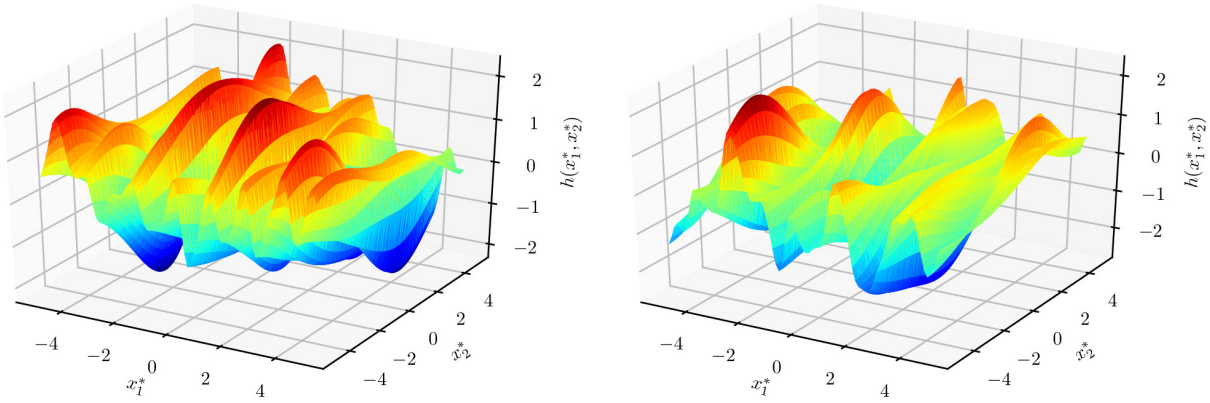
$$k_{\text{SM}}(\boldsymbol{\tau}) = \sum_{j=1}^{q}\sigma_j^2 \cos(2\pi\boldsymbol{\mu}_j \cdot \boldsymbol{\tau}) \prod_{p=1}^{d} \exp(-2\pi^2\tau_p^2 v_j^{(p)}), \tag{2.26}$$

where $\tau = \mathbf{x} - \mathbf{x}'$. In this kernel, frequency hyperparameters are $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_q$, inverse lengthscales are $V_1, ..., V_q$, and weigths (or signal variances) are $\sigma_1^2, ..., \sigma_q^2$. According to (Wilson, 2014), with a sufficient number of components, this kernel is capable of approximating all stationary kernels.

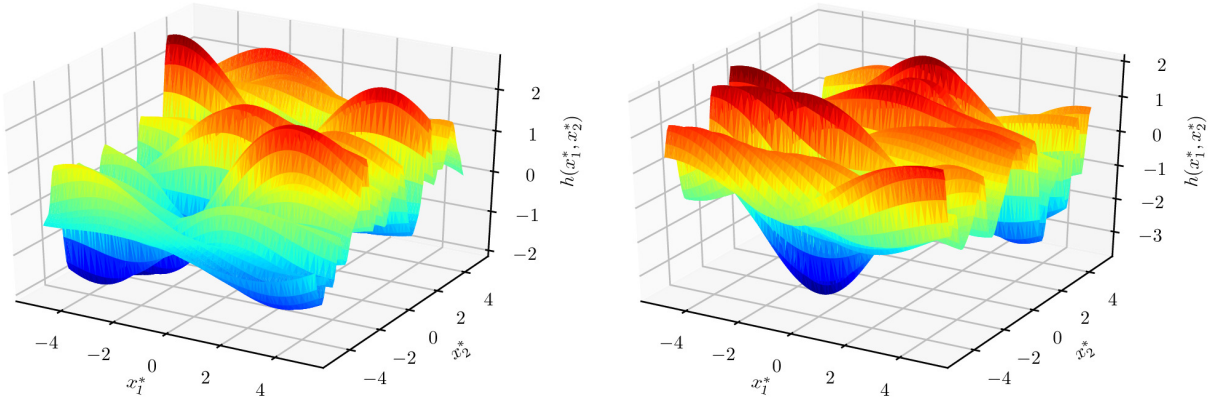
By increasing the number of Gaussian components, it is possible to represent more complex structures in a very direct manner. While this is very simple to perform from a computational perspective, it introduces a considerable difficulty in the hyperparameter



(a) $\lambda_1 = \lambda_2 = 1.25$ (isotropic)



(b) $\lambda_1 = 0.25$, $\lambda_2 = 2.25$ (anisotropic)



(c) $\lambda_1 = 2.25$, $\lambda_2 = 0.25$ (anisotropic)

Figure 11 – Prior samples of Gaussian processes with a squared exponential kernel, two inputs, and $\sigma_f = 1$. We may observe that the anisotropic cases have a direction with a noticeable variation compared to the other one. In contrast, the isotropic case does not have this property.

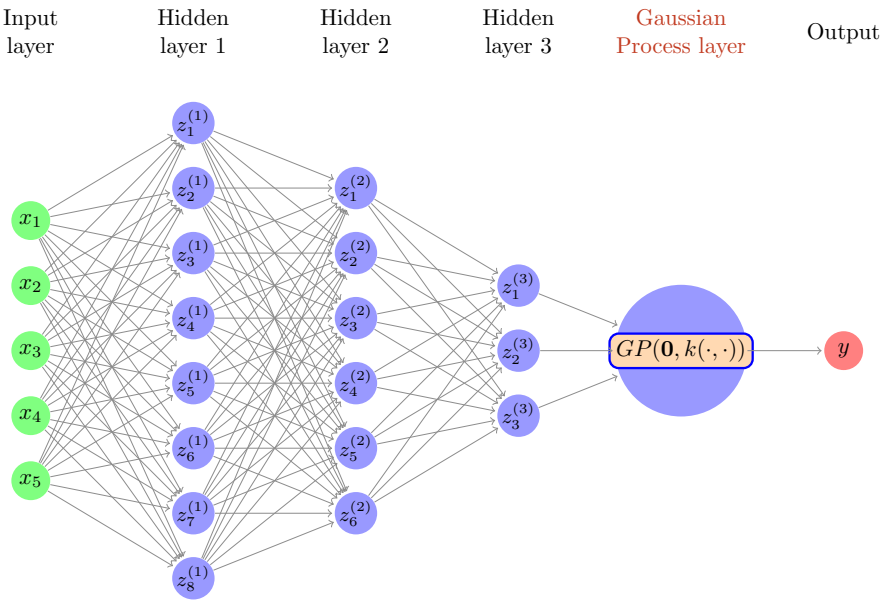


Figure 12 – Graphical representation of Deep kernel learning (DKL). In this particular case, the input has 5 dimensions and the neural network has 3 hidden layers of distinct sizes.

tuning process, as the marginal likelihood becomes more multimodal (specially in relation to the frequency hyperparameters). In (Wilson, 2014), some strategies of hyperparameter initialization are discussed so as to alleviate this adversity.

2.3.5.4 Deep kernel learning

Nowadays, it is difficult not to notice the use of deep learning techniques applied in regression problems. Deep learning is based on the use of deep neural networks (DNNs, also known as multilayer feedforward neural networks) (Goodfellow; Bengio; Courville, 2016; Schmidhuber, 2015), which are essentially nonlinear functions that work nicely in high dimensional function learning problems. A good introductory mathematical description of DNNs is outlined in (Higham; Higham, 2019). More recently, an approach that combines Gaussian processes with DNNs named Deep kernel learning (DKL) was proposed in (Wilson et al., 2016). The main idea is to transform the inputs using a neural network $g : \mathbb{R}^d \to \mathbb{R}^p$ and define a Gaussian process over the transformed inputs

$$h(\mathbf{x}) \sim GP(0,\ k(g(\mathbf{x}), g(\mathbf{x}'))). \tag{2.27}$$

The deep kernel is defined by a neural network and a base kernel. Any of the previously presented kernels may be used as a base kernel in DKL. Figure 12 contains a graphical representation of a DKL with a neural network with 3 hidden layers of distinct sizes.

In addition to the base kernel hyperparameter vector $\boldsymbol{\psi}$, a deep kernel is described by the neural network weight $\mathbf{w}$ and bias $\mathbf{b}$ vectors. All of these parameters are jointly learned

through the hyperparameter tuning process. We will denote the deep kernel parameter vector as $\boldsymbol{\gamma} = [\boldsymbol{\psi}, \mathbf{w}, \mathbf{b}]$ and the corresponding covariance matrix will be denoted as K_γ. Gradient based optimization is possible by using the chain rule to compute derivatives of the log marginal likelihood (Wilson et al., 2016) in the following way:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\psi}} = \frac{\partial \mathcal{L}}{\partial K_\gamma}\frac{\partial K_\gamma}{\partial \boldsymbol{\psi}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial K_\gamma}\frac{\partial K_\gamma}{\partial g(\mathbf{x};\mathbf{w},\mathbf{b})}\frac{\partial g(\mathbf{x};\mathbf{w},\mathbf{b})}{\partial \mathbf{w}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \frac{\partial \mathcal{L}}{\partial K_\gamma}\frac{\partial K_\gamma}{\partial g(\mathbf{x};\mathbf{w},\mathbf{b})}\frac{\partial g(\mathbf{x};\mathbf{w},\mathbf{b})}{\partial \mathbf{b}}. \tag{2.28}$$

It is noteworthy that the derivatives $\dfrac{\partial g(\mathbf{x};\mathbf{w},\mathbf{b})}{\partial \mathbf{w}}$ and $\dfrac{\partial g(\mathbf{x};\mathbf{w},\mathbf{b})}{\partial \mathbf{b}}$ are obtained using standard backpropagation algorithm.

One interesting property of DKLs is that the non stationary structure is captured by the neural network, while the stationary structure is incorporated by the base kernel (Wilson et al., 2016). Moreover, the use of a Gaussian process in the final layer maintains the advantages of a probabilistic inference compared to classical methods. On the other hand, the extensive number of parameters to be tuned, as well as the challenge of defining an adequate neural network (number of hidden layers and their sizes) for a certain regression problem, must be considered. While not exactly an issue, it is also worth mentioning that, unlike the base kernel hyperparameters, the neural network parameters may not have a clear interpretation (Higham; Higham, 2019).

2.3.6 Multi-output Gaussian Processes

So far we have presented some aspects of GPR as a tool of learning scalar functions $f : \mathbb{R}^d \to \mathbb{R}$. We may also be interested in learning vector functions $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^n$, a process named multi-task learning in the literature. In this work, we present two approaches to handle this type of regression problems.

2.3.6.1 Independent outputs

By assuming that the output dimensions are independent, the process is essentially the same as the scalar case. Denoting $\mathbf{y}^{(i)} = [y_i^{(1)}, \ldots, y_i^{(n_{\text{train}})}]^T$ as the training set associated with the ith output coordinate, we have the following posterior distribution for a new input $\mathbf{x}^*$:

$$h_i(\mathbf{x}^*)|\mathbf{y}^{(i)} \sim \mathcal{N}(K_*(K + \sigma_n^2 \mathrm{I})^{-1}\mathbf{y}^{(i)},\ K_{**} - K_*(K + \sigma_n^2 \mathrm{I})^{-1}K_*^T). \tag{2.29}$$

In order to take into account the influences on all output dimensions, the choice of the hyperparameter vector $\boldsymbol{\psi}$ is obtained by carrying out

$$\begin{aligned}\boldsymbol{\psi}^* &= \arg\max \ \ln\left[\prod_{i=1}^{n_{\text{tasks}}} (p(\mathbf{y}^{(i)}|\mathbf{x},\boldsymbol{\psi}))\right] \\ &= \arg\max \ \sum_{i=1}^{n_{\text{tasks}}} \ln(p(\mathbf{y}^{(i)}|\mathbf{x},\boldsymbol{\psi})),\end{aligned} \tag{2.30}$$

in which n_{tasks} is the number of output dimensions.

This technique is interesting due to its simplicity, although it can be inadequate in certain problems. In (Wang; Chen, 2015), for example, such inedaquacy is noticeable in learning a simple $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2$ in which both outputs are periodic.

2.3.6.2 Correlated outputs

The case where the output dimensions are correlated may be formulated using a special type of covariance function (Bonilla; Chai; Williams, 2008):

$$k([\mathbf{x},t],[\mathbf{x}',t']) = k_{\text{tasks}}(t,t') \times k_{\text{inputs}}(\mathbf{x},\mathbf{x}'). \tag{2.31}$$

in which $k_{\text{tasks}}(t,t')$ is the task covariance kernel. Once introducing task t and input $\mathbf{x}$ data, the product in (2.31) translates into the Kronecker product of the task K_{tt} and input $K_{\mathbf{xx}}$ covariance matrices. This strategy is a particular case of co-kriging (Bonilla; Chai; Williams, 2008).

Let us denote $\mathbf{Y} = [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n_{\text{tasks}})}]$ as the output matrix, $\text{vec}(\mathbf{Y})$ as a column vector obtained by vertical concatenation of all the matrix columns, and $\Sigma_{tt} \in \mathbb{R}^{n_{\text{tasks}} \times n_{\text{tasks}}}$ as a matrix representing the noise for all tasks. The log-likelihood is expressed by

$$\mathcal{L}(\boldsymbol{\psi}) = -\frac{1}{2}\text{vec}(\mathbf{Y})^T\mathbf{K}^{-1}\text{vec}(\mathbf{Y}) - \frac{1}{2}\ln(\det(\mathbf{K})) - \frac{n_{\text{train}}n_{\text{tasks}}}{2}\ln(2\pi), \tag{2.32}$$

in which

$$\mathbf{K} = \underbrace{K_{tt} \otimes K_{\mathbf{xx}}}_{\text{model covariance}} + \underbrace{\Sigma_{tt} \otimes \mathrm{I}_{\mathbf{xx}}}_{\text{noise covariance}}, \tag{2.33}$$

and the posterior distribution is

$$\mathbf{h}(\mathbf{x}^*|\text{vec}(\mathbf{Y})) \sim \mathcal{N}\left(\text{vec}(\mathbf{M}^*), \mathbf{V}^*\right), \tag{2.34}$$

in which

$$\mathbf{M}^* = (K_{tt} \otimes K_{\mathbf{x}^*\mathbf{x}})\mathbf{K}^{-1}\text{vec}(\mathbf{Y}), \tag{2.35}$$

$$\mathbf{V}^* = (K_{tt} \otimes K_{\mathbf{x}^*\mathbf{x}^*}) - (K_{tt} \otimes K_{\mathbf{x}^*\mathbf{x}})\mathbf{K}^{-1}(K_{tt} \otimes K^T_{\mathbf{x}^*\mathbf{x}}). \tag{2.36}$$

The naive approach for hyperparameter tuning and posterior inference involves constructing the matrix $\mathbf{K}$ (2.33) and finding its Cholesky factorization. This has $O(n^3_{\text{tasks}}n^3_{\text{inputs}})$

and $O(n_{\text{tasks}}^2 n_{\text{inputs}}^2)$ time and storage complexities, respectively (Trefethen; III, 1997). Since this may be prohibitive depending on the number of inputs and tasks, other strategies involving iterative linear solvers, approximation of the kernel input matrix $K_{\mathbf{xx}}$, and exploitation of Kronecker product structure may be utilized (Rakitsch et al., 2013; Pleiss et al., 2018; Titsias, 2009).

2.4 Discussion

This chapter presented the mathematical foundations, computational aspects, and applicability of Gaussian processes. In particular, we discussed how it is possible to use GPs to solve regression (or function learning) problems by:

1. Defining a GP, which involves the selection of an appropriate kernel function.
2. Finding the hyperparameters which maximize the marginal likelihood function. This process is essentially tuning the defined GP to match the training set.
3. Computing posterior mean and variance by solving linear systems with numerical linear algebra methods. The result is, not only the model outcome for a certain input, but also a measure of its uncertainty.

In the following chapter, we will perform Gaussian process regression in some examples, so as to explore and build intuition on the concepts presented in this chapter.

3 Exploring Gaussian Process Regression

Based on the previously discussed aspects of Gaussian processes, this chapter is dedicated to comparing the use of distinct kernels in some function learning problems. We aim to identify the benefits and disadvantages associated with a certain choice of the kernel function, considering both the hyperparameter tuning and prediction stages. Of note, one of the advanced used kernels is the following composition of kernels:

$$\begin{aligned} k_1(x, x') &= k_{\text{Matern}_1}^{(\nu=2.5)}(x, x') \times (k_{\text{Matern}_2}^{(\nu=2.5)}(x, x') + k_{\text{Per}}(x, x')), \\ k_2(x, x') &= k_{\text{SE}}(x, x') \times k_{\text{Dot}}(x, x'), \\ k_{\text{Composed}}(x, x') &= k_1(x, x') + k_2(x, x'). \end{aligned} \tag{3.1}$$

In (Duvenaud, 2014), a similar kernel to (3.1) is employed in regression experiments. Also, in all of the following experiments, the Deep kernels have a DNN with size $d-100-50-5-d$, in which d is the input dimension, with the base kernels being the squared exponential and spectral mixture kernels. In the literature, there are objective criteria for comparing distinct GP and selecting which one is adequate to a certain problem (Gorbach et al., 2017), although this chapter main goal is to introduce the topic of Gaussian process regression and build intuition.

We also note that the hyperparameter σ_n, related to the hypothesized noise in the empirical data, is obtained along with the kernel hyperparemeters in the tuning process. It is also possible to specify its value in order to use it as a regularization parameter or if it the level of noise is known in advance, although such approaches are not explored in this work.

The optimization method of choice was the Adaptive moment estimation (Adam) (Kingma; Ba, 2014; Ruder, 2016; Goodfellow; Bengio; Courville, 2016), which is the default method for hyperparameter tuning in the GPyTorch library. This optimizer is also vastly used by the machine learning community, specially in the context of DNN. It is worth noting that, depending on kernel complexity and problem dimensionality, a larger number of iterations were executed in this procedure. In the following experiments, a total of 8 restarts were executed in all cases. This value was chosen considering a good balance between computational time and regression performance. Also, in order to better evaluate the effects of distinct kernel choices, we opted for the same optimization procedure setup.

We perfomed a Gaussian process regression over six function learning problems, each one with a different structure. The kernels used in all these examples are summarized in Table 1 in which we also indicate their corresponding hyperparameters. The results were obtained using Python 3 (Rossum; Drake, 2009) with GPyTorch (Gardner et al.,

2018), a powerful and easy to use Gaussian process for machine learning library (for more details, see <https://gpytorch.ai>). The source code is available at <https://github.com/jvitordeoliveira96/GPRExamples>.

Kernel	Definition	Hyperparameters
Squared Exponential	$k_{\text{SE}}(x,x') = {\sigma_f}^2 \exp\left(-\frac{1}{2}\frac{r^2}{{\lambda_f}^2}\right)$	$\sigma_f, \lambda_f, \sigma_n$
Matérn	$k^{(\nu)}_{\text{Matern}}(x,x') = {\sigma_f}^2 \exp\left(-\frac{\sqrt{2\nu}r}{\lambda_f}\right)\frac{s!}{(2s)!}\sum_{j=0}^{s}\frac{(s+j)!}{j!(s-j)!}\left(\frac{\sqrt{8\nu}r}{\lambda_f}\right)^{s-j}$	$\sigma_f, \lambda_f, \sigma_n$
	$\nu = 0.5 + s$	
Dot	$k_{\text{Dot}}(x,x') = {\sigma_f}^2(x-\sigma_0)(x'-\sigma_0)$	$\sigma_f, \sigma_0, \sigma_n$
Periodic	$k_{\text{Per}}(x,x') = {\sigma_f}^2 \exp\left(-\frac{2}{{\lambda_f}^2}\sin^2\left(\frac{\pi r}{p}\right)\right)$	$\sigma_f, \lambda_f, p, \sigma_n$
Composed	$k_{\text{Composed}}(x,x') = k_1(x,x') + k_2(x,x')$	$\sigma_f^{(k_1)}, \lambda_f^{(\text{Matern}_1)}, \lambda_f^{(\text{Matern}_2)},$
	$k_1(x,x') = k^{(\nu=2.5)}_{\text{Matern}_1}(x,x') \times (k^{(\nu=2.5)}_{\text{Matern}_2}(x,x') + k_{\text{Per}}(x,x'))$	$\lambda_f^{(Per)}, p, \sigma_f^{(k_2)},$
	$k_2(x,x') = k_{\text{SE}}(x,x') \times k_{\text{Dot}}(x,x')$	$\lambda_f^{(\text{SE})}, \sigma_0, \sigma_n$
Spectral Mixture	$k_{\text{SM}}(x,x') = \sum_{j=1}^{q}\sigma_j^2\cos(2\pi\mu_j(x-x'))\exp(-2\pi^2(x-x')^2 v_j)$	$\sigma_j, \mu_j, v_j, \sigma_n$
	q Gaussians	
DKL(SE)	$k^{\text{SE}}_{\text{DKL}}(x,x') = k_{\text{SE}}(g(x), g(x'))$	$\sigma_f, \lambda_f, \sigma_n,$
	DNN $g(\cdot)$ with size $1-100-50-5-1$	$\mathbf{w}, \mathbf{b}$
DKL(SM)	$k^{\text{SM}}_{\text{DKL}}(x,x') = k_{\text{SM}}(g(x), g(x'))$	$\sigma_j, \mu_j, v_j, \sigma_n,$
	DNN $g(\cdot)$ with size $1-100-50-5-1$	$\mathbf{w}, \mathbf{b}$

Table 1 – Kernels used in the following examples, assuming unidimensional inputs, for which we use $r = |x - x'|$

3.1 $f_1(x) = \sin(4x)$

The first function of choice is a periodic scalar unidimensional function $f_1 : \mathbb{R} \to \mathbb{R}$. We used 35 evenly spaced training $x \in [-2, 2]$ and computed the estimates for $x^* \in [-4, 4]$. In order to complicate this regression problem, a noise $\epsilon \sim \mathcal{N}(0, 0.2^2)$ was added to the outputs.

A Gaussian process regression with a squared exponential kernel was able to learn the function for $x^* \in [-2, 2]$, as seen by the nice interpolation and low levels of uncertainty for inputs in this range in Figure 13. On the other hand, incorrect estimates and considerable higher levels of uncertainty corresponding to inputs outside the training set range indicate that a GPR with this kernel is incapable of performing reliable extrapolation. Figure 14 shows that the same occurs when Matérn kernel is used with $\nu = 2.5$.

It was previosly stated that a Gaussian process with a dot product kernel is composed of a distribution of affine functions. Considering that $f_1(x)$ is a sinusoidal function, the regression in Figure 15 is inadequate. The mean function is the best line (in the sense of having the lowest squared error), while the confidence region encompasses the observed data as noise.

Periodic functions are appropriately represented by a Gaussian process with a Periodic kernel. This is depiced in Figure 16: even in the presence of noisy data, the mean function learns (has great interpolation and extrapolation capabilities) the true function $f_1(x)$.

Figure 17 indicates that using the advanced kernel k_{Composed} results in a GPR with a good interpolation capability, although there are noticeable differences between the true function and the mean function when extrapolation is performed. Compared to the previous kernels, there is also a larger variance in the estimates in the training set range.

The greater difficulty in using the spectral mixture kernel is the high number of bad local optima in the hyperparameter tuning procedure. Without restarts, this step may fail, returning an unusable prediction model as shown in Figure 18. By using restarts, the chances of this occurring are reduced, and we are able to obtain GPR with better estimates. With a single Gaussian, it is shown in Figure 19 that it is already possible to learn the true function. As the number of Gaussians increases, small "artifacts" (small wiggles near the training points, suggesting an overfitting issue) are visible in Figures 20 and 21, obtained with 2 and 4 Gaussians, respectively, due to the increased model variance that surpasses what is necessary to represent the true function.

Lastly, GPR with deep kernels (DKL with Square Exponential - DLK(SE) and DKL with Spectral Mixture - DKL(SM)) were not capable of performing a better interpolation or extrapolation compared to the previous kernels, as noticed in Figures 22 and 23. There are multiple factors that could explain this performance. First of all, the considerable larger number of parameters worsens the issue of obtaining bad local optima, making the use of restarts essentially mandatory. Even with the use of multiple restarts, it is still possible that the optimization procedure fails. The increased model variance is also a plausible reason, considering that the "artifacts" near the training points are noticeable in both cases, and extrapolation is incorrect using both of the base kernels.

Regarding the hyperparameter tuning process, the same learning rate and number of iterations were used. For the advanced kernel k_{Composed}, a larger number of iterations were necessary. The spectral mixture and deep kernels required higher number of iterations as well as lower learning rates. The values are reported in Table 2. In general, a larger number of iterations with lower learning rates are necessary due to increased number of parameters. This also happens when the likelihood functions are highly sensitive to variations in the hyperparameters.

	α (learning rate)	Iterations
Basic kernels	0.1	450
Composed kernel	0.1	900
Spectral mixture kernel	0.06	1200
Deep kernel (SE)	0.00066	2100
Deep kernel (SM)	0.00066	8000

Table 2 – Hyperparameter tuning process learning rate and iterations values for the $f_1(x)$, $f_2(x)$, and $f_4(x)$ function learning problems. We may notice that more complex kernels require more computations and are more sensitive to variations in the hyperparameters.

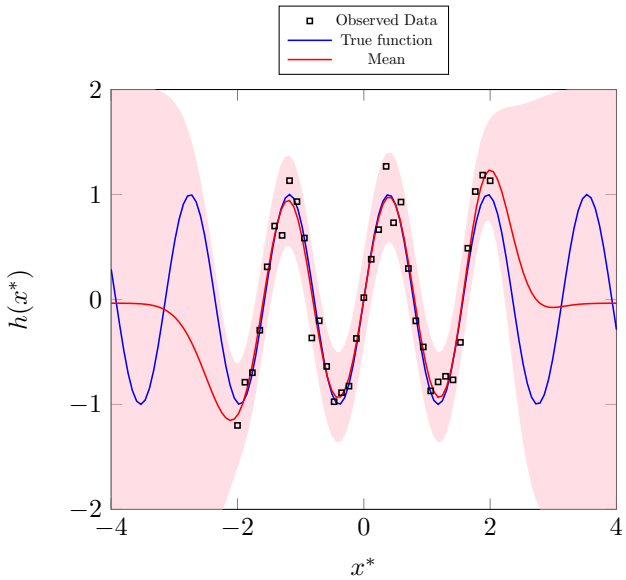


Figure 13 – Gaussian process regression with a squared exponential kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 1.025$, $\lambda_f = 0.458$, and $\sigma_n = 0.035$. We may notice a nice interpolation for $x^* \in [-2, 2]$, since the posterior mean function is in accordance with $f_1(x)$ and there is low variance in the estimates. For inputs outside the training set range, the prediction model is incapable of performing reliable extrapolation, as the mean function reverts to the prior $m(x) = 0$ and considerable higher variance in the estimates is present.

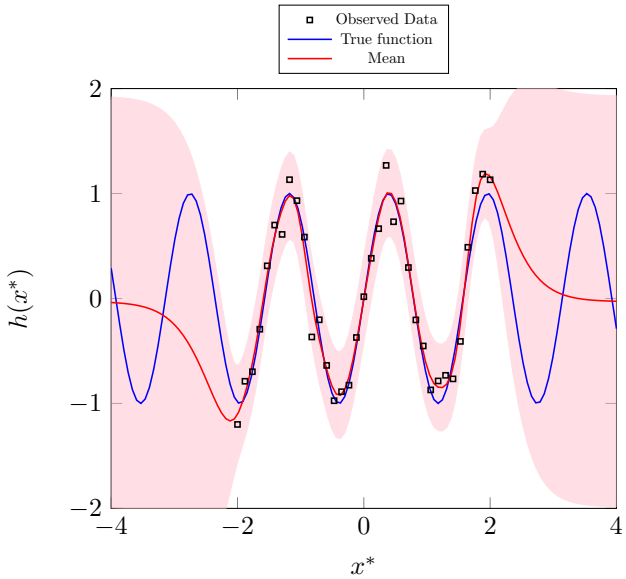


Figure 14 – Gaussian process regression with a Matérn ($\nu = 2.5$) kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 0.93$, $\lambda_f = 0.514$, and $\sigma_n = 0.03$. Similar to the GPR with a squared exponential kernel, the prediction model learns the true function for inputs in the training set range, however it is not able to perform good extrapolation.

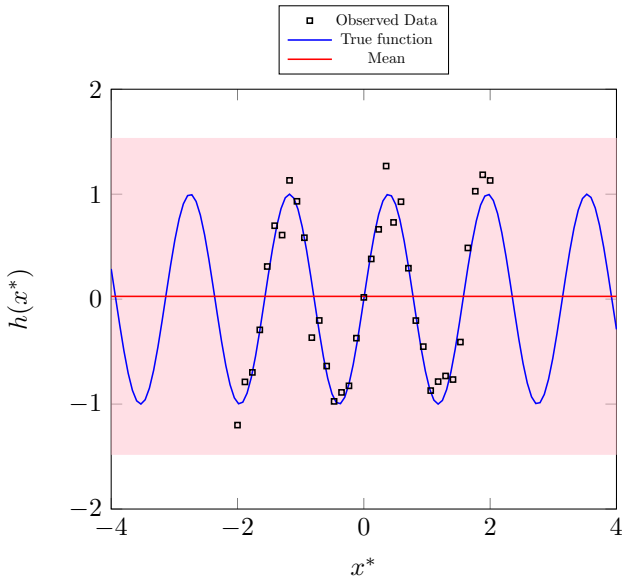


Figure 15 – Gaussian process regression with a dot product kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 1 \times 10^{-15}$, $\sigma_0 = -305916.54$, $\sigma_n = 0.565$. The prediction model is inadequate, considering the true function nonlinearity. The mean function is the line associated with the lowest squared error, while the uncertainty around it is accounted as noise.

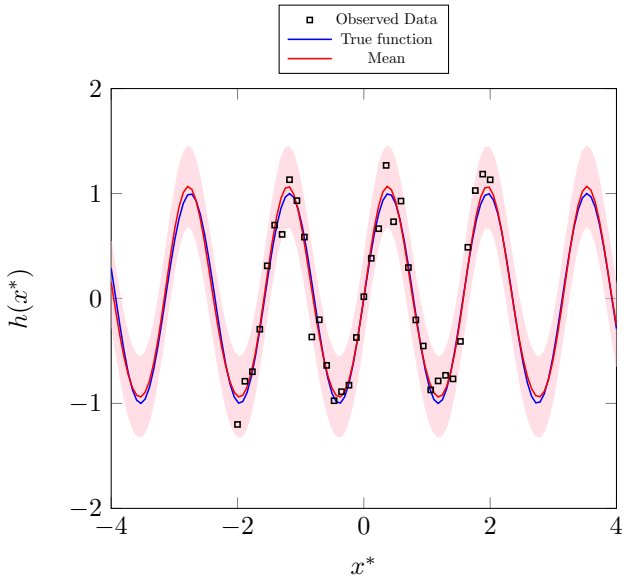


Figure 16 – Gaussian process regression with a Periodic kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 2.59$, $\lambda_f = 9.88$, $p = 1.58$, $\sigma_n = 0.0326$. Even in the presence of noise in the training set, the GPR was able to learn the true function, as the mean function practically coincides with the true function on the whole domain and the variance levels are similar for both interpolation and extrapolation ranges.

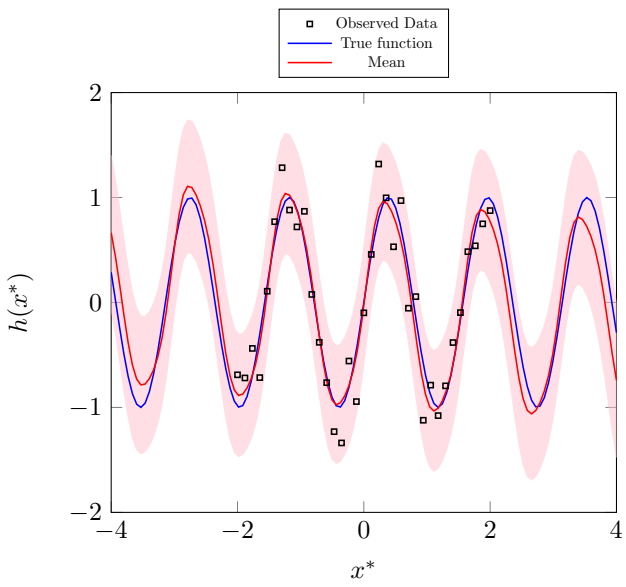


Figure 17 – Gaussian process regression with the Composed kernel defined in (3.1). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f^{(k_1)} = 0.5311$, $\lambda_f^{(\text{Matern}_1)} = 16.73$, $\lambda_f^{(\text{Matern}_2)} = 15$, $\lambda_f^{(\text{Per})} = 15$, $p = 1.616$, $\sigma_f^{(k_2)} = 3 \times 10^{-5}$, $\lambda_f^{(\text{SE})} = 1.39$, $\sigma_0 = 1.7$, $\sigma_n = 0.067$. In both interpolation and extrapolation ranges, there are some noticeable differences between the mean function and the estimates. Overall, the performance is worse when compared to utilizing the Periodic kernel. This suggests that this kernel function is more complex than what is necessary for this particular regression problem.

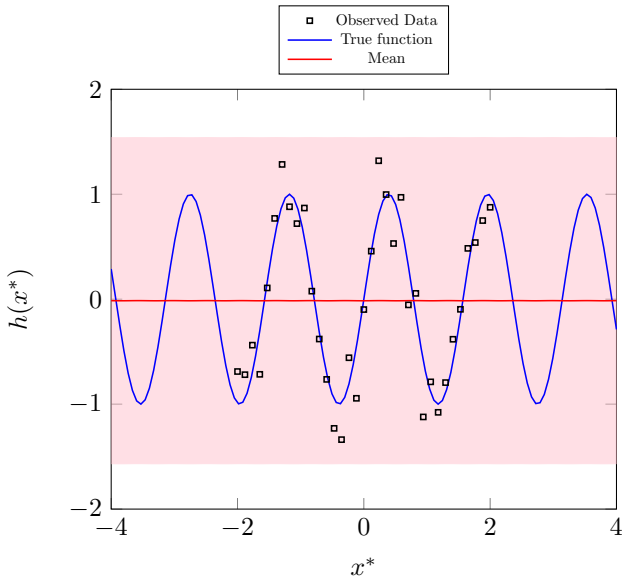


Figure 18 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian, failed attempt). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 1.83$, $v = 0.016$, $\sigma = 2 \times 10^{-4}$, $\sigma_n = 0.6$. The optimizer was unable to find a good local optima due to a bad random initial guess.

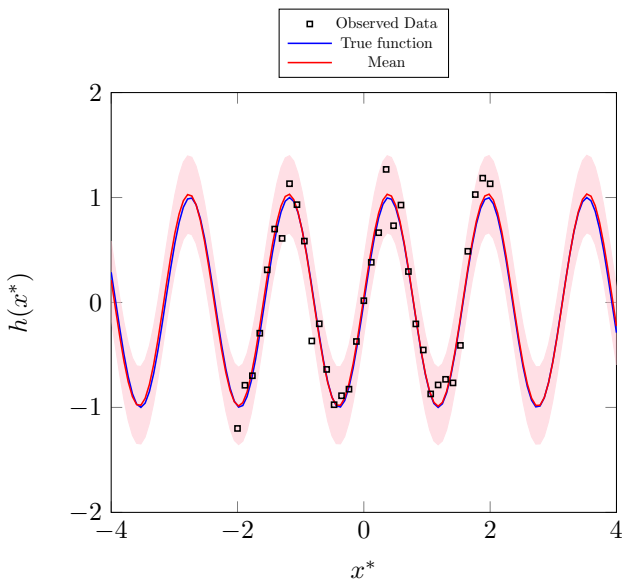


Figure 19 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian) . The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 0.634$, $v = 0.0011$, $\sigma = 0.507$, $\sigma_n = 0.0320$. Using restarts, a GPR with predictive power is obtained. For these hyperparameter values, the exponential term in (2.26) is approximately 1, indicating that the periodic component was comparably more prominent. The mean function and the levels of variances are similar to those obtained using the Periodic kernel.

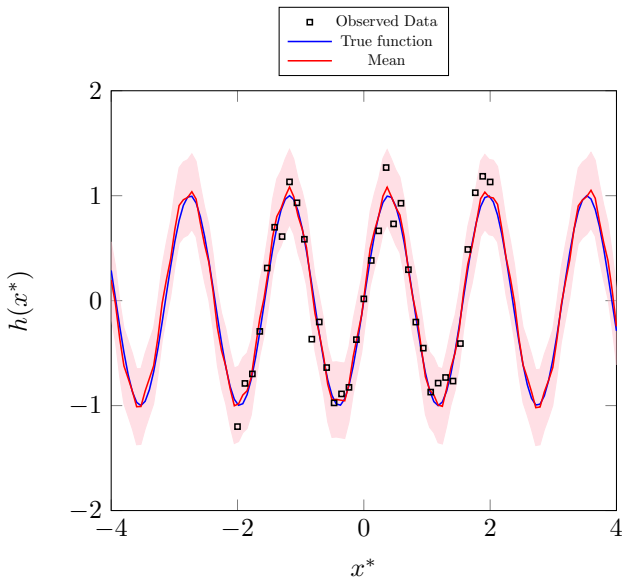


Figure 20 – Gaussian process regression with a Spectral mixture kernel (2 Gaussians) . The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.633, 3.959]$, $V = [0.0015, 0.0073]$, $\boldsymbol{\sigma} = [0.507, 0.0023]$, $\sigma_n = 0.0296$. The performance is very similar to the single Gaussian case, although small "artifacts" are noticeable near the training points, in which the mean functions appears to be trying to "overfit" the data.

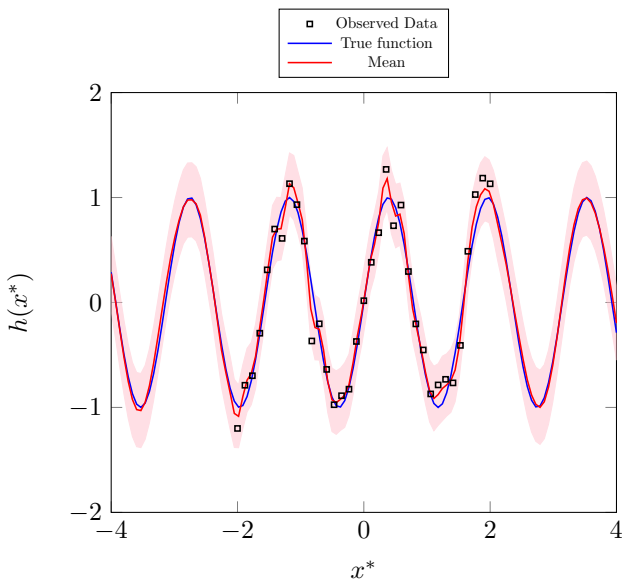


Figure 21 – Gaussian process regression with a Spectral mixture kernel (4 Gaussians) . The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [2.29, 0.6346, 1.388, 4.145]$, $V = [8.353 \times 10^{-1}, 6.1015 \times 10^{-4}, 5.429 \times 10^{-3}, 3.1664 \times 10^{-1}]$, $\boldsymbol{\sigma} = [0.0117, 0.5043, 0.0024, 0.0033]$, $\sigma_n = 0.014$. The presence of "artifacts" due to a "overfitting" tendency is worsened by the increased number of Gaussians.

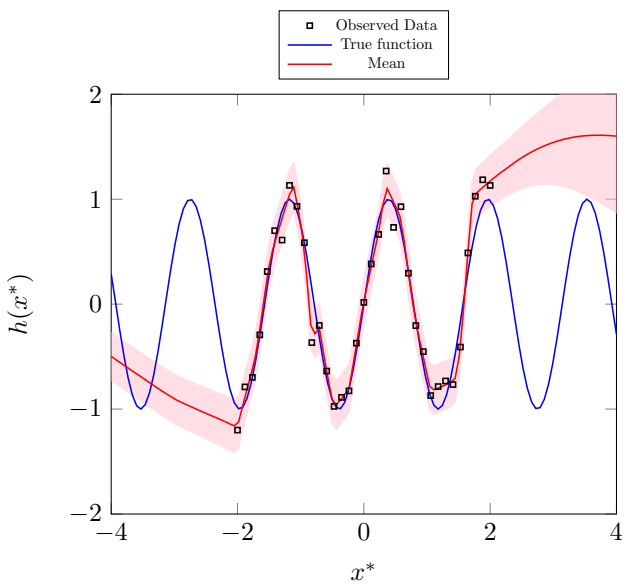


Figure 22 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. Similar to using a spectral mixture kernel with 4 Gaussians, "overfitting" appears to occur due to the "artifacts" in the prediction at the training points. As in the case of using a squared exponential kernel without a DNN, the GPR was not capable of performing a reasonable extrapolation.

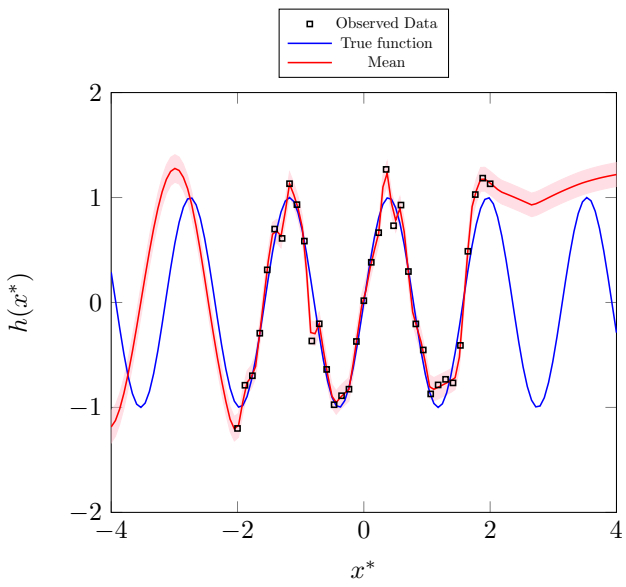


Figure 23 – Gaussian process regression with a deep kernel, using the spectral mixture (with 4 Gaussians) as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The used of DNN did not introduce any gain compared to only using the spectral mixture. On the contrary, "overfitting" appears to be even more prominent and extrapolation quality is considerable inferior.

3.2 $f_2(x) = \sin(4x) - 0.12x^3$

The second function is also a unidimensional function $f_2 : \mathbb{R} \to \mathbb{R}$, although it is not a periodic function as f_1. For inputs $x \in [-2, 2]$, the sinusoidal term is dominant, while the polynomial term becomes dominant for inputs outside this interval. We used 35 evenly spaced training $x \in [-4.5, 4.5]$ and computed the estimates for $x^* \in [-6, 6]$. The same type of noise $\epsilon \sim \mathcal{N}(0, 0.2^2)$ was added to the outputs.

Figure 24 indicates that, once again, a GPR with a squared exponential kernel has a nice interpolation with low levels of uncertainty in the training input range, although it reverts to the prior $m(x) = 0$ with higher levels of uncertainty for extrapolated estimates. The same characteristics are present when a Matérn kernel with $\nu = 2.5$ is used, as depicted in Figure 25. Another similarity is noticed with the experiments on f_1 in Figure 26: the lacking performance when a dot product kernel is chosen, by reason of f_2 clearly not being an affine function.

The fact that f_2 is not a periodic function now shows the limitations of using a Periodic kernel. Unlike the successful regression in the task of learning f_1, the GPR fails in learning f_2, with the mean function curve in Figure 27 having an incorrect "electrocardiogram" shape. The reason behind this bad performance is the same as in the dot product case: the prediction model prior distribution does not have functions in

accordance with the function being learned. On a lesser extent, the spectral mixture kernel with a single Gaussian is not capable of learning the true function in its whole domain, with Figure 29 depicting results similar to those obtained using the squared exponential. In Figures 30 and 31, we may observe that with a higher number of Gaussians, extrapolation quality is slightly superior, although the GPR is not yet capable of learning the f_2 for $x^* \in [-\infty, +\infty]$.

The function f_2 is a sum of a periodic and a polynomial function, and for this reason, the associated kernel function has to incorporate a corresponding similarity measure in the input space. A possibility of constructing such kernel function is by adding and multiplying basic kernels. When the advanced kernel k_{Composed} is employed, Figure 28 expresses that the GPR is efficient in performing prediction at both interpolation and extrapolation ranges.

Once again, the use of deep kernels did not improve the GPR compared to using only a base kernel. Comparing the results in Figures 32 and 24, the GPR with deep kernel using a squared exponential as the base kernel has a performance close to only using the squared exponential kernel (without a DNN). Similarly, employing a deep kernel with a spectral mixture kernel, Figure 33 shows that the estimates in the extrapolation range are compromised compared to the results in Figure 31.

It is worth mentioning that the same learning rate and iterations values in Table 2 are used in this function learning problem.

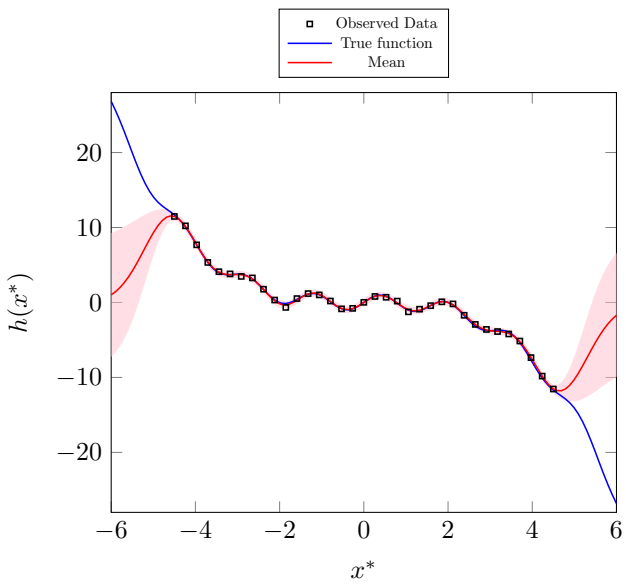


Figure 24 – Gaussian process regression with a squared exponential kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 17.96$, $\lambda_f = 0.727$, $\sigma_n = 0.0309$. As expected, the GPR computes estimates in accordance with the true function in the training set range, whereas the extrapolated estimates are incorrect and have larger variance.

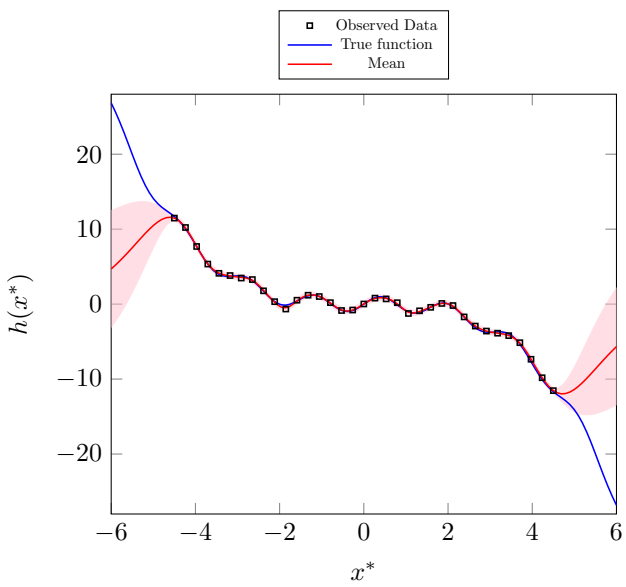


Figure 25 – Gaussian process regression with a Matérn ($\nu = 2.5$) kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 20.268$, $\lambda_f = 1.21$, $\sigma_n = 0.02$. Once again, the interpolated and extrapolated estimates are pretty much the same as the ones obtained with a squared exponential kernel.

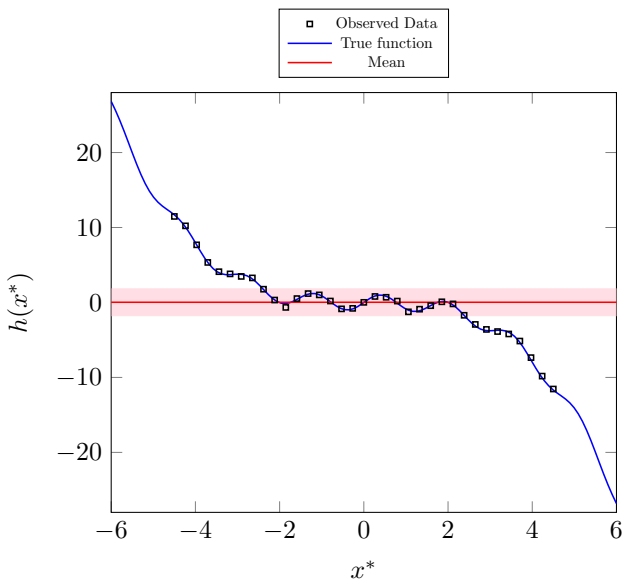


Figure 26 – Gaussian process regression with a dot product kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 9 \times 10^{-11}$, $\sigma_0 = -2779$, $\sigma_n = 0.799$. Considering that the function being learned is not an affine function, the prediction model is inadequate, with the posterior mean not even changing from the initial prior $m(x) = 0$.

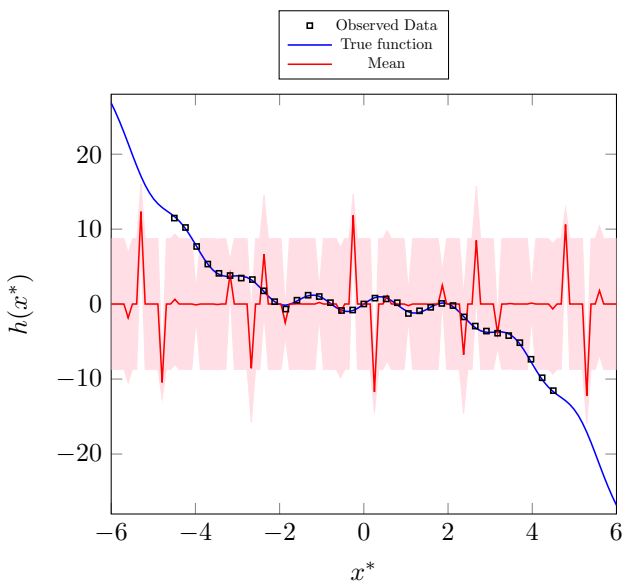


Figure 27 – Gaussian process regression with a Periodic kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 18.18$, $\lambda_f = 0.012$, $p = 0.265$, $\sigma_n = 0.795$. Since the function being learned is not periodic, the GPR is incapable of learning the true function, with the prediction for both interpolation and extrapolation ranges being an incorrect "electrocardiogram" shaped curve.

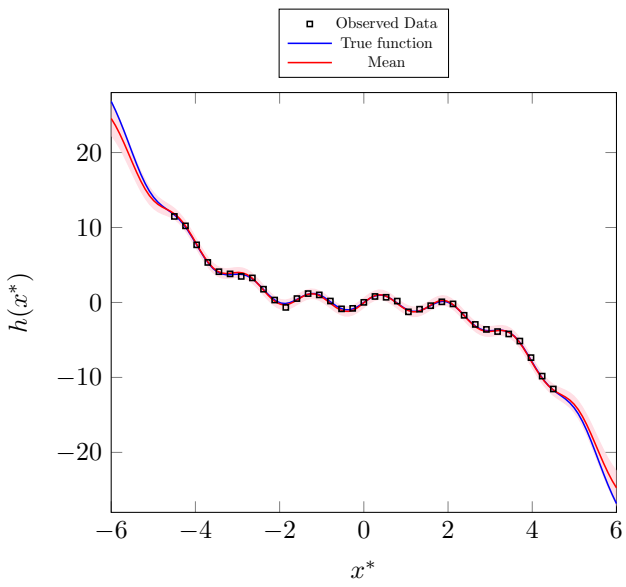


Figure 28 – Gaussian process regression with the Composed kernel defined in (3.1). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f^{(k_1)} = 6.67$, $\lambda_f^{(\text{Matern}_1)} = 29.6$, $\lambda_f^{(\text{Matern}_2)} = 12.7$, $\lambda_f^{(\text{Per})} = 20.25$, $p = 1.57$, $\sigma_f^{(k_2)} = 9 \times 10^{-8}$, $\lambda_f^{(\text{SE})} = 7.94$, $\sigma_0 = -276655$, $\sigma_n = 0.085$. The GPR estimates are consonant with the true function for $x^* \in [-6, 6]$, with low levels of uncertainty, outperforming the GPRs using basic kernels. This kernel composed structure is consistent with f_2, a function with a periodic and a polynomial component.

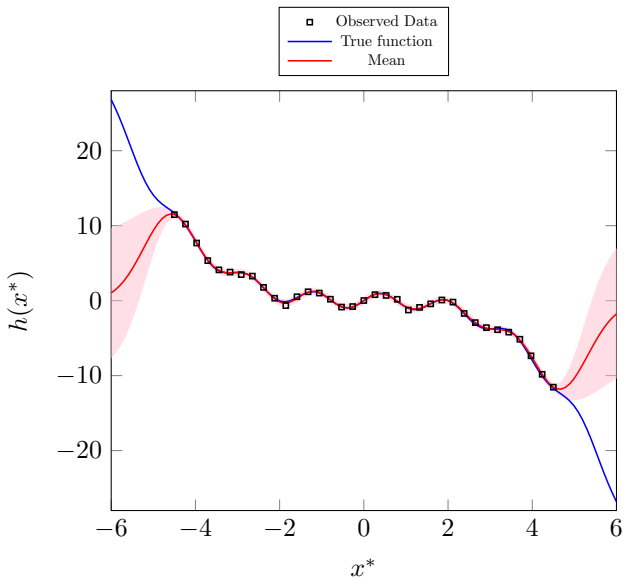


Figure 29 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 0.058$, $v = 0.208$, $\sigma = 20.035$, $\sigma_n = 0.031$. We may notice that the GPR is not capable of learning the true function for $x^* \in [-\infty, +\infty]$, having a similar performance to the squared exponential case.

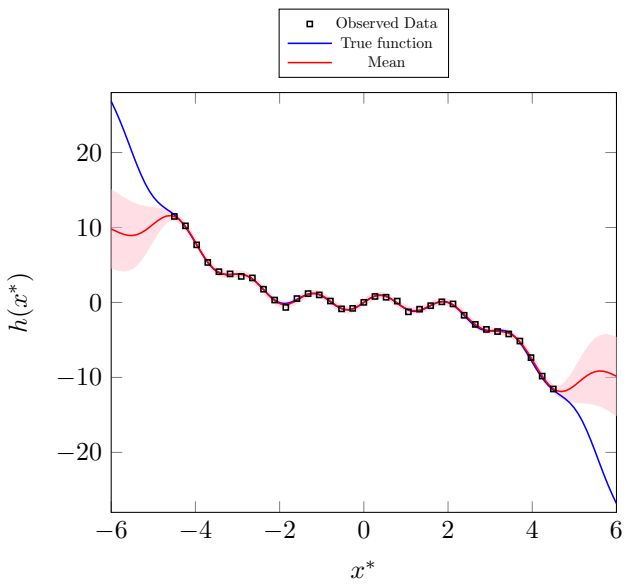


Figure 30 – Gaussian process regression with a Spectral mixture (2 Gaussians). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.075, 0.452]$, $V = [0.033, 0.1]$, $\boldsymbol{\sigma} = [22.4, 1.834]$, $\sigma_n = 0.033$. Compared to the case with a single Gaussian, the GPR is still not capable of performing an adequate extrapolation, although slightly superior estimates are obtained.

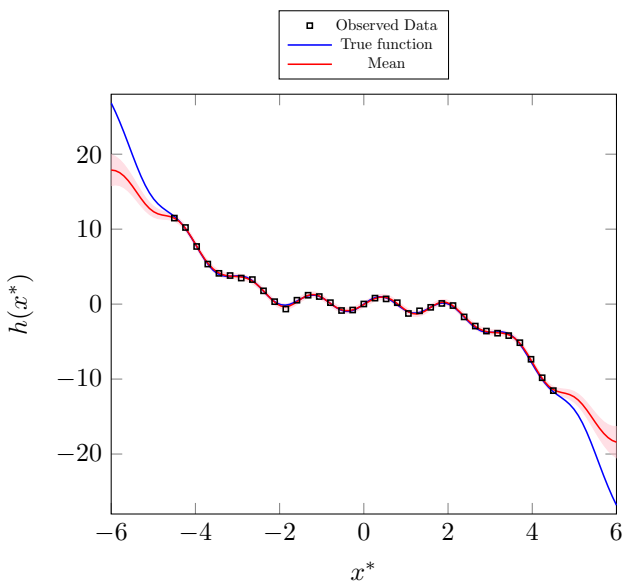


Figure 31 – Gaussian process regression with a Spectral mixture (4 Gaussians). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.635, 0.04, 0.96, 0.132]$, $V = [0.0005, 0.0007, 0.0014, 0.022]$, $\boldsymbol{\sigma} = [5.13 \times 10^{-1}, 2.063 \times 10^{1}, 5.78 \times 10^{-3}, 1.3845 \times 10^{1}]$, $\sigma_n = 0.024$. The increased number of Gaussians resulted in a GPR capable of computing noticeable better estimates in the extrapolation range.

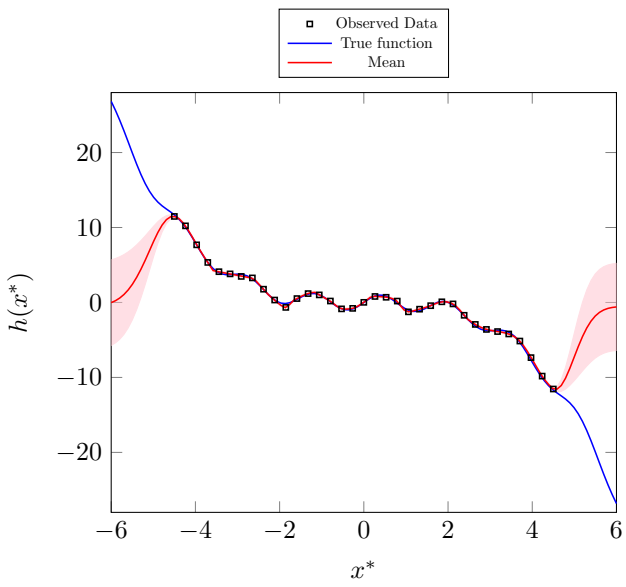


Figure 32 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. Observing the results, the introduction of a DNN did not improve the performance of the GPR with a squared exponential kernel, as the estimates in both interpolation and extrapolation ranges did not have considerable differences.

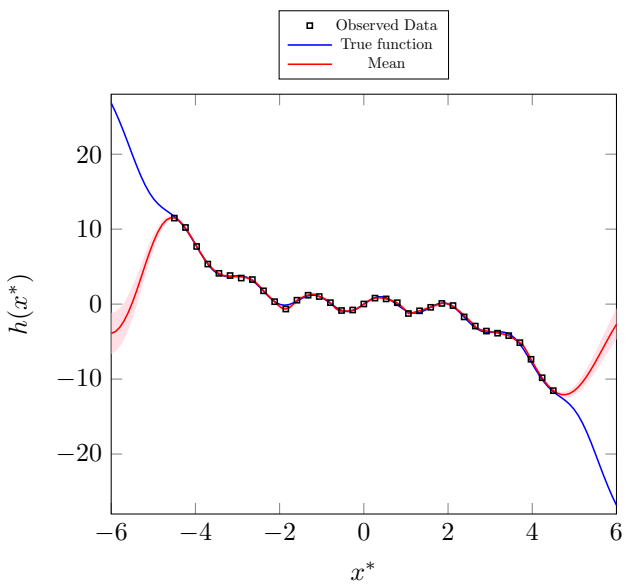


Figure 33 – Gaussian process regression with a deep kernel, using the spectral mixture (with 4 Gaussians) as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. We are able to notice that this more complex GPR computed estimates on the extrapolation range of inferior quality compared to the ones obtained using only the base kernel.

3.3 f_3 (step function)

Both previous functions were continuous infinitely differentiable for all possible values x in the input space (these are referred as C^∞ functions in the literature). We may also investigate how a GPR performs when the function being learned has discontinuities, since this violates some of the assumptions for certain kernels such as the squared exponential. The step function

$$f_3(x) = \begin{cases} -2, & x < -2 \\ 0, & -2 \leq x < -1 \\ 2, & -1 \leq x < 1 \\ 1, & 1 \leq x < 2 \\ -2, & x \geq 2 \end{cases}, \tag{3.2}$$

was chosen for the following experiments.

As mentioned previously, the squared exponential kernel is adequate for learning smooth functions. The results in Figure 34 depicts the limitation of not being capable of learning a function with jump discontinuities. The GPR prediction curve is a smooth function which approximates the true function in the interpolation range. Interestingly, the lack of capability of passing through the training set data is highlighted by the confidence region, an effect of the GPR attempting to adjust to the jump discontinuities. Not surprisingly, estimates in the extrapolation range are incorrect and have a increasing level of uncertainty. Using the Matérn kernel with a weaker smoothness assumption, in other words, using $\nu = 0.5$, results in better estimates in the interpolation as presented in Figure 35.

Unlike the previous regressions, the use of the composed kernel k_{Composed} results in a GPR with a lacking prediction performance. The prediction curve in Figure 36 has a considerable amount of "wiggles" in the interpolation range (suggesting an overfitting issue), while the extrapolation predictions appear to be a noisy version of the obtained one using the squared exponential kernel.

The results for the spectral mixture kernel case are particularly interesting, since its Fourier theory foundations are noticed in Figures 37, 38 and 39. By increasing the number of Gaussians, the GPR is able to better capture the true function behavior in the training set range. The mean function prediction at the jump discontinuities is the average between the true function values in either side of the jump, which is a known result from Fourier analysis (Pinsky, 2008). Even though prediction quality in the interpolation range increases, the same does not occur for extrapolation estimates.

Lastly, the results associated with the use of deep kernels showed an impressive performance for both interpolation and extrapolation ranges. The results for using squared exponential and spectral mixture as base kernels are depicted in Figures 40 and 41, respectively, the former appearing to be slightly superior. These results indicate that kernels of these types are appropriate for functions with sharp discontinuities, which is the case of one of function learning problems presented in (Wilson et al., 2016).

The learning rate and iterations values were the same as in Table 2, with the exception of the composed kernel. The higher number of iterations and lower learning rate still resulted in an inadequate GPR with this kernel.

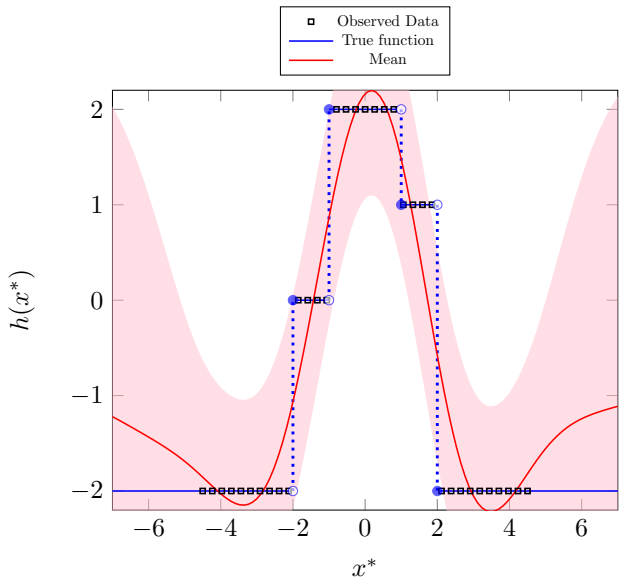


Figure 34 – Gaussian process regression with a squared exponential kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 2.836$, $\lambda_f = 1.7$, and $\sigma_n = 0.257$. The resulting mean function is a smooth function which approximates the true function in the interpolation range. Considering the true function discontinuities and this kernel strong smoothness assumption, the GPR does not coincide with the true function even for the training set, compensating it with a large confidence region.

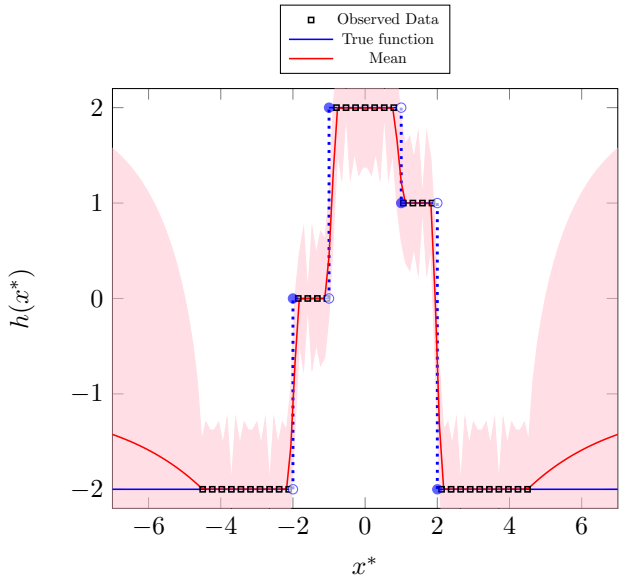


Figure 35 – Gaussian process regression with a Matérn kernel ($\nu = 0.5$). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 2.6$, $\lambda_f = 2.5$, and $\sigma_n = 0.0005$. The prediction model is very similar to the squared exponential case in the extrapolation range, although it is superior in terms of interpolation.

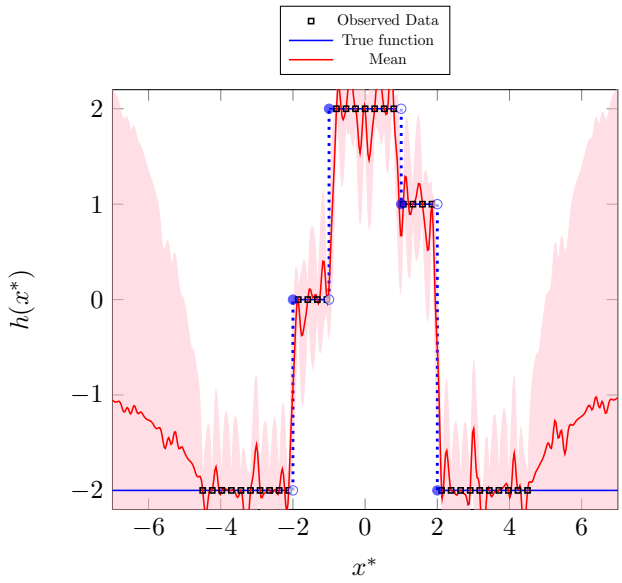


Figure 36 – Gaussian process regression with the Composed kernel defined in (3.1). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f^{(k_1)} = 0.175$, $\lambda_f^{(\text{Matern}_1)} = 2.44$, $\lambda_f^{(\text{Matern}_2)} = 1.13$, $\lambda_f^{(\text{Per})} = 0.12$, $p = 1.57$, $\sigma_f^{(k_2)} = 4.9 \times 10^{-10}$, $\lambda_f^{(\text{SE})} = 1.7$, $\sigma_0 = -73904.5$, $\sigma_n = 0.014$. The posterior mean function is a noisy version of the ones obtained with the squared exponential and Matérn kernels, with noticeable overfitting in the interpolation range.

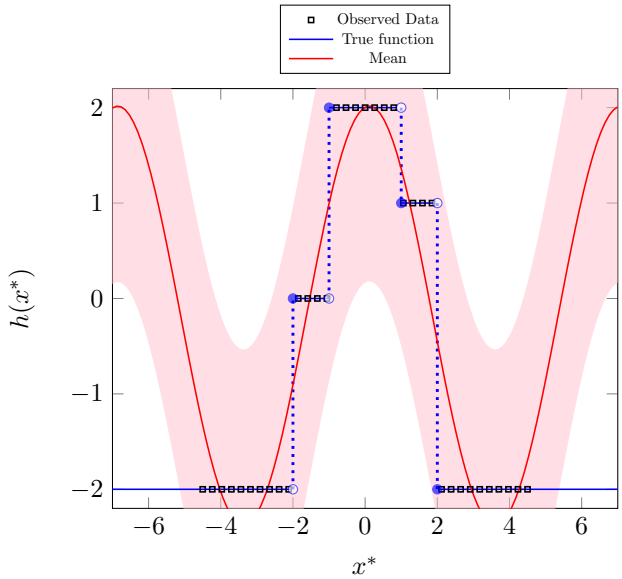


Figure 37 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 0.143$, $v = 0.001$, $\sigma = 2.417$, $\sigma_n = 0.8$. With only a single Gaussian, the posterior mean function predicts similar values to the squared exponential kernel in the interpolation range, although the extrapolation has a periodic profile.

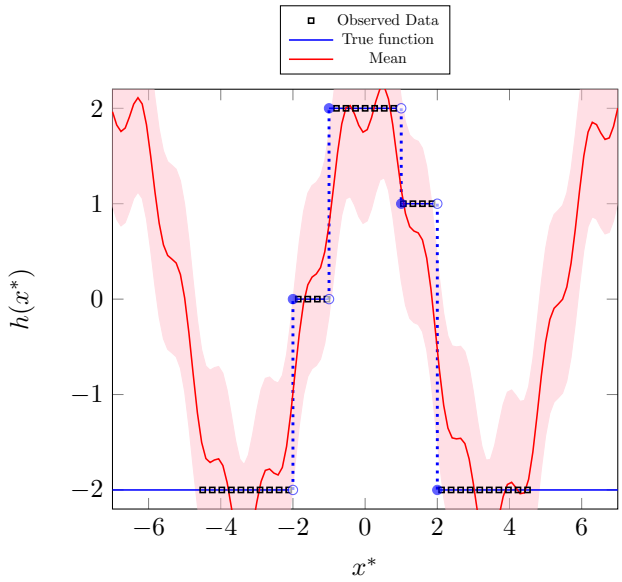


Figure 38 – Gaussian process regression with a Spectral mixture kernel (2 Gaussians). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.145, 0.888]$, $V = [0.0024, 0.0028]$, $\boldsymbol{\sigma} = [1.9711, 0.0583]$, $\sigma_n = 0.208$. The increased number of gaussians highlights the Fourier spectral theory foundations of this kernel. An additional periodic component is present, allowing the prediction model to represent the training set data slightly better.

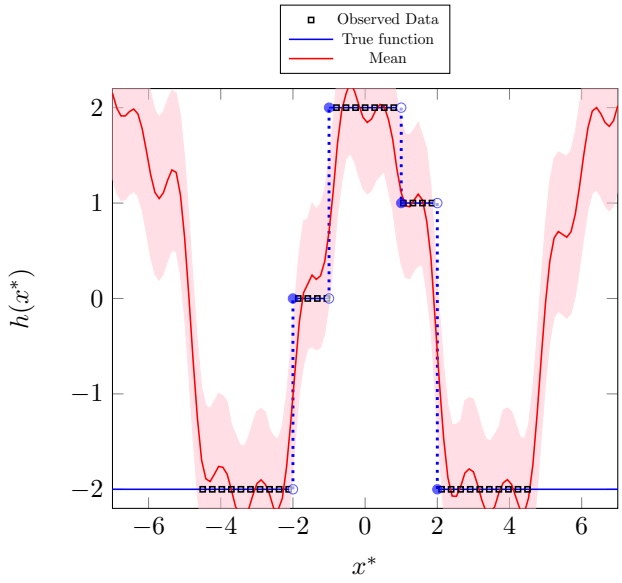


Figure 39 – Gaussian process regression with a Spectral mixture kernel (4 Gaussians). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.8774, 0.5233, 0.1483, 0.494]$, $V = [3.05\times10^{-4}, 4.601\times10^{-2}, 2.494\times10^{-4}, 9.123]$, $\boldsymbol{\sigma} = [0.0713, 0.118, 2.549, 0.0363]$, $\sigma_n = 0.079$. An even higher number of Gaussians has a corresponding mean function with superior predictions and a lower confidence region in the interpolation range. It is worth mentioning that the extrapolation range predictions remains not consonant with the true function.

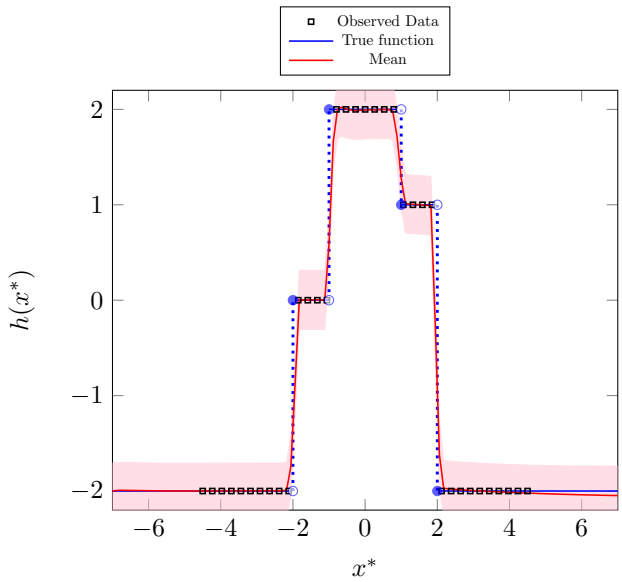


Figure 40 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The mean function successfully captures the true function in both interpolation and extrapolation ranges, indicating the importance of the inclusion of a DNN in this particular function learning problem.

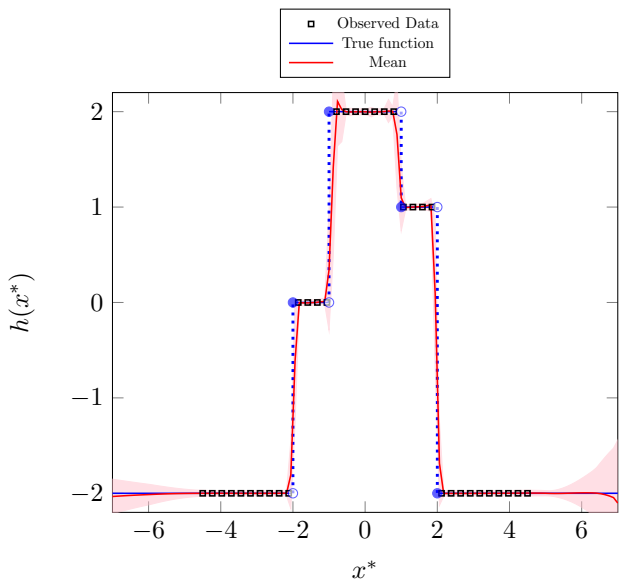


Figure 41 – Gaussian process regression with a deep kernel, using the spectral mixture (with 4 Gaussians) as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The inclusion of a DNN was also important in the process of learning the true function, even though the prediction quality is slightly inferior to the squared exponential case (particularly in the extrapolation range $[6, +\infty]$).

3.4 $f_4(x)$ (CO_2 dataset)

In all of the previously presented regression problems, we generated some data from a known function, and performed regression afterward. By doing so, we could identify if any GPR estimate $h(x^*)$ was in accordance with the true function value $f(x^*)$. This will not be the case in this example, in which only observations are available. This is essentially the case where regression is performed in real problems: we lack the knowledge of the true function and try to learn it from data and model assumptions. The chosen data was a portion of the Mauna Loa CO_2 record data (Keeling et al., 2001), particularly from years 1958 to 1986. The hyperparameter tuning process also had the set up shown in Table 2.

As we see in Figure 42, a GPR with a squared exponential kernel has a posterior mean function that approximates the training set data, although it does not match exactly the training points. It seems that some of the training set data is considered to have noise, and the 95 % confidence region incorporates most of the training set. Once again, its extrapolation predictions are quite lacking. While the Matérn kernel performs similarly in the extrapolation range, Figure 43 shows that the Matérn kernel captured an increasing periodic trend, interpolating the training set.

When a Dot product kernel is employed, the GPR mean function coincides with the best line obtained using classical linear regression with least squares. Like the squared

exponential case, the confidence region encompasses a greater number of observations, in other words, interpreting data as noise.

The composed structure in k_{Composed} permits a GPR to adjust to an increasing periodic tendency in this particular problem. In Figure 45, predictions in both interpolation and extrapolation ranges follow this tendency, with low levels of uncertainty.

As for the use of a spectral mixture kernel, we notice in Figure 46 the prediction model using a single Gaussian performs similarly to the Matérn kernel: good interpolation and bad extrapolation prediction performance. When the number of Gaussians is increased to 4, the posterior mean function in Figure 47 shows that the GPR extrapolates an increasing periodic trend.

Unlike the previous step function, none of the Deep kernels had distinct performance from those obtained with simpler kernels. Figures 48 and 49 depict posterior mean and confidence regions very similar to the Dot product case, although the prediction is not exactly an affine function in the considered interpolation and extrapolation ranges.

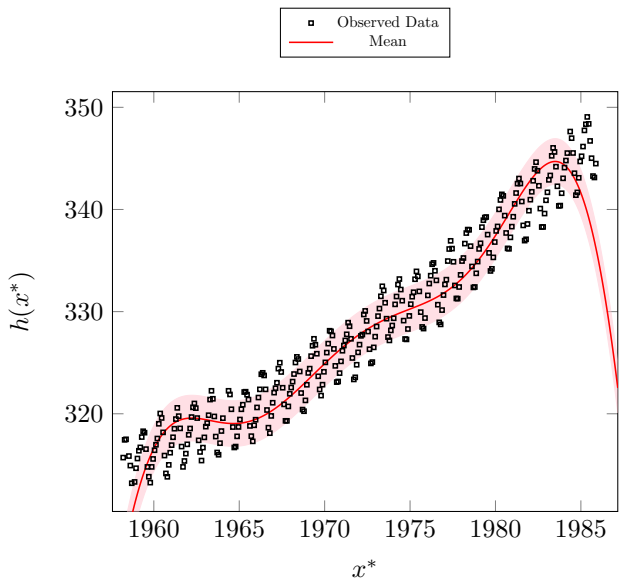


Figure 42 – Gaussian process regression with a squared exponential kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 6.718$, $\lambda_f = 6.254$, $\sigma_n = 1.225$. The posterior mean curve does not exactly interpolate the observed data, it is an approximation curve. We may notice that the confidence region encompasses the training set, interpreting the data as noise. As expected, the GPR is not capable of performing reliable extrapolation with this kernel.

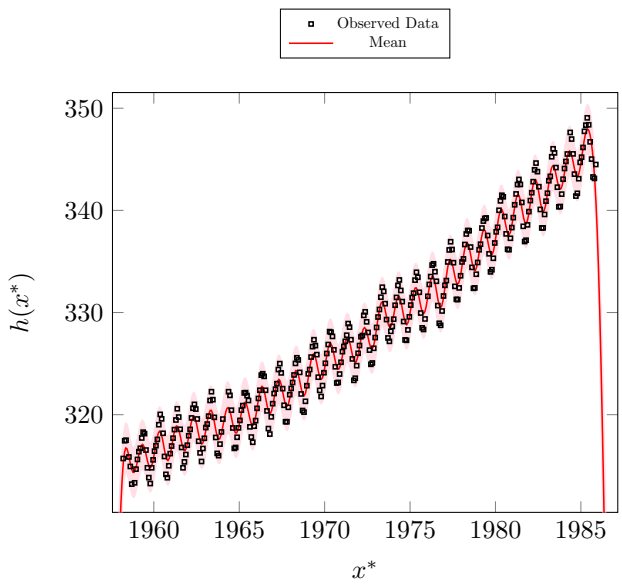


Figure 43 – Gaussian process regression with a Matérn kernel ($\nu = 2.5$). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 291.57$, $\lambda_f = 1.542$, $\sigma_n = 1.21$. The prediction in the extrapolation range are revert to the mean 0 as expected, although it differs from the squared exponential considerably this time: it interpolates the training set data. At least in the training set range, the GPR captures an increasing periodic tendency in the data.

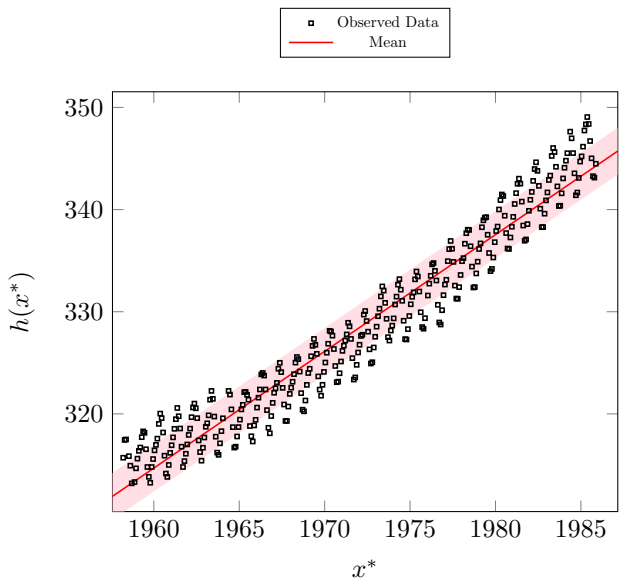


Figure 44 – Gaussian process regression with a dot product kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 159.07$, $\sigma_0 = 1681.3$, $\sigma_n = 1.24$. The posterior mean function coincides with the best line (in the least squares sense), with the confidence region incorporating a good portion of the training set data.

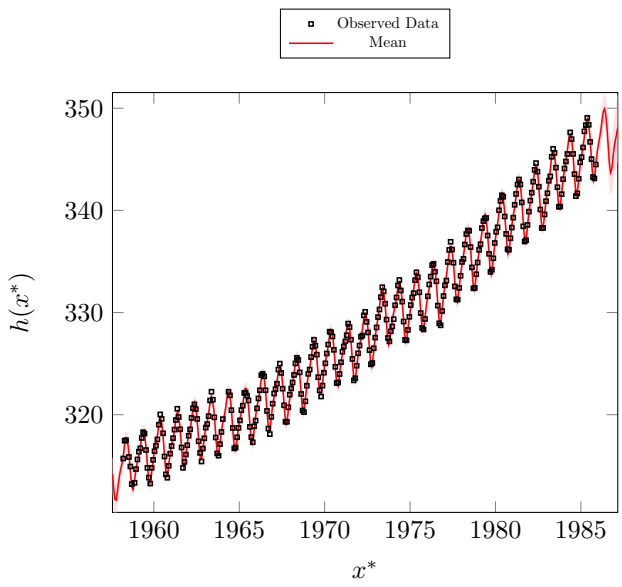


Figure 45 – Gaussian process regression with the Composed kernel defined in(3.1). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f^{(k_1)} = 1.87$, $\lambda_f^{(\text{Matern}_1)} = 20.82$, $\lambda_f^{(\text{Matern}_2)} = 1.64$, $\lambda_f^{(\text{Per})} = 1.45$, $p = 0.99$, $\sigma_f^{(k_2)} = 2 \times 10^{-8}$, $\lambda_f^{(\text{SE})} = 8.28$, $\sigma_0 = 135336.6$, $\sigma_n = 0.05$. The composed kernel structure allows an increasing periodic trend to be present in both interpolation and extrapolation ranges, with low levels of uncertainty in the predictions.

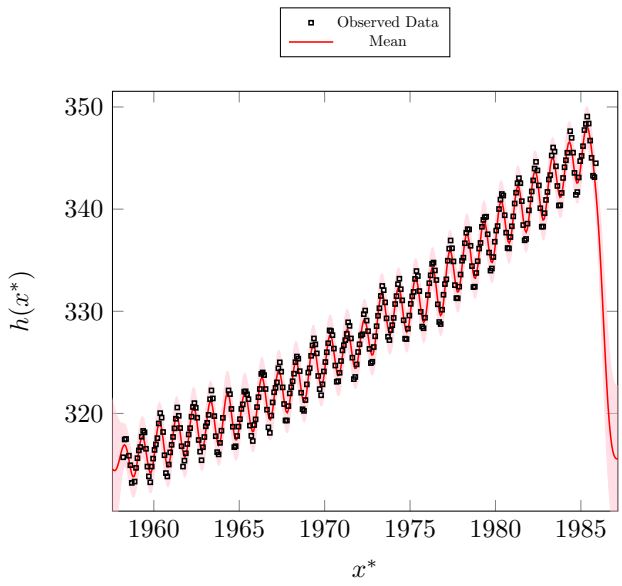


Figure 46 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 0.255$, $v = 0.312$, $\sigma = 12$, $\sigma_n = 0.8$. The overall GPR performance is very similar to the Matérn case: the posterior mean interpolates the training set data, although it lacks reliable extrapolated predictions.

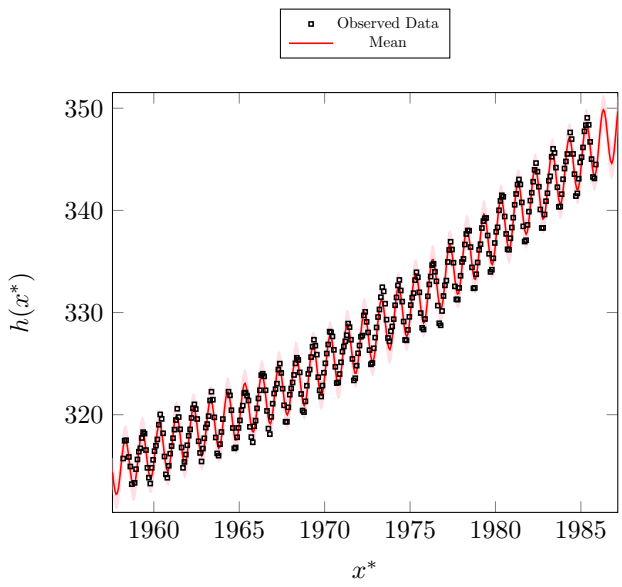


Figure 47 – Gaussian process regression with a Spectral mixture kernel (4 Gaussians). The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [1, 1.19 \times 10^{-3}, 1.57 \times 10^{-5}, 2.7]$, $V = [0.001, 0.007, 0.0002, 0.0026]$, $\boldsymbol{\sigma} = [3.17, 7.5 \times 10^{1}, 6.856 \times 10^{1}, 5.93 \times 10^{-5}]$, $\sigma_n = 0.46$. With the additional number of Gaussians, the GPR is capable of capturing an increasing periodic trend in both interpolation and extrapolation ranges, having an overall performance similar to the Composed kernel case.

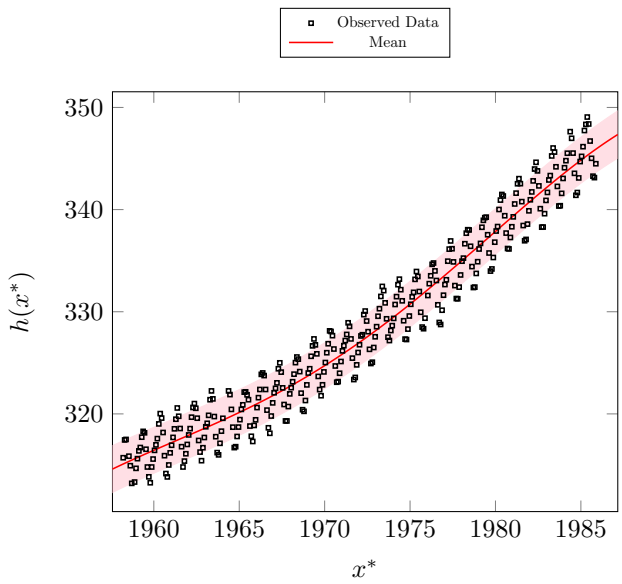


Figure 48 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The prediction curve as well as the confidence is akin to the Dot product case, although the posterior mean is not exactly an affine function.

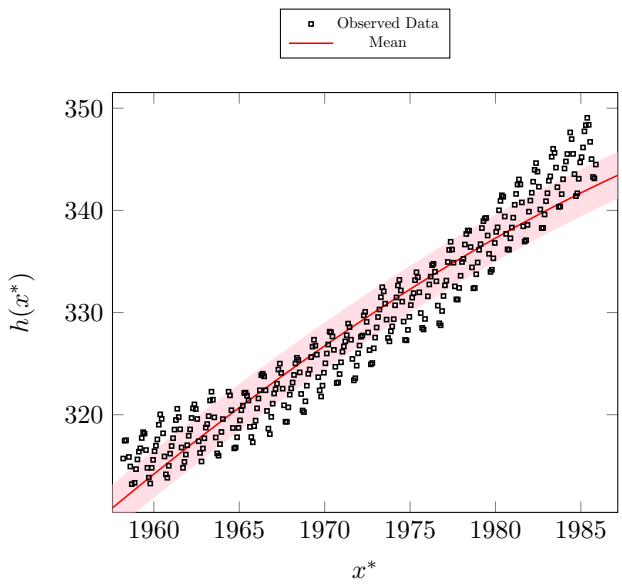


Figure 49 – Gaussian process regression with a deep kernel, using the spectral mixture (with 4 Gaussians) as base kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. Estimates in the interpolation range do not differ significantly from the ones obtained with the squared exponential as base kernel, although it is possible to notice a distinct posterior mean curve profile.

3.5 $f_5(\mathbf{x}) = -\sqrt{x_1}\sqrt{x_2}\sin(x_1)\sin(x_2)$

We already mentioned that it is possible to define Gaussian processes with a multidimensional input $\mathbf{x}$. The following function learning problem is to learn the two dimensional Alpine function with $x_1, x_2 \in [0, 10]$. This function is commonly used for benchmarking distinct optimizers, since it is a function with multiple local optimum and has a single global minimum at $\mathbf{x}_{\text{opt}} = [7.917, 7.917]$ (Jamil; Yang, 2013). Figure 50 depicts a graphical representation of this function. We aim to observe if a GPR is able to capture the true function global minimum, using $n_{\text{train}} = 85$ input samples of form $\mathbf{x} \sim [\mathcal{U}(0, 8), \mathcal{U}(0, 8)]$ as the training set. The sampled inputs $\mathbf{x}$ are represented in the yellow plane in Figure 50. In the context of global optimization, the idea of constructing a GP capable of capturing the global minimum of a certain objective function is particularly interesting if such function is costly to evaluated. Works such as (Yang et al., 2019; Preuss; Toussaint, 2018) discuss strategies in this sense. As in the f_1 and f_2 regressions, a noise $\epsilon \sim \mathcal{N}(0, 0.2^2)$ was added to the outputs.

Starting with the anisotropic squared exponential kernel, we notice that the GPR prediction surface is similar to the true function in the interpolation range, capturing the global minimum as shown in Figure 51. As for a GPR with a anisotropic Matérn kernel, Figure 52 presents a similar behaviour, although its global maximum has a higher function

value than in the squared exponential case. None of the aforementioned cases has a reliable extrapolation.

Considering that the true function can be seen as the product of two dimensional monotone and sinusoidal functions, it is possible that a GPR with spectral mixture kernel learns the true function successfully. This is indeed observed in Figures 53 and 54, with a single and 4 Gaussians, respectively. Both models are able to perform nice predictions in both interpolation and extrapolation ranges, in other words, they are able to represent the global minimum effectively.

Like some of the previous function learning problems, the results obtained with deep kernels are not superior to the ones obtained with simpler kernels. With the squared exponential kernel as base kernel, it is possible to notice an incorrect function value for the local minimum around $\mathbf{x} = [7, 4]$. This is certainly problematic if such model is used in place of the true function in a global optimization problem. Lastly, using the spectral mixture as base kernel results in a noisy version of the GPR without a DNN.

It is worth mentioning that the variances associated with the GPR were omitted for presentation purposes, although they are certainly important in order to characterize the estimates uncertainties.

The hyperparameter tuning procedure values are detailed in Table 3. We notice that, although this problem is two dimensional in the input space, the optimization process had similar set up to the previous examples.

	α (learning rate)	Iterations
Sqr. Exp. and Matérn (Anisotropic)	0.1	450
Spectral mixture kernel	0.06	1200
Deep kernel (SE)	0.00066	4000
Deep kernel (SM)	0.00066	8000

Table 3 – Hyperparameter tuning process learning rate and iterations values for the $f_5(\mathbf{x})$ function learning problem. As in the previous examples, kernel functions with higher complexity required more robust choices for learning rate and iterations.

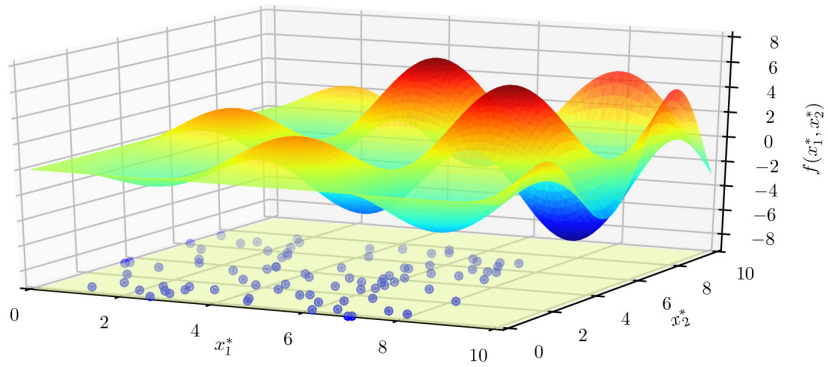


Figure 50 – Two dimensional Alpine function with $x_1, x_2 \in [0, 10]$. This function has multiple maximum and minimum points, in particular the global minimum at $\mathbf{x}_{\text{opt}} = [7.917, 7.917]$. Sampled training set inputs, used in the process of learning this function, are the blue points present in the yellow plane.

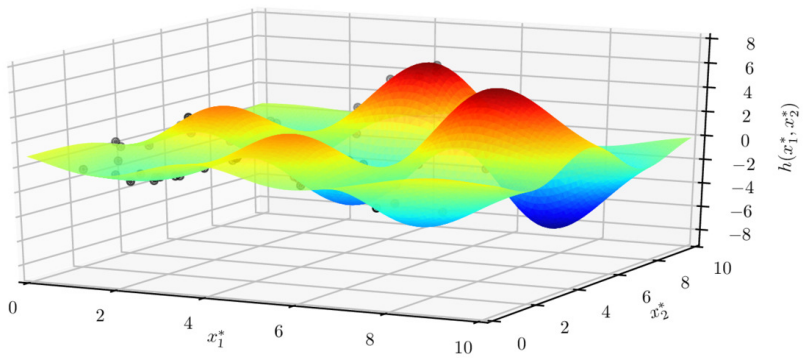


Figure 51 – Gaussian process regression with an anisotropic squared exponential kernel. The optimum hyperparameters are $\sigma_f = 7.117$, $\lambda_1 = 1.414$, $\lambda_2 = 1.405$, $\sigma_n = 0.026$. The mean function is a good approximation of the true function in the interpolation range, being capable of reproducing the global minimum. As for the extrapolation range, as in unidimensional problems, the GPR reverts to the prior $h(\mathbf{x}^*) = 0$.

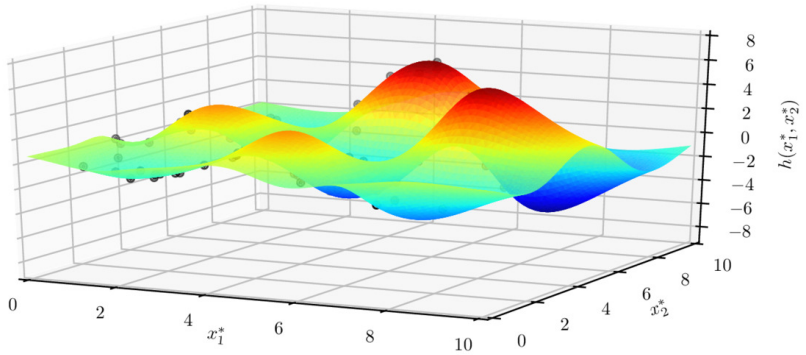


Figure 52 – Gaussian process regression with an anisotropic Matérn kernel ($\nu = 2.5$). The optimum hyperparameters are $\sigma_f = 7.97$, $\lambda_1 = 1.917$, $\lambda_2 = 1.942$, $\sigma_n = 0.011$. The mean surface is akin to the one obtained using the squared exponential kernel, although the global minimum has a higher function value.

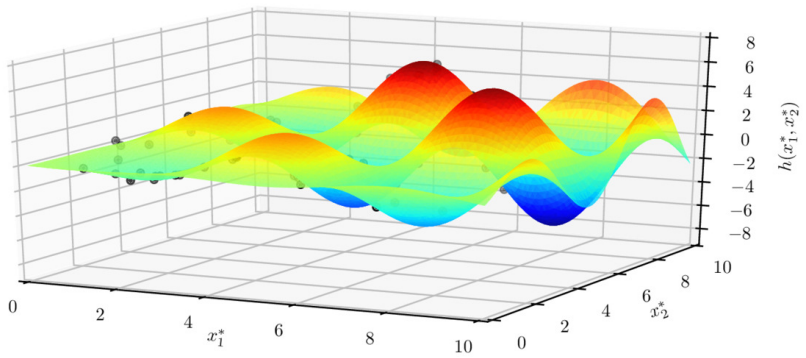


Figure 53 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian). The optimum hyperparameters are $\boldsymbol{\mu} = [0.1607, 0.1581]$, $V = [0.0185, 0.0173]$, $\sigma = 4.4549$, $\sigma_n = 0.1002$. Since the true function can be seen as the product of two dimensional monotone and sinusoidal functions (amplitudes and periodicities), a GPR with a spectral mixture kernel is successful in the learning process in both interpolation and extrapolation ranges. Considering that the GPR is able to learn the true function in the whole domain, it also learns the global minimum.

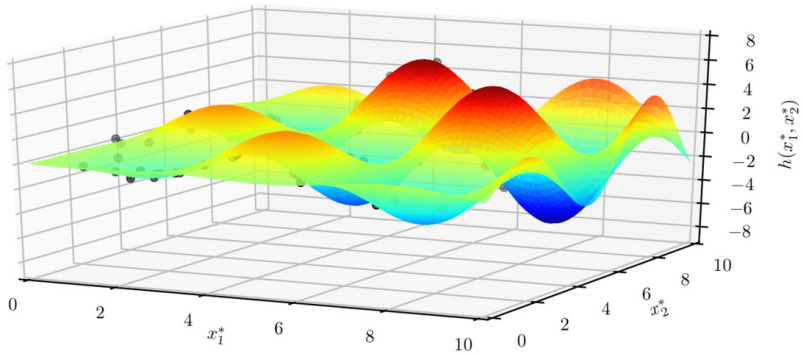


Figure 54 – Gaussian process regression with a Spectral mixture kernel (4 Gaussians). The optimum hyperparameters are $\boldsymbol{\mu} = [[5.505, 8.5], [0.161, 0.16][14.07, 0.23], [6.56, 8.43]]$, $V = [[0.001, 0.002], [0.018, 0.0176], [0.003, 0.0015], [0.0026, 0.0004]]$, $\boldsymbol{\sigma} = [1.52 \times 10^{-4}, 4.434, 3 \times 10^{-4}, 9.9 \times 10^{-4}]$, $\sigma_n = 0.1$. The GPR is capable of learning the true function as in the single Gaussian case. Interestingly, the signal variances, with the exception of the one that matches the single Gaussian case, are not very meaningful. This suggests that only one Gaussian is necessary in this regression problem.

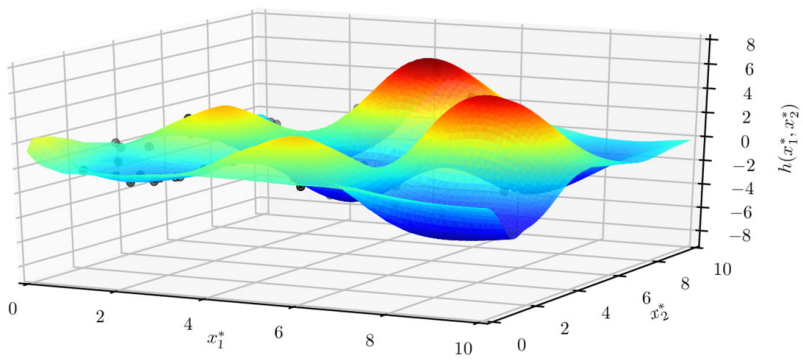


Figure 55 – Gaussian process regression with a deep kernel, using the squared exponential as base kernel. The performance is noticeably worse than the case without a DNN, since it is possible to notice an incorrect function value for the local minimum around $\mathbf{x} = [7, 4]$.

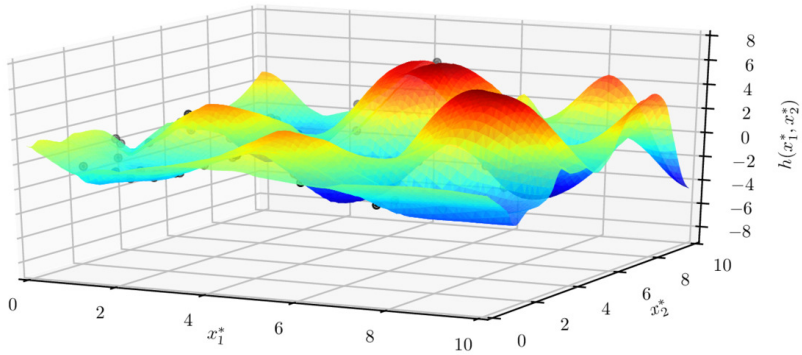


Figure 56 – Gaussian process regression with a deep kernel, using the Spectral mixture (4 Gaussians) as base kernel. Once again the inclusion of a DNN compromised the predictions: the mean surface appears to be a noisy version of the case without a DNN, as well as the obtained incorrect function value for the local minimum around $\mathbf{x} = [7, 4]$.

3.6 $\mathbf{f}_6(x) = [3\cos(x),\ 2\cos(x + 0.3)]$

We mentioned previously it is possible to apply Gaussian processes regression for vector functions, in other words, functions with more than a single output. These Gaussian processes may be defined by considering that outputs are independent or not. We selected a function learning problem described in (Wang; Chen, 2015), although we considered distinct kernels and training set in our experiments. The chosen function has two periodic outputs, possibly having a potential covariance. This function may also be interpreted as a parametric form of an elipse. The hyperparameter tuning process set up is detailed in Table 4.

Firstly, we used a squared exponential kernel assuming outputs are independent and also using the Multitask Kernel defined in (2.31). Here, the employed task covariance kernel is

$$k_{\text{task}}(i, j) = [\mathbf{b}\mathbf{b}^T + \text{diag}(\boldsymbol{\sigma}_{task})]_{ij} \ .$$

Figures 57 and 58 show similar results, with the outcomes having some noticeable differences in the extrapolation range predictions. Using the spectral mixture kernel (with a single and 4 Gaussians), both approaches have predictions in accordance with the true function, as observed in Figures 59, 60, 61, and 62. As in the previous examples, the spectral mixture kernel is successful in the process of learning a periodic function.

The predictions obtained using either independent outputs or the Multitask kernel result in similar predictions. In (Wang; Chen, 2015), it is presented that using training inputs in distinct ranges for each output helps in the GPR prediction when a Multitask kernel is used.

	α (learning rate)	Iterations
Sqr. Exp. (Ind. outputs and Multitask)	0.06	600
Spectral mixture kernel (Ind. outputs and Multitask)	0.06	1200

Table 4 – Hyperparameter tuning process learning rate and iterations values for the $\mathbf{f}_6(x)$ function learning problem. Even though the Multitask kernel introduced more hyperparameters to be estimated, the same set up was sucessful for both cases.

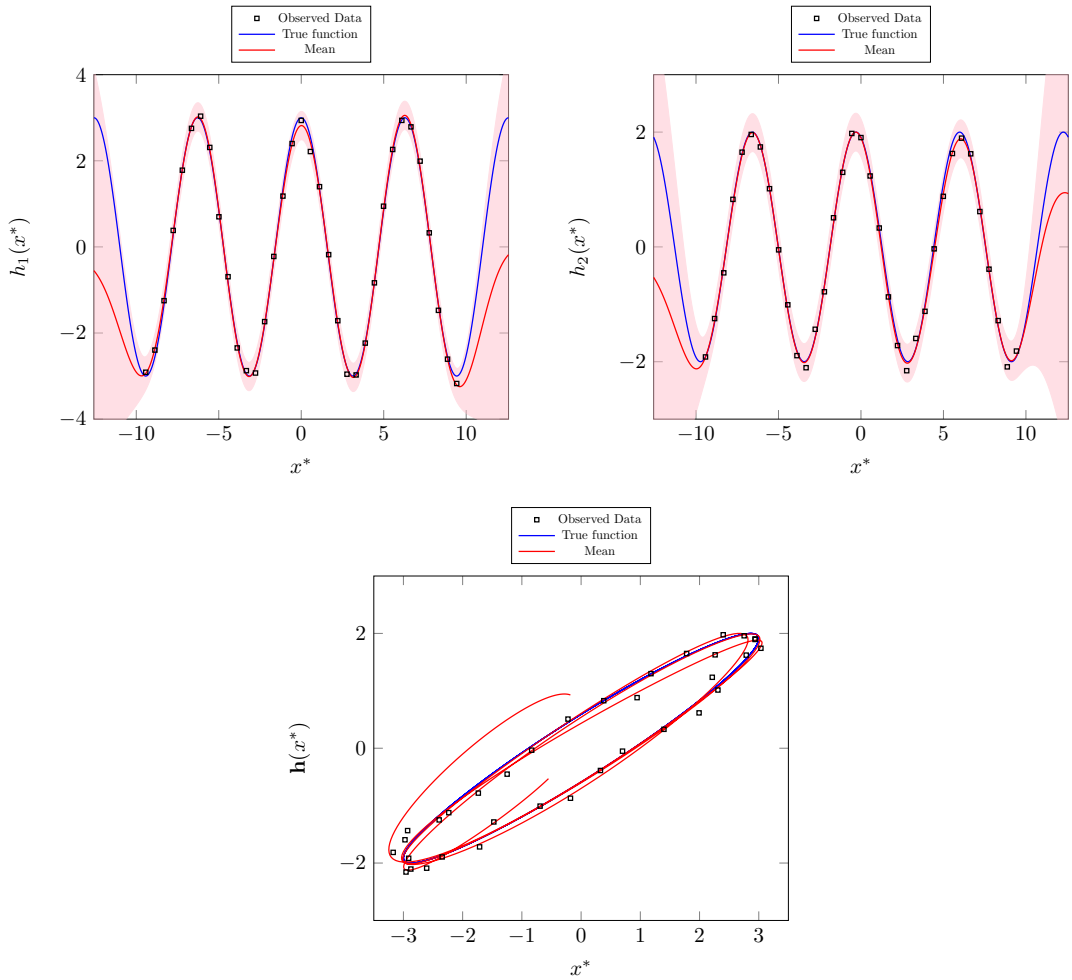


Figure 57 – Gaussian process regression with a squared exponential kernel and assuming outputs are independent. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\sigma_f = 7.8499$, $\lambda_f = 2.0448$, $\sigma_n = 0.0101$. On the top left: plot of GPR and true function (first output). On the top right: plot of GPR and true function (second output). On the center: plot of the GPR and true parametric curve. In this case, the predictions are reminiscent of the previous examples: good predictions in the interpolation range and unreliable predictions in the extrapolation range.

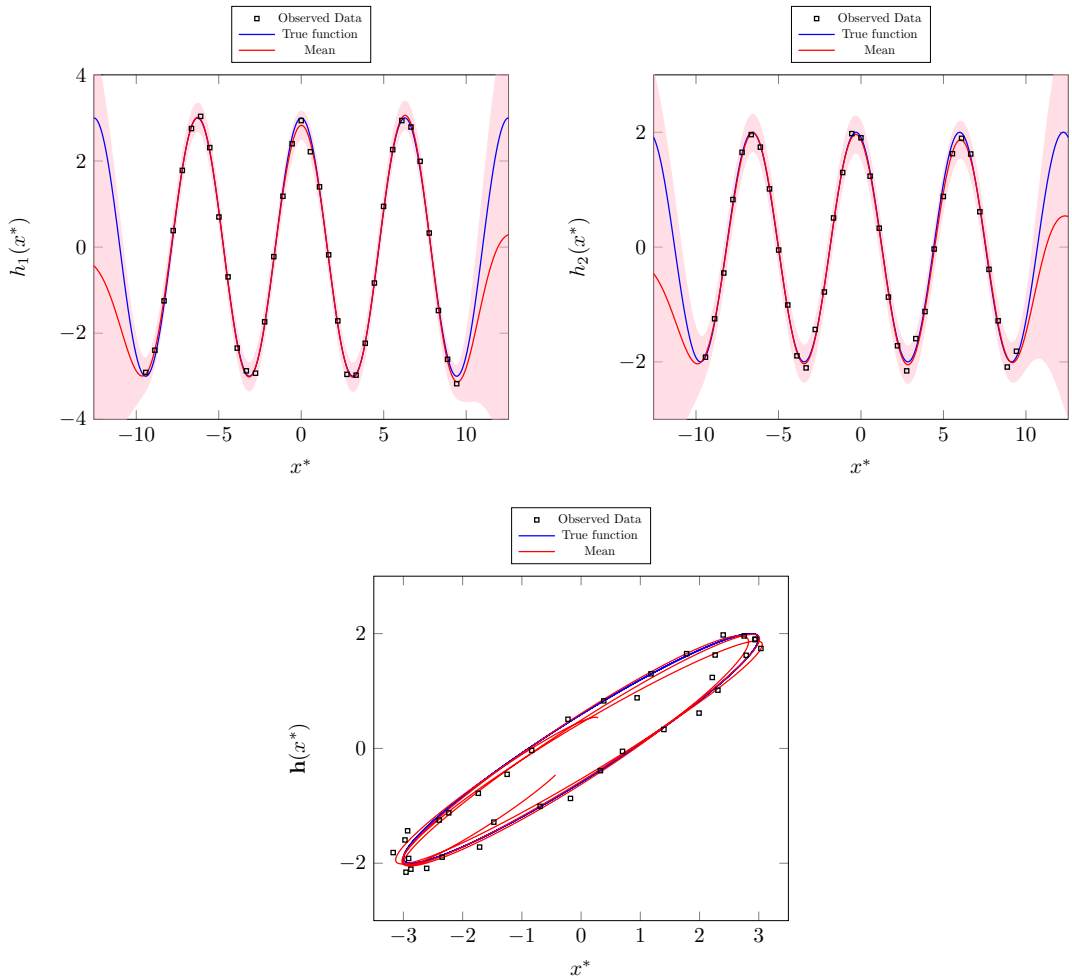


Figure 58 – Gaussian process regression with a squared exponential kernel combined with a multitask kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\lambda_f = 1.9466$, $\boldsymbol{\sigma}_{task} = [0.0096, 0.4536]$, $\mathbf{b} = [-3.0546, -1.933]$, $\sigma_n = 0.01$. On the top left: plot of GPR and true function (first output). On the top right: plot of GPR and true function (second output). On the center: plot of the GPR and true parametric curve. The posterior mean is similar to the independent outputs case in the interpolation range, although some differences are noticeable in the extrapolation range for both $h_1(x)$ and $h_2(x)$.

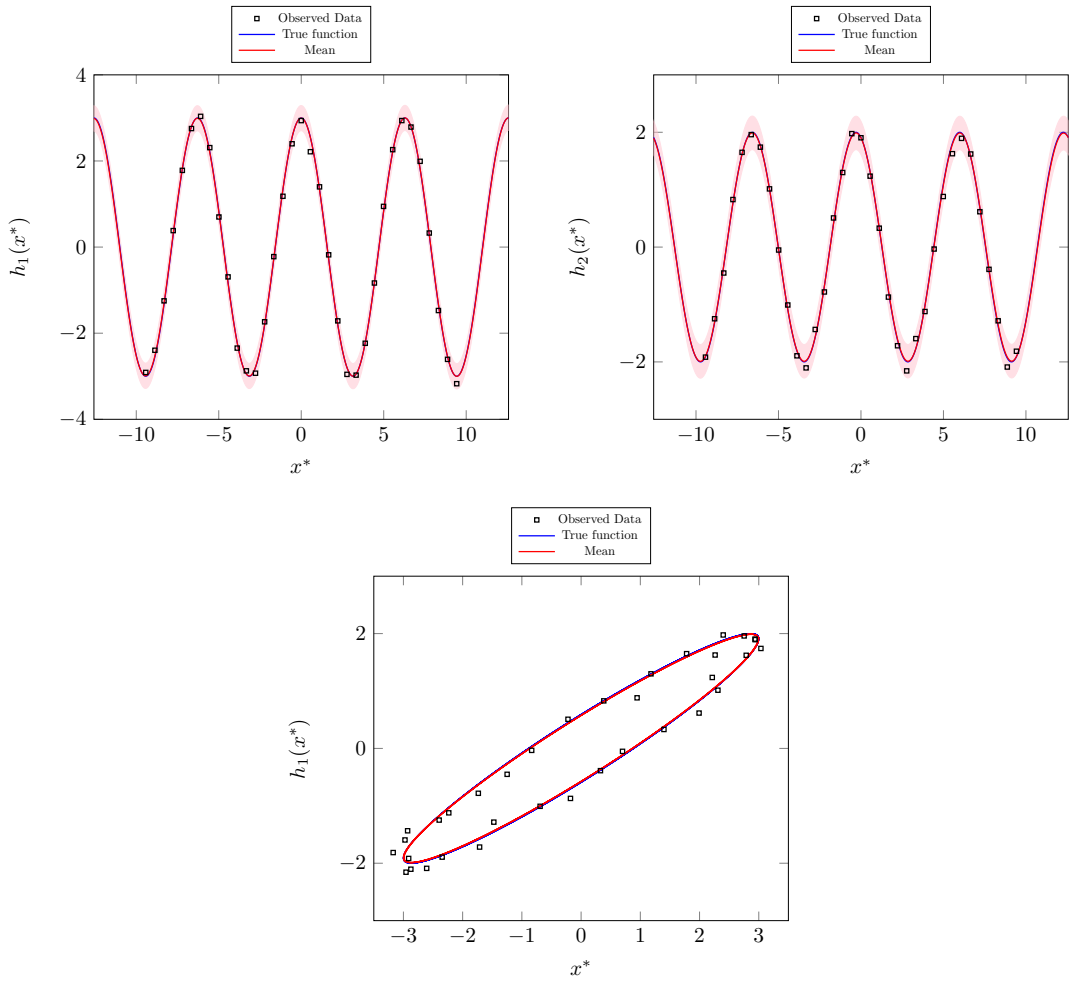


Figure 59 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian) and assuming outputs are independent. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 0.159$, $v = 0.0001$, $\sigma = 3.18$, $\sigma_n = 0.1$. On the top left: plot of GPR and true function (first output). On the top right: plot of GPR and true function (second output). On the center: plot of the GPR and true parametric curve. The posterior mean curves are virtually the same as the true function, indicating that a good kernel choice is sufficient for this regression problem.

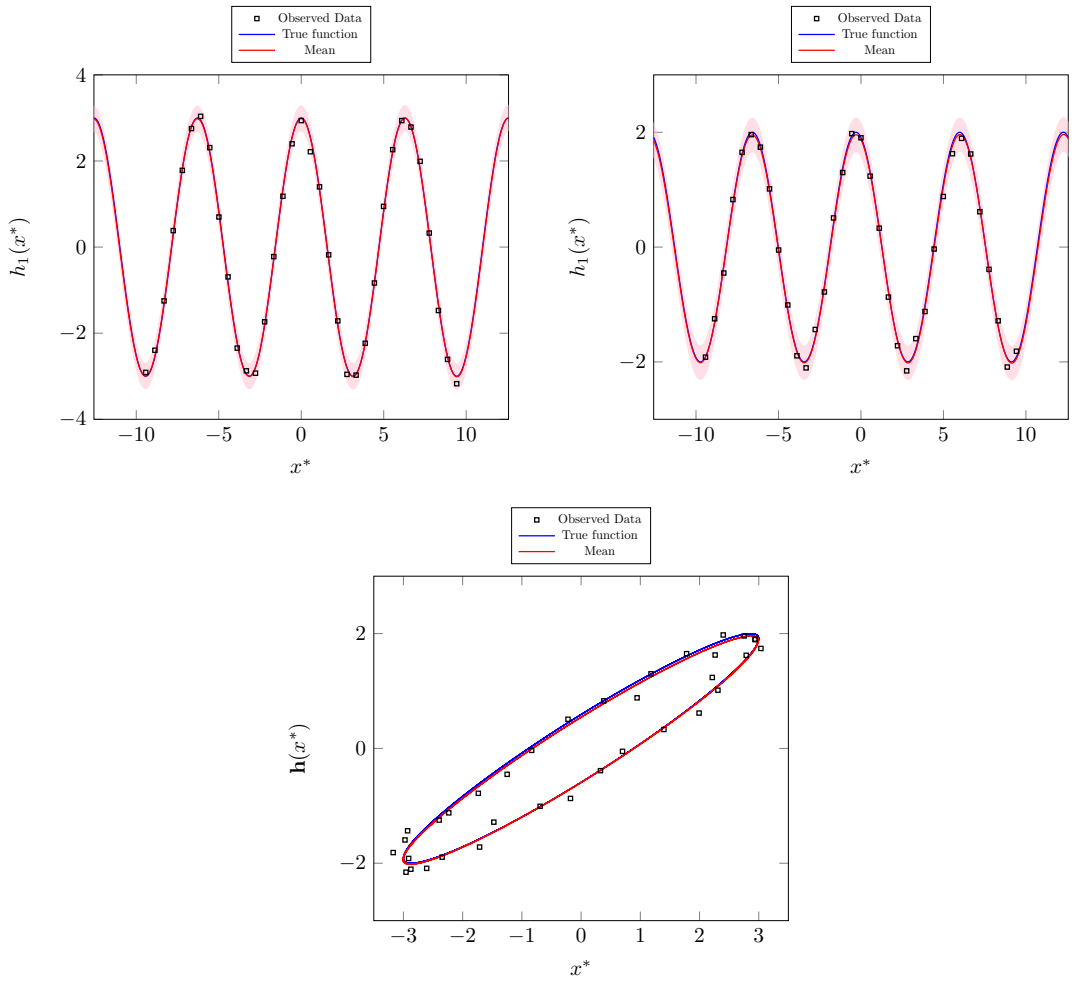


Figure 60 – Gaussian process regression with a Spectral mixture kernel (1 Gaussian) combined with a multitask kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\mu = 0.159$, $v = 0.0002$, $\sigma = 0.755$, $\boldsymbol{\sigma}_{task} = [0.4326, 0.03]$, $\mathbf{b} = [2.2575, 1.545]$, $\sigma_n = 0.01$. On the top left: plot of GPR and true function (first output). On the top right: plot of GPR and true function (second output). On the center: plot of the GPR and true parametric curve. The introduction of the Multitask kernel did not compromise the function learning process, as the GPR learns the true function as in the independent outputs case.

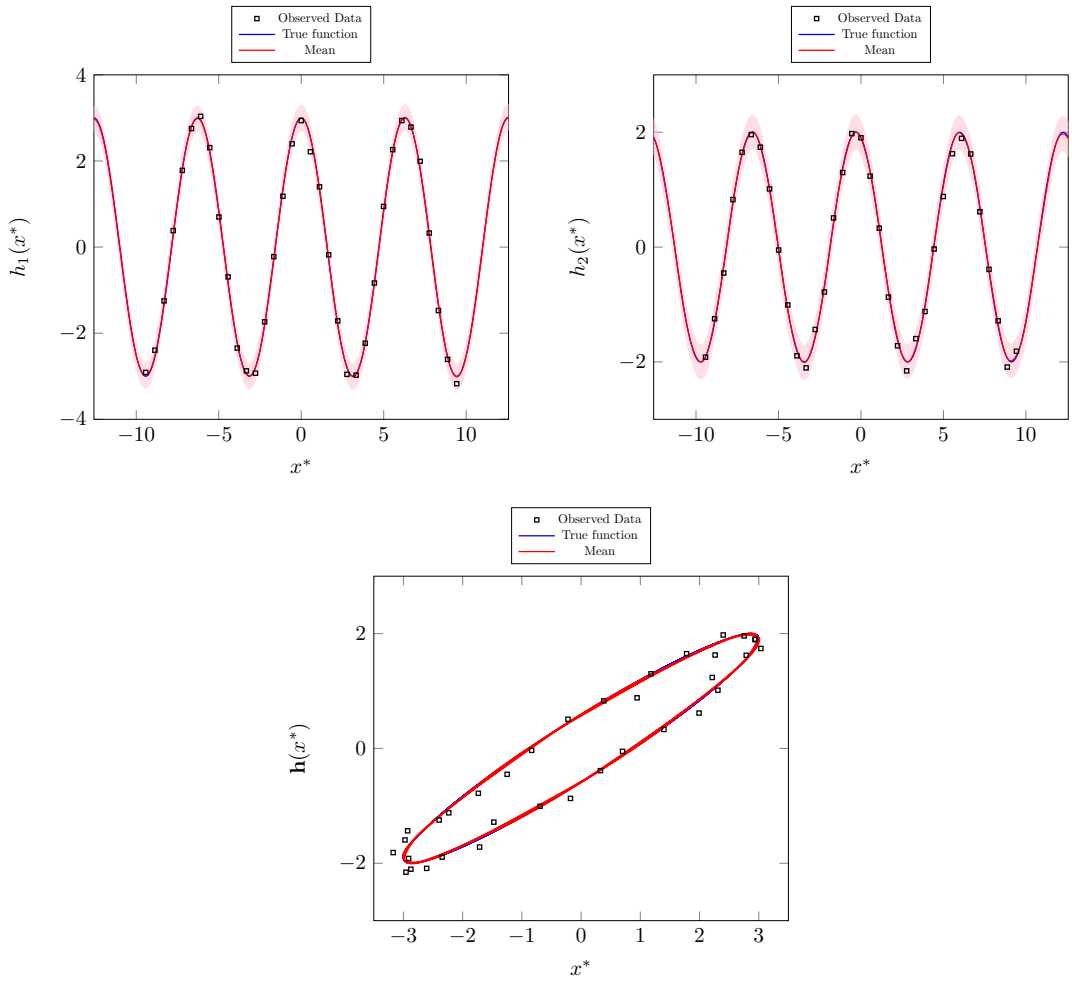


Figure 61 – Gaussian process regression with a Spectral mixture kernel (4 Gaussians) and assuming outputs are independent. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.3637, 0.82, 0.159, 0.1591]$, $V = [0.0022, 0.0038, 0.0002, 0.0023]$, $\boldsymbol{\sigma} = [1.6008 \times 10^{-5}, 3.993 \times 10^{-4}, 3.12, 7.698 \times 10^{-4}]$, $\sigma_n = 0.1$. On the top left: plot of GPR and true function (first output). On the top right: plot of GPR and true function (second output). On the center: plot of the GPR and true parametric curve. Once again, the results show that the GPR is capable of computing reliable estimates in both interpolation and extrapolation ranges.

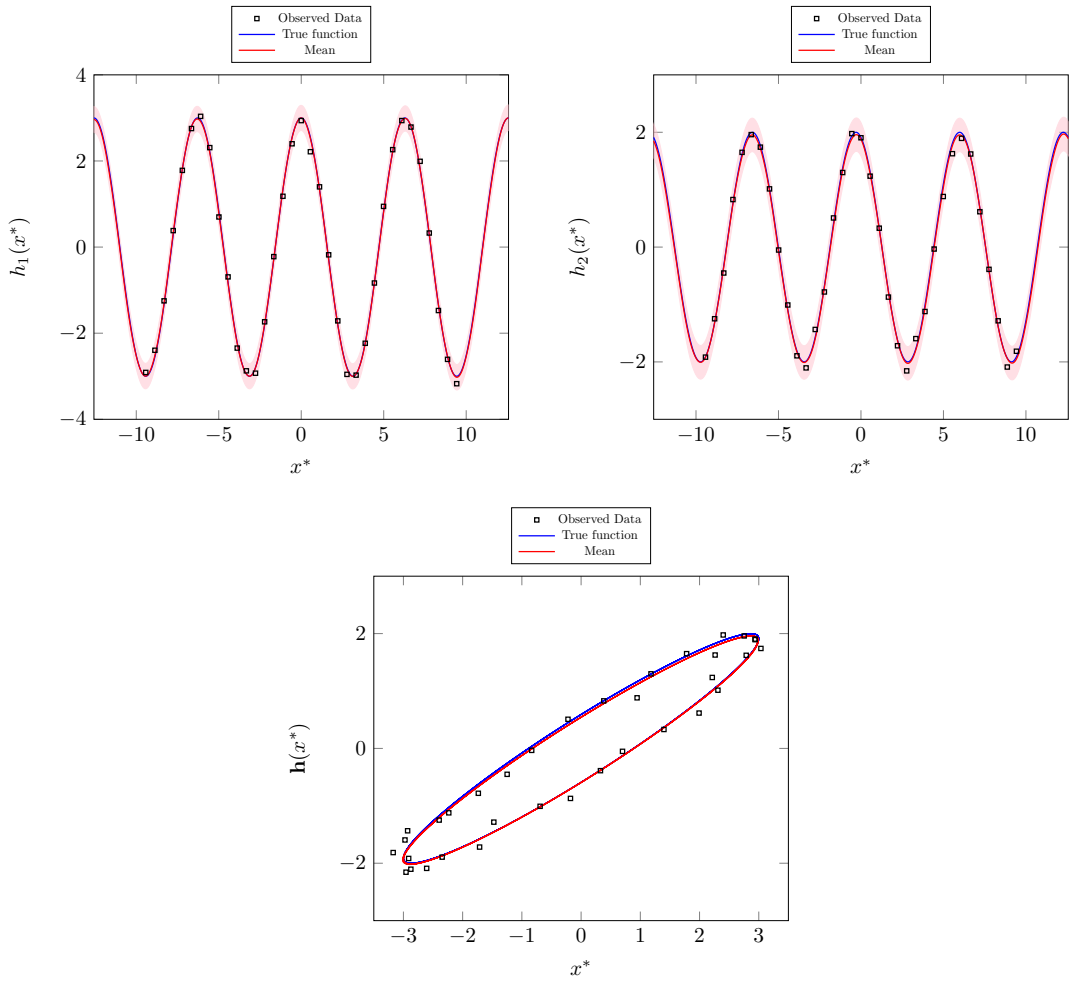


Figure 62 – Gaussian process regression with a Spectral mixture kernel (4 Gaussians) combined with a Multitask kernel. The posterior mean is in red and the shaded area represents a 95% confidence region. The optimum hyperparameters are $\boldsymbol{\mu} = [0.5393, 0.159, 0.902, 0.5384]$, $V = [0.0003, 0.0002, 0.0036, 0.006]$, $\boldsymbol{\sigma} = [3.29 \times 10^{-5}, 2.4895, 1.3523 \times 10^{-4}, 2.4133 \times 10^{-5}]$, $\boldsymbol{\sigma}_{task} = [0.138, 0.0062]$, $\mathbf{b} = [-1.2246, -0.8348]$, $\sigma_n = 0.01$. On the top left: plot of GPR and true function (first output). On the top right: plot of GPR and true function (second output). On the center: plot of the GPR and true parametric curve. The inclusion of a Multitask kernel did not introduce considerable changes compared to the GPR obtained assuming outputs are independent.

3.7 Discussion

In this chapter, we emphasized the importance of a kernel function in order to obtain reliable estimates. In every example, we highlighted the properties of a certain kernel, as well as if the GPR was adequate with the training set in question, with both theoretical and visual reasoning. Considering concerns about overfitting, extrapolation, and how visual reasoning may be subjective, what is considered as an adequate model may vary. For example, in the Mauna Loa CO_2 regression, one could argue that the extrapolation of

an periodic trend in Figures 45 and 47 are incorrect, since the true function is essentially an affine function (as in Figure 44), but the data is noisy.

We also note that, while extrapolation has to be performed with certain caution, interpolation in the case of smooth function is carried out without difficulties using a GP with a squared exponential kernel. Considering both GP prediction and confidence regions, estimates in the interpolation region were in accordance with the true function. Based on this knowledge and other observations in this chapter, we will discuss how to use a GP as a surrogate model for Bayesian inference of mechanistic tumor growth models.

4 Gaussian Process as a Surrogate Model

4.1 Introduction

Mechanistic (or physics-based) models are mathematical models formulated based on physical aspects (e.g., biological, chemical, mechanical, etc.) of an observed phenomenon. In contrast, Gaussian processes are considered statistical (or data-driven) models, as these models are strictly designed based on observed data and not on a certain type of phenomenon (Bekisz; Geris, 2020; Velten, 2009; Rio-Chanona et al., 2019). Both model types have parameters, the ones associated with mechanistic models being physically interpretable. In particular, tumor growth models have parameters related to the time necessary for tumor cells to produce new cells, how they uptake nutrients, and respond to tumor microenvironment stimuli, to mention a few mechanisms. Of note, it is also possible to interpret how parameters affect data-driven models (as we did in the previous chapter), although only from a mathematical standpoint.

Cancer is a group of diseases that involve complex relations in distinct scales in space and time (Rockne et al., 2019; Altrock; Liu; Michor, 2015). In the literature, there are models which are dedicated to understanding how the disease evolves on a specific scale (for example, only at the tissue level) on a limited time span (Lowengrub et al., 2009; Jackson et al., 2006; Bellomo; Li; Maini, 2008). There are also models which try to capture mechanisms occuring at different time and spatial scales, these are known as multiscale, usually hybrid, models (Yankeelov et al., 2016; Oden et al., 2016; Lima; Oden; Almeida, 2014; Rocha, 2019). In general, multiscale models are likely to have a high dimensional parameter space and have a considerable computational burden for their evaluation. The parameter estimation process is very challenging in these types of complex models, in particular using Bayesian inference.

4.1.1 Bayesian inference

Bayesian inference is a form of probabilistic inference. This approach not only computes the model parameters which are in most accordance with the observed data, but also a joint probability distribution for the parameters. This probability distribution is interesting from an uncertainty quantification perspective, as it incorporates both model and data uncertainties (Gamerman; Lopes, 2006; Kruschke, 2014; Collis et al., 2017). Bayesian inference is based on the subjective interpretation of probability, updating an initial degree of belief about the parameters with the well known Bayes theorem. Let us describe, in mathematical terms, how this technique is carried out.

Let $\mathcal{M}(\boldsymbol{\theta})$ be a mechanistic model, $\boldsymbol{\theta} \in \Theta$ its corresponding parameter vector, $\mathbf{y}$ the empirical quantity of interest (QoI), and $\mathbf{y}^*$ the predicted (simulated) QoI. We start considering that the parameter vector $\boldsymbol{\theta}$ is not fixed: it follows a certain joint probability distribution $\pi(\boldsymbol{\theta})$ known as the prior distribution. Secondly, it is necessary to define a likelihood function $l(\mathbf{y}|\boldsymbol{\theta})$. This function performs a significant task in the Bayesian updating procedure: it is the means through which the data modifies the prior knowledge of the parameter vector, being proportional to the probability distribution of $\mathbf{y}$ given $\boldsymbol{\theta}$. In other contexts, the logarithm of this function is interpreted as negative loss functions (or misfit) used to evaluate how distinct empirical and predicted QoI are for a fixed $\boldsymbol{\theta}$ (Goodfellow; Bengio; Courville, 2016). It is worth mentioning that $l(\mathbf{y}|\boldsymbol{\theta})$ is a function of $\mathcal{M}(\boldsymbol{\theta})$. This means that it is necessary to evaluate $\mathcal{M}(\boldsymbol{\theta})$ in order to obtain $l(\mathbf{y}|\boldsymbol{\theta})$.

The prior distribution and the likelihood function are combined by the Bayes theorem in the following way:

$$\pi(\boldsymbol{\theta}|\mathbf{y}) = \frac{l(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\pi(\mathbf{y})}, \tag{4.1}$$

in which $\pi(\boldsymbol{\theta}|\mathbf{y})$ is the joint posterior distribution of $\boldsymbol{\theta}$ given the data $\mathbf{y}$. We note that $\pi(\mathbf{y})$ does not depend on $\boldsymbol{\theta}$, being a normalization constant denominated model evidence. Being a probability distribution, the posterior expresses more information than a point estimate.

In general, it is not possible to obtain the analytical form of the posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{y})$. Fortunately, there are computational methods capable of generating an approximate posterior distribution through sampling. Monte Carlo Markov Chain (MCMC) algorithms are widely used for this purpose, with the Metropolis-Hastings being one of the first ones to be developed (Robert; Casella, 2011; Hastings, 1970). It consists of the following steps:

1. Generate a sample from the prior distribution $\boldsymbol{\theta}^{(0)} \sim \pi(\boldsymbol{\theta})$.
2. Given the current state $\boldsymbol{\theta}^{(t)}$, compute a new state from the proposal distribution $\overline{\boldsymbol{\theta}} \sim q(\overline{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(t)})$.
3. Compute the acceptance ratio
$$r = \min\left(1, \frac{l(\mathbf{y}|\overline{\boldsymbol{\theta}})\pi(\overline{\boldsymbol{\theta}})q(\boldsymbol{\theta}^{(t)}|\overline{\boldsymbol{\theta}})}{l(\mathbf{y}|\boldsymbol{\theta}^{(t)})\pi(\boldsymbol{\theta}^{(t)})q(\overline{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(t)})}\right).$$
4. Perform the acceptance test:
   i Generate a sample $u \sim \mathcal{U}(0,1)$.
   ii Set
$$\boldsymbol{\theta}^{(t+1)} = \begin{cases} \overline{\boldsymbol{\theta}}, & \text{if } u < r\ , \\ \boldsymbol{\theta}^{(t)}, & \text{otherwise}\ . \end{cases}$$

5. Unless the maximum number of accepted samples is achieved, return to step 2.

One may note that this algorithm starts from a sample of the prior distribution $\boldsymbol{\theta}^{(0)}$. Next, it performs a loop consisting of generating a new candidate $\overline{\boldsymbol{\theta}}$ from a chosen proposal distribution $q(\overline{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(t)})$, and accepting or discarding it. This proposal distribution depends only on the current parameter vector $\boldsymbol{\theta}^{(t)}$ and $\overline{\boldsymbol{\theta}}$, thus turning the problem into sampling from a Markov Chain. Any distribution $q(\cdot,\cdot)$ may be selected, although a bad choice may lead to a high number of rejected samples. A common choice is to use a normal distribution centered in the current state $\boldsymbol{\theta}^{(t)}$, using an appropriate variance for the inference problem of interest (Rosenthal et al., 2011). This simplifies computations, as a symmetric distribution cancels the numerator and denominator proposal terms in the acceptance ratio expression r. This acceptance ratio r is defined in such form that regions with higher probability are necessarily accepted, which corresponds to having $r = 1$. There is also a possibility of accepting a new candidate with a lower associated probability, otherwise the sampling procedure would be stuck around a mode (or maximum) of the posterior distribution, being incapable of sampling along all its support. Lastly, when a desired number of accepted samples is obtained, the algorithm ends. These accepted samples correspond to the approximate posterior distribution.

There are other MCMC algorithms that are built upon the outlined Metropolis-Hastings algorithm. There is the Gibbs sampler (Robert; Casella, 2011; Gamerman; Lopes, 2006; Wasserman, 2013), which conditions each parameter vector dimension separately, thus reducing the challenge of producing samples in high dimensional problems. There are also adaptive strategies which avoid a large number of proposed candidates $\overline{\boldsymbol{\theta}}$ being discarded (Rosenthal et al., 2011; Andrieu; Thoms, 2008). Moreover, there are methods which use gradient information in order to produce more candidates in regions of higher probability (Hoffman; Gelman, 2014; Betancourt, 2017). Unfortunately, all of these methods have the same bottleneck: evaluating the likelihood (or even its gradient) several times for each new candidate, which corresponds to simulating the model $\mathcal{M}(\overline{\boldsymbol{\theta}})$ repeatedly. Since this task is prone to be computationally prohibitive for complex models, and in particular for tumor growth models, it is interesting to consider alternative approaches.

4.1.2 Approximate Bayesian Computation

Aiming to alleviate the computational burden in traditional MCMC methods, a popular strategy known as Approximate Bayesian Computation (ABC) (Beaumont, 2019) has been proposed. This technique replaces likelihood function evaluations present in MCMC algorithms with simpler computations. This is particularly useful when $l(\mathbf{y}|\cdot)$ is costly or unknown. It starts by defining an augmented posterior distribution

$$\pi(\boldsymbol{\theta}, \mathbf{y}^*|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})\pi(\mathbf{y}^*|\boldsymbol{\theta})\pi(\boldsymbol{\theta}), \tag{4.2}$$

in which $p(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})$ is the weight function (a measure of similarity between empirical $\mathbf{y}$ and predicted $\mathbf{y}^*$ QoIs), and $\pi(\mathbf{y}^*|\boldsymbol{\theta})$ is known as generative model (the same as a model simulation $\mathcal{M}(\theta)$). When $\mathbf{y} = \mathbf{y}^*$, $p(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})$ is a positive constant such that $\pi(\boldsymbol{\theta}, \mathbf{y}^*|\mathbf{y}) \propto \pi(\mathbf{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) \propto \pi(\boldsymbol{\theta}|\mathbf{y})$. In other words, the augmented posterior is proportional to the standard posterior. An approximate posterior distribution is attained through marginalization:

$$\pi_{ABC}(\boldsymbol{\theta}|\mathbf{y}) \propto \pi(\boldsymbol{\theta}) \int_{\mathcal{Y}} p(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})\pi(\mathbf{y}^*|\boldsymbol{\theta})dy. \tag{4.3}$$

Considering the case where $p(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})$ is a probability mass function consisting of $P(\mathbf{y} = \mathbf{y}^*) = 1$, $\pi_{ABC}(\boldsymbol{\theta}|\mathbf{y})$ coincides with the true posterior $\pi(\boldsymbol{\theta}|\mathbf{y})$. In general, it is very unlikely that the model $\mathcal{M}(\boldsymbol{\theta})$ outputs a simulated $\mathbf{y}^*$ equal to the empirical $\mathbf{y}$. The main idea is to not require that these two quantities are exactly the same, but also to consider situations in which $\mathbf{y} \approx \mathbf{y}^*$. To this end, we first have to define how similar these two values are using a function (e.g., Euclidean or Manhattan distances) $d(\mathbf{y}, \mathbf{y}^*)$ and a certain threshold $\epsilon > 0$. We also introduce a smoothing kernel function $\mathcal{K}$ so that

$$p_\epsilon(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta}) = \frac{1}{\epsilon}\mathcal{K}\left(\frac{d(\mathbf{y}, \mathbf{y}^*)}{\epsilon}\right) , \tag{4.4}$$

for which $\lim_{\epsilon \to 0} p_\epsilon(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})$ is a mass point at $\mathbf{y}$. Both $\mathcal{K}$ and ϵ must be chosen with care, otherwise the ABC method may be compromised. One of the most used kernel smoothing functions is the uniform density kernel, which results in the following weight function:

$$p_\epsilon(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta}) = \begin{cases} 1, & d(\mathbf{y}, \mathbf{y}^*) \leq \epsilon \ , \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

The expression in (4.5) is based on a simple concept: if $d(\mathbf{y}, \mathbf{y}^*)$ is less than the specified threshold, we consider that the simulated QoI $\mathbf{y}^*$ matches the empirical QoI $\mathbf{y}$. Other choices of $\mathcal{K}$ are described in (Sisson; Fan; Beaumont, 2018).

There are multiple ABC variants in the literature, one of those is the ABC-MCMC presented in (Marjoram et al., 2003; Beaumont, 2019). This method is pretty similar to the Metropolis-Hastings method, although it differs essentially in the acceptance test step. As ABC methods are likelihood-free, this acceptance test consists of evaluating the weight function $p_\epsilon(\mathbf{y}|\mathbf{y}^*, \boldsymbol{\theta})$, and then consider an acceptance ratio based on the product of prior and proposal distributions. The ABC-MCMC involves the following steps:

1. Generate a sample from the prior distribution $\boldsymbol{\theta}^{(0)} \sim \pi(\boldsymbol{\theta})$.
2. Given the current state $\boldsymbol{\theta}^{(t)}$, compute a new state from the proposal distribution $\overline{\boldsymbol{\theta}} \sim q(\overline{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(t)})$.
3. Compute $\mathbf{y}^*$ from $\pi(\mathbf{y}^*|\overline{\boldsymbol{\theta}})$.
4. Perform acceptance test:

i If $d(\mathbf{y}, \mathbf{y}^*) > \epsilon$, set $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)}$ and return to step 2. Otherwise, continue.

ii Compute acceptance ratio

$$r = \min\left(1, \frac{\pi(\overline{\boldsymbol{\theta}})q(\boldsymbol{\theta}^{(t)}|\overline{\boldsymbol{\theta}})}{\pi(\boldsymbol{\theta}^{(t)})q(\overline{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(t)})}\right).$$

iii Generate a sample $u \sim \mathcal{U}(0, 1)$.

iv Set

$$\boldsymbol{\theta}^{(t+1)} = \begin{cases} \overline{\boldsymbol{\theta}}, & \text{if } u < r \ , \\ \boldsymbol{\theta}^{(t)}, & \text{otherwise} \ . \end{cases}$$

5. Unless the maximum number of accepted samples is achieved, return to step 2.

Even though this strategy is capable of reducing the overall computational time, step 3 involves the evaluation of $\mathcal{M}(\overline{\boldsymbol{\theta}})$, and performing this repeatedly may be not feasible.

There are alternatives to ABC such as Variational Bayes (Blei; Kucukelbir; McAuliffe, 2017), which substitutes a sampling procedure to an optimization one that requires less iterations. On the other hand, it demands strong additional assumptions around the posterior distribution that may not hold. Moreover, this optimization procedure still requires evaluation of $\mathcal{M}(\overline{\boldsymbol{\theta}})$ consecutively. Thus, aiming to attenuate this cost in particular, we consider the introduction of a surrogate model.

4.1.3 Surrogate model

A surrogate model, metamodel or emulator is a "model of a model", in other words, a simplified model of the actual model. Surrogate models can be mechanistic, in which they are lower-fidelity versions of the original model (Koziel; Ciaurri; Leifsson, 2011; Razavi; Tolson; Burn, 2012). These may be obtained after simplifications in the model equations or by using coarser discretization meshes. From a computational standpoint, they are cheaper to be evaluated. Another approach is to build a dataset of previously executed model runs and then construct data-driven surrogates. The advantage of using these types of surrogate models in place of the original model is that they provide output approximations which require less computational effort (Bhosekar; Ierapetritou, 2018; Koziel; Ciaurri; Leifsson, 2011; Razavi; Tolson; Burn, 2012). Distinct types of statistical models such as polynomial regressions, support vector regression, neural networks, and Gaussian processes have been used for this purpose (Bhosekar; Ierapetritou, 2018; Tripathy; Bilionis, 2018). The use of surrogates is common in scientific problems that require repeated model evaluation. Some examples include optimization, mesh solution interpolation, sensitivity analysis, and inverse problems (Yang et al., 2019; Preuss; Toussaint, 2018; Chiplunkar; Bosco; Morlier, 2017; Cheng et al., 2020; Kiessling; Ström; Tempone, 2021; Freitas et al., 2020).

In our work, we propose a Gaussian process surrogate to speed up the calculations of $\mathcal{M}(\theta)$ required for the parameter estimation performed by the ABC-MCMC method. We begin defining a regression problem and the properties of the GP. Afterward, we describe the steps on how to build such surrogate GPR and how to incorporate it in the ABC-MCMC method.

4.2 Regression and surrogate model description

Before defining a data-driven surrogate to a mechanistic model, we first define a regression problem. Since our objective is to solve a parameter estimation problem, we are interested in a metamodel capable of capturing how the parameter vector $\boldsymbol{\theta}$ influences the simulated QoI $\mathbf{y}^*$. We define $\mathbf{y}^*(\boldsymbol{\theta}) \in \mathbb{R}^n$ as the simulated QoI obtained using $\boldsymbol{\theta}$, in the times when there is empirical data $\{t_1, \ldots, t_n\}$. It is possible to build from m model runs the following dataset:

$$\mathcal{D} = \{\boldsymbol{\theta}^{(j)}, \mathbf{y}^*(\boldsymbol{\theta}^{(j)})\}_{j=1}^m \ . \tag{4.6}$$

Therefore, the regression problem consists in learning a multi-output function, having $\boldsymbol{\theta}$ as the input and $\mathbf{y}^*(\boldsymbol{\theta})$ as the output. The associated training set is the dataset $\mathcal{D}$ defined in (4.6).

We consider a multi-output Gaussian process as a surrogate model. In the previous chapter, we stated that it is necessary to select a kernel function so as to define a GP. As the process of producing reliable estimates requires an appropriate kernel, we choose a kernel suitable for this form of surrogate modeling.

In order to choose our GP surrogate kernel, we highlight some of the aspects involved in a Bayesian inference process. Firstly, we note that only parameter vectors $\boldsymbol{\theta}$ in the support of the prior distribution $\pi(\boldsymbol{\theta})$ are considered. This means that we are not interested in the behavior of $\mathbf{y}^*(\boldsymbol{\theta})$ if $\boldsymbol{\theta}$ is not in this set. Furthermore, we are capable of generating new data to incorporate in (4.6), as our training set is composed by previous runs of the mechanistic model $\mathcal{M}(\boldsymbol{\theta})$. In view of these, we opted for the anisotropic squared exponential kernel, as we know the range of inputs for which the estimates must be computed. In other words, the support of the prior distribution defines the range in which we use the predictions of the surrogate model during the inference process. In this way, we are able to generate new samples of the prior distribution and use the predictions of the surrogate model, which is a good proxy for the actual model, not requiring any form of extrapolation.

4.3 Surrogate model: construction and prediction

We now describe how to construct the surrogate model, considering that our main objetive is to infer $\boldsymbol{\theta}$ using the ABC-MCMC method. The basic idea is to generate a number m of parameter vector samples from the prior distribution, obtain their associated predicted QoIs, and finally perform a Gaussian process regression. The choice of m is of high importance, since an insufficient number of samples may result in an inadequate surrogate. Also, although it is possible to introduce new samples, the cost of evaluating the model excessively undermines the use of a surrogate. Thus, it is important to define a criterion to select m. One possible criterion is presented in the following GP construction steps:

1. Generate m samples of $\boldsymbol{\theta}$ such that $\boldsymbol{\theta}^{(j)} \sim \pi(\boldsymbol{\theta})$, $j = 1, \cdots, m$.
2. Run the mechanistic model $\mathcal{M}(\boldsymbol{\theta}^{(j)})$ for each $\boldsymbol{\theta}^{(j)}$. Afterward, compute their associated QoI estimates $\mathbf{y}^*(\boldsymbol{\theta}^{(j)})$, so as to obtain the training set $\mathcal{D} = \{\boldsymbol{\theta}^{(j)}, \mathbf{y}^*(\boldsymbol{\theta}^{(j)})\}_{j=1}^m$.
3. Build a GP $h(\cdot)$ using $\mathcal{D}$. This involves the hyperparameter tuning process.
4. Generate r samples from $\bar{\boldsymbol{\theta}} \sim \pi(\boldsymbol{\theta})$ to be used as test set $R = \{\bar{\boldsymbol{\theta}}^{(j)}, \mathbf{y}^*(\bar{\boldsymbol{\theta}}^{(j)})\}_{j=1}^r$.
5. Compute the GP QoI prediction $h(\bar{\boldsymbol{\theta}}^{(j)})$ and check if the error between its mean $\boldsymbol{\mu}_j^*$ and $\mathbf{y}^*(\bar{\boldsymbol{\theta}}^{(j)})$ is below a certain threshold. If the error is below the threshold, consider the GP as a substitute for the mechanistic model in the ABC-MCMC process. Otherwise, include the r test samples in the training set $\mathcal{D}$ and return to step 3 .

Figures 63 and 64 provide a visual overview of how to construct and perform predictions using a surrogate GP, which are related to steps 1-3 and 4-5, respectively. Possible choices of error metric in step 5 include the mean squared and mean absolute errors (MSE and MAE) (Goodfellow; Bengio; Courville, 2016).

After performing the steps to construct a GP, incorporating it in the ABC-MCMC method is very straightforward. In step 3, where $\mathbf{y}^*$ is obtained by running $\mathcal{M}(\bar{\boldsymbol{\theta}})$ (see page 84), we use the mean from $h(\bar{\boldsymbol{\theta}})$.

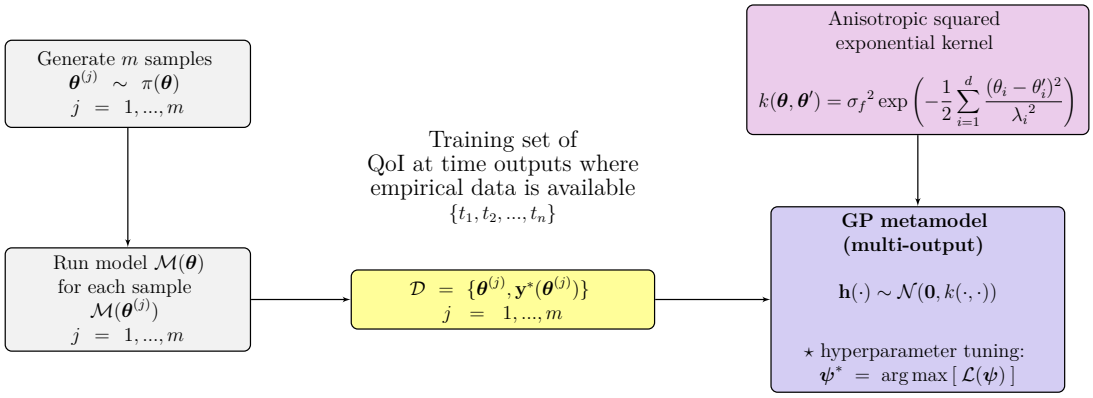


Figure 63 – Outline of the GP surrogate construction.

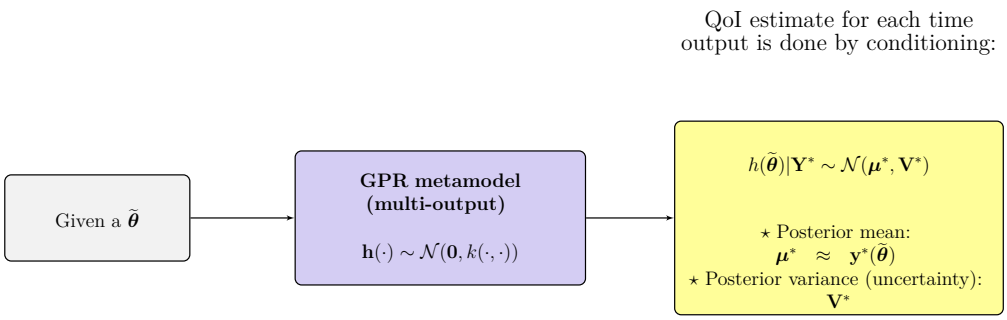


Figure 64 – Outline of how the GP surrogate produces QoI estimates at empirical times $\{t_1, t_2, ..., t_n\}$ associated to a parameter vector $\widetilde{\boldsymbol{\theta}}$.

4.4 Discussion

This chapter highlighted the challenges of performing Bayesian inference on a complex mechanistic model $\mathcal{M}(\boldsymbol{\theta})$. Using either Metropolis-Hastings or ABC-MCMC methods, we identified the necessity of lowering the cost of running $\mathcal{M}(\boldsymbol{\theta})$. We presented the definition of surrogate models and how their use is not restricted to the context of parameter estimation.

We outlined how to construct a surrogate GP dedicated to reducing the computational burden in the ABC-MCMC method. This multi-output GP is constructed by previously computed QoI estimates $\mathbf{y}^*(\boldsymbol{\theta})$ at times where empirical data is available $\{t_1, t_2, ..., t_n\}$. Since only estimates $\mathbf{y}^*(\boldsymbol{\theta})$ corresponding to parameter vectors from the prior distribution $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})$ are of interest, we consider that a GP with a squared exponential kernel is a good proxy for the true model.

There are ways of improving even further the process of building a surrogate GP. We note that, in step 5 of the GP construction steps (see page 87), it is possible to also consider the variance of the GP estimates. This could be an additional criterion in order to decide if the GP needs a training set with more samples. Furthermore, it is possible to

verify if the variance is below a defined tolerance inside the ABC-MCMC method. If the variance is above the preset tolerance, a single true model run may be performed, and this new sample can be incorporated in the GP. Other adaptative strategies may be performed, like the one developed in (Rocha, 2019).

In the next chapter, we verify how the described methodology performs on some mechanistic tumor growth model.

5 GP Surrogate Models in Tumor Growth

In this chapter, we perform parameter inference using the methodology described in Chapter 4 in two mechanistic tumor growth models. We compare the posterior distributions using the proposed approach (ABC-MCMC with a GP surrogate) and directly applying the ABC-MCMC method. In order to compare two probability distributions, the measure known as the Kullback-Leibler (KL) divergence (Shlens, 2014; Wasserman, 2013) is used.

5.1 Verhulst model

The first tumor growth model is the well-known Verhulst (or logistic) model. In the present work, this model depicts the *in vitro* growth of SUM-149PT cells (invasive breast cancer). Denoting $C(t)$ as the density of tumor cells, this mathematical model is expressed by the following ordinary differential equation (ODE):

$$\begin{cases} \dfrac{dC}{dt} = rC\left(1 - \dfrac{C}{\kappa}\right), \quad t \in (0, T], \\ C(0) = C_0. \end{cases} \tag{5.1}$$

This model has two parameters: the maximum growth rate r and the carrying capacity κ. These parameters are related to the velocity in which tumor cells reproduce and how this process is affected by nutrient scarcity and spatial limitation. We aim to infer $\boldsymbol{\theta} = \{r, \kappa\}$ using data from (McKenna, 2017). Before performing any form of Bayesian inference, it is necessary to assign a prior distribution $\pi(\boldsymbol{\theta})$. Regarding the ABC-MCMC method, it is also required to select a kernel smoothing function $\mathcal{K}$. We used the function defined in (4.5), which still involves specifying a distance function $d(\mathbf{y}, \mathbf{y}^*)$ and a threshold ϵ. These decisions are presented in Table 5.

Here, we denote $\pi(\boldsymbol{\theta}|\mathbf{y})$ and $\pi_{GP}(\boldsymbol{\theta}|\mathbf{y})$ as the posterior distributions obtained using ABC-MCMC and ABC-MCMC with GP surrogate approaches. The marginal posterior distributions for both parameters are portrayed in Figure 65. While it is possible to notice similar results between the distributions from a visual standpoint, we also compare the distribution using the KL divergence.

In this context, we comment on some aspects regarding the computation of KL divergence for a certain parameter θ. Firstly, the comparison is performed between two posterior distributions approximations in the form of histograms. This is achieved by considering the histograms as probability mass functions, and then using the KL divergence expression for discrete distributions

$$D_{\mathrm{KL}}(\pi(\theta|\mathbf{y}) \parallel \pi_{GP}(\theta|\mathbf{y})) = \sum_i \pi(\theta_i|\mathbf{y}) \ln\left(\frac{\pi(\theta_i|\mathbf{y})}{\pi_{GP}(\theta_i|\mathbf{y})}\right). \tag{5.2}$$

Unfortunately, there may be a drawback in using equation (5.2). If the supports of the posterior distributions are different, this expression results in ∞. In order to circumvent this difficulty, we applied a technique known as Laplace smoothing (or rule of succession) (Singer, 1999; Simonoff, 1995; Schütze; Manning; Raghavan, 2008).

The KL divergences of the marginal distributions are

$$D_{\mathrm{KL}}(\pi(\kappa|\mathbf{y}) \parallel \pi_{GP}(\kappa|\mathbf{y})) = 0.004825 \quad \text{and} \quad D_{\mathrm{KL}}(\pi(r|\mathbf{y}) \parallel \pi_{GP}(r|\mathbf{y})) = 0.00395.$$

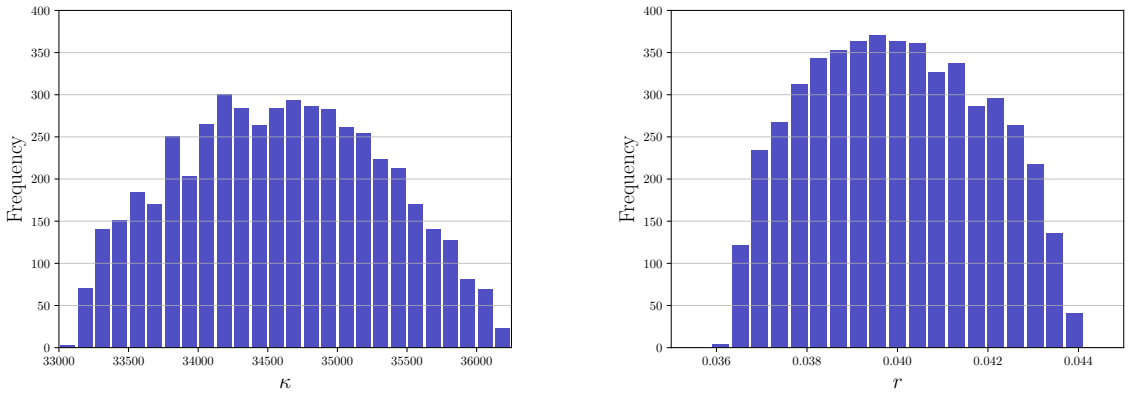
This suggests that substituting the ABC-MCMC distributions by their counterparts using ABC-MCMC with the GP surrogate does not result in considerable loss of information, as the values are smaller than 1×10^{-2}. As for the reduction in the mechanistic model runs, the GP surrogate was constructed using 140 runs, while the execution of the ABC-MCMC method with the true model required 140629 runs. Lastly, the mechanistic model predictions using the maximum *a posteriori* (MAP) estimates associated with both approaches are presented in Figure 66, indicating good agreement, differing in less than 2% in the carrying capacity. The MAP estimates are outlined in Table 6.

Parameter	**Unit**	**Prior**	**IC**	**Distance function**	ϵ
r	hr^{-1}	$\mathcal{U}(0.0, 0.3)$	$C_0 = 1700.0$	$\sqrt{\sum_{i=1}^{n} \frac{(y_i - y_i^*)^2}{\hat{\sigma}_i^2}}$	16.5
κ	cell density	$\mathcal{U}(3 \times 10^4, 3.7 \times 10^4)$			

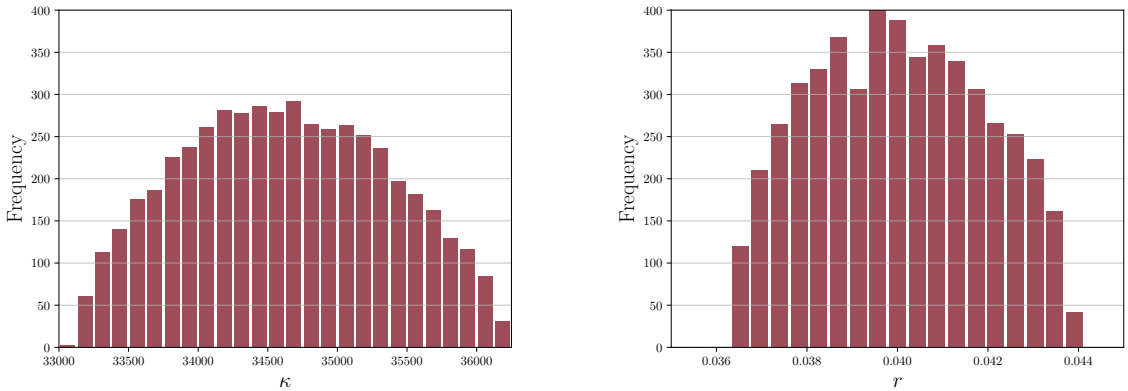
Table 5 – Bayesian inference set up for the Verhulst model parameters. The distance function is weighted by the empirical data standard deviations $\hat{\sigma}_1 \ldots \hat{\sigma}_n$.

Parameter	**ABC-MCMC**	**ABC-MCMC with GP**
r	0.03935	0.03935
κ	34125.0	34625.0

Table 6 – Maximum *a posteriori* estimate using ABC-MCMC and ABC-MCMC with GPR for the Verhulst model, in appropriate units (a.u.).



(a) Posterior distributions (ABC-MCMC).



(b) Posterior distributions (ABC-MCMC with GP surrogate).

Figure 65 – Posterior distributions for the parameters κ (carrying capacity) and r (maximum growth rate). The Kullback-Leibler divergences of the marginal distributions $D_{\mathrm{KL}}(\pi(\kappa|\mathbf{y}) \parallel \pi_{GP}(\kappa|\mathbf{y}))$ and $D_{\mathrm{KL}}(\pi(r|\mathbf{y}) \parallel \pi_{GP}(r|\mathbf{y}))$ are 0.004825 and 0.00395, respectively. The GP was constructed using a training set $\mathcal{D}$ of size 140. In constrast, the ABC-MCMC applied over the true model required 140629 runs.

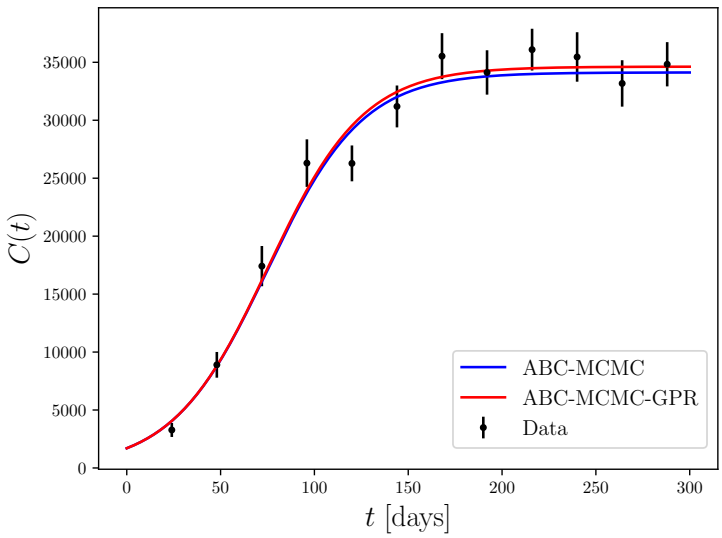


Figure 66 – Verhulst model predictions with the MAP estimates returned from ABC-MCMC and ABC-MCMC with GP surrogate approaches. The results are in accordance, with less than 2% difference in the carrying capacity.

5.2 Multiscale model (two-scale)

The other tumor growth model is presented in (Rocha, 2019). It is a hybrid two scale model, which aims to describe the dynamics involved in both tissue and cellular scales. The main features of the model are portrayed in Figure 67. Similar to the previous model, only cancer cells are considered. In the tissue scale, the (normalized) nutrient concentration ϕ disperses in the medium, and its availability is decisive for cell survival. Mathematically, for a given $\Omega \in \Re^2$ with a regular boundary $\partial\Omega$, this is represented by the following reaction-diffusion PDE:

$$\begin{cases} \dfrac{\partial \phi(\mathbf{x},t)}{\partial t} = \boldsymbol{\nabla} \cdot (D \boldsymbol{\nabla} \phi(\mathbf{x},t)) - \Lambda(\mathbf{x},t)\phi(\mathbf{x},t), \quad (\mathbf{x},t) \in \Omega \times (0,T], \\ \phi(\mathbf{x},0) = \bar{\phi}, \\ \boldsymbol{\nabla}\phi(\mathbf{x},t) \cdot \boldsymbol{n} = 0, \quad \mathbf{x} \in \partial\Omega. \end{cases} \tag{5.3}$$

The parameter D is a constant representing a hypothetical isotropic diffusion in the medium. Moreover, the term $\Lambda(\mathbf{x},t)$ stands for the nutrient uptake rate, which depends on the disposition of the cells in the medium at a certain time. This disposition is modeled by the Agent Based Model (ABM), which describes events and cell interactions in the cellular scale. We note that $\Lambda(\mathbf{x},t)$ is responsible for bridging the dynamics occurring at each scale. Since only viable cells uptake nutrients at a constant rate λ, $\Lambda(\mathbf{x},t)$ is the average consumption rate performed by these cells around the point $\mathbf{x}$ at time t. For more details on how $\Lambda(\mathbf{x},t)$ is actually evaluated, see (Rocha, 2019). As the initial condition for

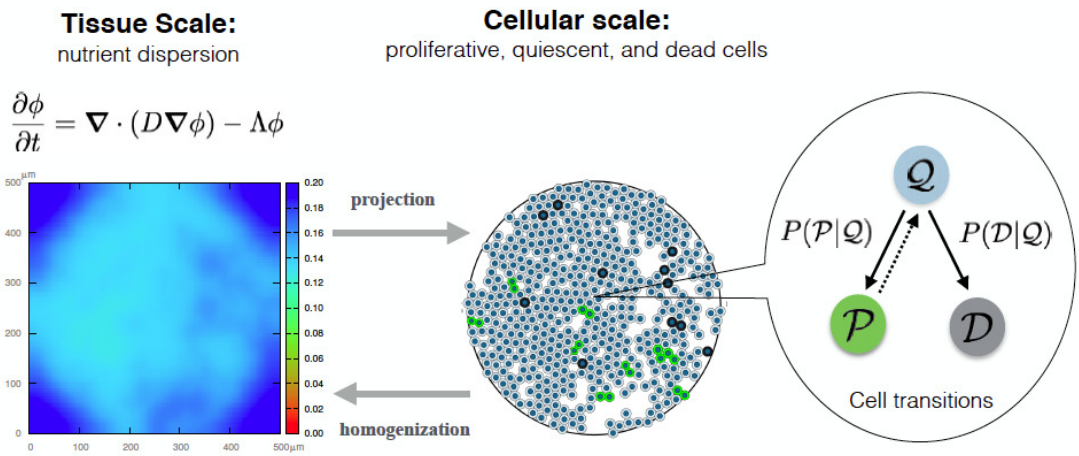


Figure 67 – Overview of the two scale model. The dynamics of the nutrient concentration in the tissue scale is governed by a reaction-diffusion PDE, while the cellular scale is modeled using the ABM model, encompassing proliferative ($\mathcal{P}$), quiescent ($\mathcal{Q}$), and dead ($\mathcal{D}$) cells. Cell transitions are shown in the quote bubble, in which black and dashed arrows indicate stochastic and deterministic transitions, respectively. Information between the two scales is carried out through projection and homogenization operators. See (Rocha, 2019) for details.

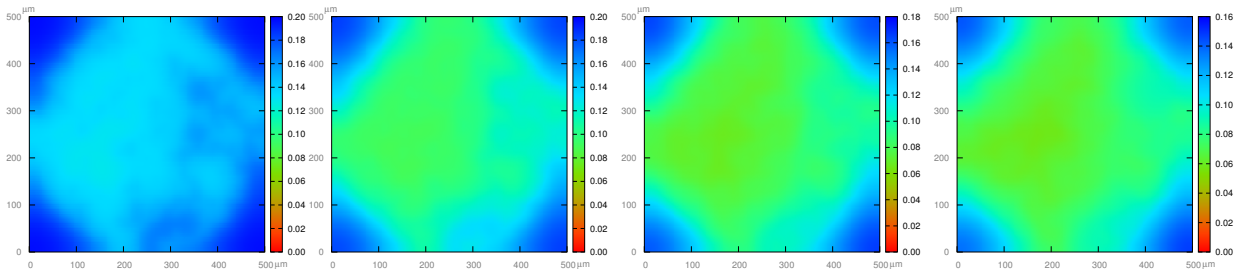
(5.3), it is assumed that the nutrient concentration is homogeneously dispersed with value $\bar{\phi}$ in the domain, a set up typically present in the initial stages of a tumor growth.

We mentioned that the ABM describes the tumor growth aspects in the cellular scale. The ABM consists of representing each cell as an individual in a lattice free context. All cells have a geometry consisting of a circular shape with a known radius. Their movement and interactions are modeled by a set of rules, detailed in (Rocha, 2019). The model also takes into account cell state transitions, as shown in the right panel of Figure 67, and we are specifically interested in tracing their evolutions. Tumor cells may be live or dead cells, and live cells may be proliferative or quiescent. Proliferative cells produce new cells by undergoing a cellular division process. This process is deterministic, that is concluded after a specified time in the model, with the cell becoming quiescent. On the other hand, transitions of quiescent cells to proliferative or dead cells are stochastic, defined by the following probability distributions (Macklin et al., 2012)

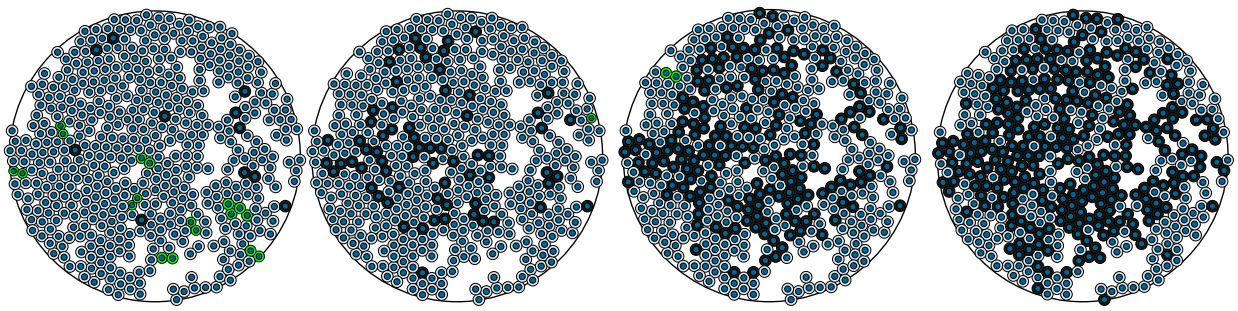
$$\mathcal{Q} \to \mathcal{P} \sim \mathrm{Exp}\left(\alpha_P \Delta t\right), \qquad \text{where } \alpha_P(t) = \max\left\{0, \bar{\alpha}_P \left(\frac{\phi - \phi_H}{1 - \phi_H}\right)\right\}, \tag{5.4}$$

$$\mathcal{Q} \to \mathcal{D} \sim \mathrm{Exp}\left(\alpha_D \Delta t\right), \qquad \text{where } \alpha_D(t) = \frac{\bar{\alpha}_D}{1 + \exp(\gamma_D(\phi - \phi_H))}. \tag{5.5}$$

In these expressions, $\bar{\alpha}_D$ is the apoptosis (programmed cell death) rate, $\bar{\alpha}_P$ is the proliferation rate, ϕ_H denotes the nutrient threshold for proliferation, and γ_D modulates the increase in cell death due to lack of nutrients. We may note that the intensities $\alpha_P(t)$ and $\alpha_D(t)$ depend on the (normalized) nutrient concentration (ϕ). If the difference between ϕ and ϕ_H is positive, there is a chance for a quiescent cell to become proliferative. On the



(a) Tissue scale: nutrient ϕ dispersion.



(b) Cellular scale dynamics.

Figure 68 – Snapshots of the model along 4 days. From left to right: simulations at $t = 1, 2, 3$, and 4 days.

contrary, when cells are subject to low levels of nutrient, $\phi - \phi_H$ assumes negative values, increasing the odds of having cell death. Setting Ω as a square domain of size 500μm and the ABM domain as a circle of radius 250μm, we may see a single model simulation in Figure 68.

Before discussing the challenges of performing parameter inference in the two-scale model, we describe the used empirical data. We actually used hypothetical *in vitro* data, typical from those obtained by live-cell imaging systems. They correspond to the confluences (area of the medium covered with a certain type of cell) of dead and live tumor cells every 3 hours, over 4 days. Here we used $\mathbf{y} = [\mathbf{C}_D, \mathbf{C}_L]$ as QoI, in which $\mathbf{C}_D, \mathbf{C}_L \in \mathbb{R}^{33}$ represent dead and live tumor cell confluences in the times $\{t_1, t_2, \ldots, t_{33}\}$ at which data is available.

It is worth commenting how the simulated QoI $\mathbf{y}^*$ is actually computed. By using an auxiliary uniform square mesh, the following ratios approximate dead and live cell confluences at a time t_i:

$$\mathbf{C}_D^*(t_i) = \frac{\#P_D(t_i)}{\#P_{\text{domain}}}, \quad \mathbf{C}_L^*(t_i) = \frac{\#P_L(t_i)}{\#P_{\text{domain}}}. \tag{5.6}$$

The values $\#P_D(t_i)$ and $\#P_L(t_i)$ are the number of mesh points that intercept dead and live cells at time t_i, respectively, while $\#P_{\text{domain}}$ is the total number of points in the mesh.

In the parameter estimation process, the described non deterministic factors of the model pose a difficulty. Considering that, for a certain parameter vector $\hat{\boldsymbol{\theta}}$, the model output $\mathcal{M}(\hat{\boldsymbol{\theta}})$ is stochastic, the simulated QoI $\mathbf{y}^*(\hat{\boldsymbol{\theta}})$ is not unique. Evidently, comparing $\mathbf{y}^*(\hat{\boldsymbol{\theta}})$ to $\mathbf{y}$ during a ABC-MCMC process would not be meaningful. Thus, we do not

consider a single model run $\mathcal{M}(\hat{\boldsymbol{\theta}})$, but rather the mean behavior of the model. It was presented in (Rocha, 2019) that the sample average obtained with 46 model runs is a good approximation for the true mean behavior of the model. From now on, we refer to a complete model sample this number of model evaluations. It is worth mentioning that this result only considers $\boldsymbol{\theta} = [\bar{\alpha}_D, \bar{\alpha}_P]$, and not the full parameter set.

It may seem questionable at first to focus on just two parameters, but this choice has the roots on parameter identifiability. It is well known that parameter estimation is subject to failure due to parameter non identifiability issues (Raue et al., 2013; Raue et al., 2009). The discussions of this problem are out of the scope of this work, although it is certainly something which dictates how to perform parameter inference. Since rates $\bar{\alpha}_D$ and $\bar{\alpha}_P$ determine the velocity of cell proliferation and death, these most likely influence the QoI $\mathbf{y}^*(\boldsymbol{\theta})$. This is suggested in (Rocha, 2019) after performing a sensitivity analysis using the Elementary effects method (Campolongo; Cariboni; Saltelli, 2007). Table 7 shows the fixed values of parameters λ, D, ϕ_H, and γ_D. Given the above, the hypothetical data used in the calibration were obtained by simulating the model only once using the parameters in Table 7, setting $\bar{\alpha}_D = 0.15$ hr^{-1} and $\bar{\alpha}_P = 0.2$ hr^{-1}, and adding a standard deviation of 0.03 to the resulting live and dead cell confluences.

Parameter	**Value**	**Unit**
λ	0.06588	hr^{-1}
D	4.64894	μm^{2}hr^{-1}
ϕ_H	0.14681	–
γ_D	100	–

Table 7 – Fixed parameters for the two-scale model.

We finally turn to the parameter estimation problem. The Bayesian inference is performed with the prior distributions $\bar{\alpha}_D \sim \mathcal{U}(0, 0.5)$ hr^{-1} and $\bar{\alpha}_P \sim \mathcal{U}(0, 0.5)$ hr^{-1}. We used the ABC-MCMC method with Euclidean distance, tolerance $\epsilon = 0.2$, and a GP surrogate with different model samples, from 100 to 1000 (recall that a model sample is composed of 46 model evaluations associated to parameter $\boldsymbol{\theta}$). The calibrated parameters are shown in Figure 69. The posterior distributions for 100 and 1000 samples are shown in Figure 70. For comparison, the posterior distributions obtained using ABC-MCMC are shown in Figure 71. To be obtained, the ABC-MCMC method required 2174 samples so as to get 1000 accepted samples. Finally, the simulated confluences are shown in Figure 72.

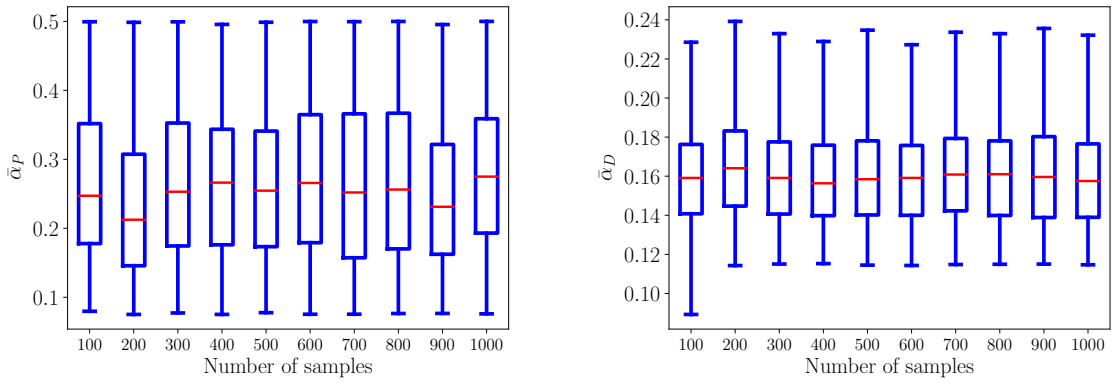


Figure 69 – Calibrated parameters $\bar{\alpha}_P$ (proliferation rate) and $\bar{\alpha}_D$ (apoptosis rate) using ABC-MCMC with GP surrogate with different number of samples.

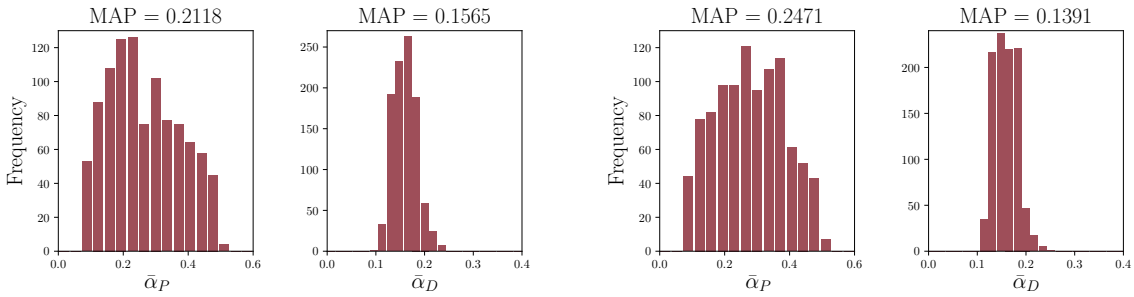


Figure 70 – Posterior distributions for the calibrated parameters obtained with ABC-MCMC and GP surrogate with 100 (left) and 1000 (right) samples.

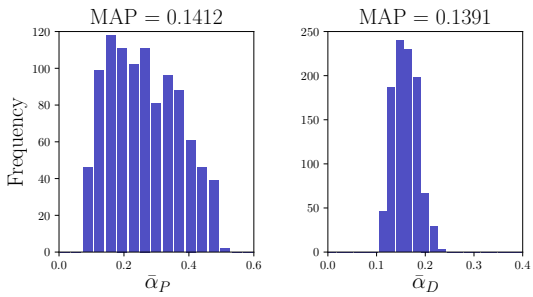


Figure 71 – Posterior distributions for the calibrated parameters obtained with ABC-MCMC method without using the GP surrogate.

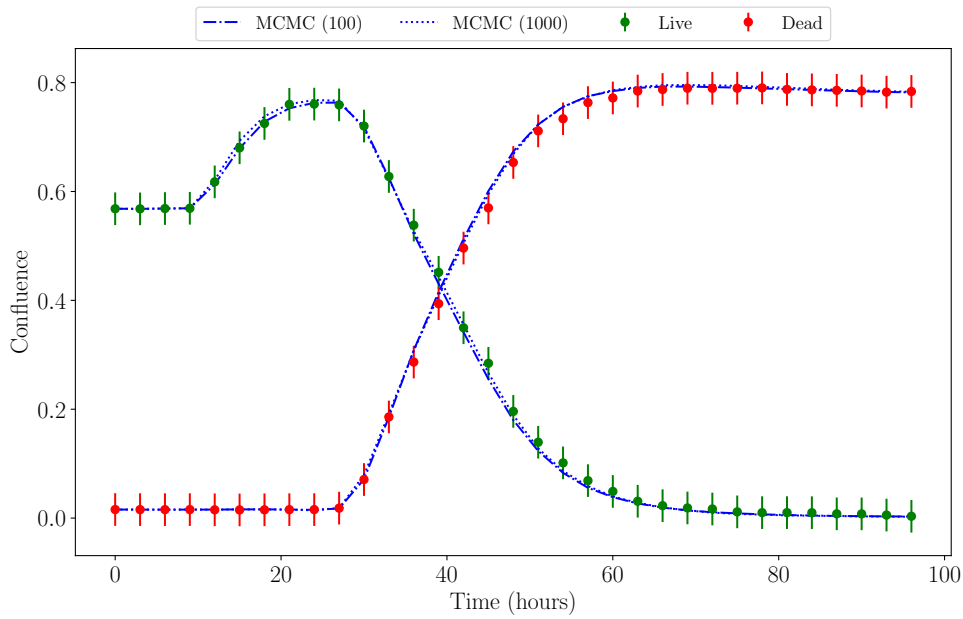


Figure 72 – Simulated live and dead cell confluences, using the MAP estimates obtained with the ABC-MCMC method with GP surrogate with 100 and 1000 samples. Red and green dots indicate the hypothetical dead and live experimental confluences, shown together with their corresponding error bars.

We consider that both approaches are in consonance, since the KL divergences of the marginal distributions of the ABC-MCMC method without and with the GP surrogate with 100 samples are

$$D_{\mathrm{KL}}(\pi(\bar{\alpha}_P|\mathbf{y}) \parallel \pi_{GP}(\bar{\alpha}_P|\mathbf{y})) = 0.01916 \quad \text{and} \quad D_{\mathrm{KL}}(\pi(\bar{\alpha}_D|\mathbf{y}) \parallel \pi_{GP}(\bar{\alpha}_D|\mathbf{y})) = 0.00713.$$

Nevertheless, it is important to highlight the computational cost associated with each approach. The ABC-MCMC method with a GP surrogate required at least 100 complete model runs. The ABC-MCMC method, on the other hand, required 2174 complete model runs. Each complete model run demands roughly 0.3 minutes (18 seconds), in a machine with configurations: Intel Core i7-4790 CPU, 3.6GHz $\times$ 8, 15.6GiB, NVIDIA GeForce GT 740. Even accounting the additional overhead of constructing and evaluating the GP surrogate model, the speedup is pretty expressive. The proposed approach demanded less than a day, while the other strategy had almost an entire week of evaluation time.

Finally, we should comment that an adaptive scheme may be used in the GP surrogate construction, as proposed in (Rocha, 2019). Its main idea is to subdivide the parametric space in regions to reduce their GP proxy estimates variance, before introducing it in the ABC-MCMC method. Such an approach is outlined in details in (Rocha, 2019).

5.3 Discussion

The present chapter showed results of the proposed methodology, indicating that the use of a GP surrogate model may be helpful in performing inference over costly mechanistic tumor models. First, we presented the process in the simple Verhulst tumor growth model, showing that the use of a GP surrogate did not compromise an Approximate Bayesian Inference procedure. A comparison between the posterior distributions, in terms of the Kullback-Leibler divergence, as well as the model predictions, indicated good agreement. We then applied the methodology to a computationally costly stochastic multiscale model, for which satisfactory results were also obtained in terms of the inference process.

We highlight that, depending on the model computational cost, the construction of a Gaussian process surrogate and its proxy estimates may be a good alternative whenever it is necessary to repeatedly evaluate the mechanistic model.

6 Conclusion

This work presented the main points of using Gaussian processes over general regression problems, as well as in metamodeling. This technique is used in a variety of contexts, such as machine learning, optimization, sensitivity analysis, and parameter inference of mechanistic problems. We started Chapter 2 defining regression problems, in which it is necessary to initially assume a form for the prediction and error models. We detailed the premises and how to solve such problems using Gaussian processes, highlighting the importance of their kernel (or covariance) functions. At first, we presented the mathematical expressions of kernel functions, along with their respective hyperparameters. By graphically depicting these functions and the Gaussian Process priors, we showed the effects of choosing a certain kernel function and changing the hyperparameter values in the defined Gaussian process. We reviewed both classical and more advanced forms present in the literature, emphasizing for each one which function type, in theory, may be learned successfully.

We continued solving in Chapter 3 several regression problems with distinct kernel functions, in order to evaluate the overall performance of kernels. We observed that, in accordance with the theory, squared exponential and Matérn kernels are able to perform good predictions in the interpolation range for sufficiently smooth data. We also verified that, while the spectral mixture kernel was able to perform good extrapolation in certain cases, the multimodality it causes in the likelihood function may result in inadequate predictions. This is also noticed in the highly complex deep kernels, which only demonstrated good interpolation and extrapolation capabilities in the example of a step function. We also detected the necessity of a higher computational cost in the process of hyperparemeter tuning for more complex kernels, due to higher dimensionality and presence of multiple local optima.

We then discussed in Chapter 4 the difficulties behind computational Bayesian inference methods, in particular the repeated mechanistic model evaluation. To overcome this difficulty, we considered the introduction of a data-driven surrogate model in the parameter estimation process. We used a GP surrogate model constructed using a dataset of model runs, in which we consider an anisotropic squared exponential kernel a good choice due to its nice interpolation properties. Our inference methodology was described in steps easily adapted to a mechanistic model of choice. We note that adaptive schemes may be used to further improve the overall process.

We applied the developed methodology in two distinct tumor growth models in Chapter 5. Starting with the famous ODE Verhulst model as a proof of concept, we

showed that the parameter inference matched the results obtained without a GP surrogate. As this model does not present a worrisome computational burden, we also applied the methodology to a multiscale tumor growth model. We noted that the non-deterministic factors, the cost of simulating two scales, and parameter non identifiability issues pose a great challenge in the process. Nevertheless, the comparison with a more standard approach showed an impressive speedup regarding the necessary computational time.

As for future works, we aim to investigate strategies that further improve the process of construction and prediction performed by Gaussian Process, specially in contexts with considerable amount of data. This is a common scenario in machine learning problems, as well as construction of surrogates of models with high dimensional parametric spaces. Thus, we are interested in investigating techniques which approximate the Gaussian Process matrix and iterative linear solvers. Moreover, an inspection of adaptive schemes for metamodeling with GP is also considered. With a better understanding of the mentioned topics, as well as the development of additional methods, we expect to improve the use of Gaussian Process as a tool for solving scientific problems, with a focus on its use at alleviating the cost of solving difficult parameter inference problems.

Bibliography

Abramowitz, M.; Stegun, I. A. *Handbook of mathematical functions with formulas, graphs, and mathematical table*. 1965. Cited in page 30.

Alber, M. et al. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ digital medicine*, Nature Publishing Group, v. 2, n. 1, p. 1–11, 2019. Cited in page 16.

Altrock, P. M.; Liu, L. L.; Michor, F. The mathematics of cancer: integrating quantitative models. *Nature Reviews Cancer*, v. 15, p. 730–745, 2015. Cited in page 81.

Andrieu, C.; Thoms, J. A tutorial on adaptive MCMC. *Statistics and computing*, Springer, v. 18, n. 4, p. 343–373, 2008. Cited in page 83.

Beaumont, M. A. Approximate Bayesian Computation. *Annual review of statistics and its application*, Annual Reviews, v. 6, p. 379–403, 2019. Cited 2 times in pages 83 and 84.

Bekisz, S.; Geris, L. Cancer modeling: From mechanistic to data-driven approaches, and from fundamental insights to clinical applications. *Journal of Computational Science*, Elsevier, v. 46, p. 101198, 2020. Cited in page 81.

Bellomo, N.; Li, N. K.; Maini, P. K. On the foundations of cancer modelling: selected topics, speculations and perspectives. *Mathematical Models and Methods in Applied Sciences*, v. 18, n. 4, p. 593–646, 2008. Cited in page 81.

Betancourt, M. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017. Cited in page 83.

Bhosekar, A.; Ierapetritou, M. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, Elsevier, v. 108, p. 250–267, 2018. Cited in page 85.

Blei, D. M.; Kucukelbir, A.; McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, Taylor & Francis, v. 112, n. 518, p. 859–877, 2017. Cited in page 85.

Blum, M.; Riedmiller, M. A. Optimization of gaussian process hyperparameters using rprop. In: Citeseer. *ESANN*. 2013. p. 339–344. Cited in page 33.

Bonilla, E. V.; Chai, K. M.; Williams, C. Multi-task Gaussian process prediction. In: *Advances in neural information processing systems*. 2008. p. 153–160. Cited in page 40.

Butler, A. et al. Efficient optimization of the likelihood function in Gaussian process modelling. *Computational Statistics & Data Analysis*, Elsevier, v. 73, p. 40–52, 2014. Cited in page 33.

Campolongo, F.; Cariboni, J.; Saltelli, A. An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, Elsevier, v. 22, n. 10, p. 1509–1518, 2007. Cited in page 96.

Cheng, K. et al. Surrogate-assisted global sensitivity analysis: an overview. *Structural and Multidisciplinary Optimization*, Springer, v. 61, n. 3, p. 1187–1213, 2020. Cited in page 85.

Chiplunkar, A.; Bosco, E.; Morlier, J. Gaussian process for aerodynamic pressures prediction in fast fluid structure interaction simulations. In: Springer. *World Congress of Structural and Multidisciplinary Optimisation*. 2017. p. 221–233. Cited 3 times in pages 17, 19, and 85.

Collis, J. et al. Bayesian calibration, validation and uncertainty quantification for predictive modelling of tumour growth: a tutorial. *Bulletin of mathematical biology*, Springer, v. 79, n. 4, p. 939–974, 2017. Cited 2 times in pages 15 and 81.

Dick, T.; Wong, E.; Dann, C. How many random restarts are enough. 2014. Cited in page 33.

Do, C. B. *Gaussian processes*. 2008. H. Lee, Ed., Stanford University, Stanford. Cited in page 21.

Duvenaud, D. *Automatic model construction with Gaussian processes*. Thesis (PhD Thesis) — University of Cambridge, 2014. Cited 4 times in pages 19, 34, 36, and 42.

Freitas, R. S. et al. An encoder-decoder deep surrogate for reverse time migration in seismic imaging under uncertainty. *arXiv preprint arXiv:2006.09550*, 2020. Cited in page 85.

Gamerman, D.; Lopes, H. F. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC, 2006. Cited 3 times in pages 15, 81, and 83.

Garbuno-Inigo, A.; DiazDelaO, F. A.; Zuev, K. Gaussian process hyper-parameter estimation using parallel asymptotically independent Markov sampling. *Computational Statistics & Data Analysis*, Elsevier, v. 103, p. 367–383, 2016. Cited in page 33.

Gardner, J. et al. Gpytorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration. In: *Advances in Neural Information Processing Systems*. 2018. p. 7576–7586. Cited in page 43.

Giordano, F. R.; Fox, W. P.; Horton, S. B. *A first course in mathematical modeling*. Cengage Learning, 2013. Cited in page 15.

Goodfellow, I.; Bengio, Y.; Courville, A. *Deep learning*. MIT press, 2016. Cited 4 times in pages 38, 42, 82, and 87.

Gorbach, N. S. et al. Model selection for gaussian process regression. In: Springer. *German Conference on Pattern Recognition*. 2017. p. 306–318. Cited in page 42.

Gustafsson, B. *Scientific Computing: A Historical Perspective*. Springer, 2018. v. 17. Cited in page 15.

Hansen, P. C.; Pereyra, V.; Scherer, G. *Least squares data fitting with applications*. JHU Press, 2013. Cited in page 15.

Hastie, T.; Tibshirani, R.; Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. Cited 2 times in pages 19 and 26.

Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. Oxford University Press, 1970. Cited in page 82.

Heath, M. T. *Scientific Computing: An Introductory Survey, Revised Second Edition*. SIAM, 2018. Cited in page 15.

Higham, C. F.; Higham, D. J. Deep learning: An introduction for applied mathematicians. *SIAM Review*, SIAM, v. 61, n. 4, p. 860–891, 2019. Cited 2 times in pages 38 and 39.

Hoffman, M. D.; Gelman, A. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, v. 15, n. 1, p. 1593–1623, 2014. Cited in page 83.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, Elsevier, v. 4, n. 2, p. 251–257, 1991. Cited in page 27.

Jackson, T. L. et al. Cancer modeling: A perspective on what's new and what's next. *Contemporary Mathematics*, v. 40, p. 229–234, 2006. Cited in page 81.

Jamil, M.; Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, Inderscience Publishers Ltd, v. 4, n. 2, p. 150–194, 2013. Cited in page 68.

Keeling, C. D. et al. Exchanges of atmospheric CO2 and 13CO2 with the terrestrial biosphere and oceans from 1978 to 2000. i. global aspects. 2001. Cited in page 63.

Kendall, B. E. et al. Why do populations cycle? A synthesis of statistical and mechanistic modeling approaches. *Ecology*, Wiley Online Library, v. 80, n. 6, p. 1789–1805, 1999. Cited in page 16.

Kiessling, J.; Ström, E.; Tempone, R. Wind field reconstruction with adaptive random Fourier features. *arXiv preprint arXiv:2102.02365*, 2021. Cited in page 85.

Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited in page 42.

Korolev, I. Identification and estimation of the SEIRD epidemic model for covid-19. *Journal of econometrics*, Elsevier, v. 220, n. 1, p. 63–85, 2021. Cited in page 16.

Koziel, S.; Ciaurri, D. E.; Leifsson, L. *Surrogate-based methods*. Springer, 2011. 33–59 p. Cited in page 85.

Krige, D. G. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, Southern African Institute of Mining and Metallurgy, v. 52, n. 6, p. 119–139, 1951. Cited 2 times in pages 16 and 19.

Kruschke, J. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press, 2014. Cited 2 times in pages 15 and 81.

Lalchand, V.; Rasmussen, C. E. Approximate inference for fully Bayesian Gaussian process regression. In: *Symposium on Advances in Approximate Bayesian Inference*. 2020. p. 1–12. Cited in page 33.

Li, Z.; Arora, S. An exponential learning rate schedule for deep learning. *arXiv preprint arXiv:1910.07454*, 2019. Cited in page 33.

Lima, E. A. B. F.; Oden, J. T.; Almeida, R. C. A hybrid ten-species phase-field model of tumor growth. *Mathematical Models and Methods in Applied Sciences*, v. 24, n. 13, p. 2569–2599, 2014. Cited in page 81.

Liu, H. et al. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, 2020. Cited 2 times in pages 16 and 19.

Lowengrub, J. S. et al. Nonlinear modelling of cancer: bridging the gap between cells and tumours. *Nonlinearity*, v. 23, n. 1, p. R1–R9, 2009. Cited in page 81.

MacKay, D. J. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, Springer Verlag, v. 168, p. 133–166, 1998. Cited in page 30.

Macklin, P. et al. Patient-calibrated agent-based modelling of ductal carcinoma *in situ* (DCIS): From microscopic measurements to macroscopic predictions of clinical progression. *Journal of Theoretical Biology*, v. 301, p. 122–140, 2012. Cited in page 94.

Marjoram, P. et al. Markov Chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 100, n. 26, p. 15324–15328, 2003. Cited in page 84.

McHutchon, A.; Rasmussen, C. E. Gaussian process training with input noise. In: *Advances in Neural Information Processing Systems*. 2011. p. 1341–1349. Cited in page 33.

McKenna, M. T. *Development and Validation of a Predictive Model of Chemotherapy in Triple Negative Breast Cancer*. Thesis (PhD Thesis) — Vanderbilt University, 2017. Cited in page 90.

Micchelli, C. A.; Xu, Y.; Zhang, H. Universal kernels. *Journal of Machine Learning Research*, v. 7, n. Dec, p. 2651–2667, 2006. Cited in page 27.

Murphy, H.; Jaafari, H.; Dobrovolny, H. M. Differences in predictions of ode models of tumor growth: a cautionary example. *BMC cancer*, BioMed Central, v. 16, n. 1, p. 1–10, 2016. Cited in page 16.

Murray, I. *Machine Learning and Pattern Recognition (Lecture notes)*. The University of Edinburgh, 2016. Available on: <https://www.inf.ed.ac.uk/teaching/courses/mlpr/2016/notes/w7c_gaussian_process_kernels.html>. Cited in page 27.

Oden, J. T. et al. Toward predictive multiscale modeling of vascular tumor growth - computational and experimental oncology for tumor prediction. *Archives of Computational Methods in Engineering*, v. 23, n. 4, p. 735–779, 2016. Cited in page 81.

Peng, G. C. et al. Multiscale modeling meets machine learning: What can we learn? *Archives of Computational Methods in Engineering*, Springer, v. 28, n. 3, p. 1017–1037, 2021. Cited in page 16.

Pérez, D.; Quintana, Y. A survey on the Weierstrass approximation theorem. *arXiv preprint math/0611038*, 2006. Cited in page 27.

Petelin, D.; Filipič, B.; Kocijan, J. Optimization of gaussian process models with evolutionary algorithms. In: Springer. *International Conference on Adaptive and Natural Computing Algorithms*. 2011. p. 420–429. Cited in page 33.

Pinsky, M. A. *Introduction to Fourier analysis and wavelets*. American Mathematical Soc., 2008. v. 102. Cited in page 58.

Pleiss, G. et al. Constant-time predictive distributions for gaussian processes. *arXiv preprint arXiv:1803.06058*, 2018. Cited in page 41.

Pomerantsev, A. L. Confidence intervals for nonlinear regression extrapolation. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 49, n. 1, p. 41–48, 1999. Cited in page 19.

Pošík, P.; Huyer, W. Restarted local search algorithms for continuous black box optimization. *Evolutionary computation*, MIT Press, v. 20, n. 4, p. 575–607, 2012. Cited in page 33.

Preuss, R.; Toussaint, U. V. Global optimization employing Gaussian process-based bayesian surrogates. *Entropy*, Multidisciplinary Digital Publishing Institute, v. 20, n. 3, p. 201, 2018. Cited 4 times in pages 17, 19, 68, and 85.

Rakitsch, B. et al. It is all in the noise: Efficient multi-task Gaussian process inference with structured residuals. In: *Advances in neural information processing systems*. 2013. p. 1466–1474. Cited in page 41.

Rasmussen, C. E.; Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006. Cited 4 times in pages 16, 19, 30, and 34.

Raue, A. et al. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, Oxford University Press, v. 25, n. 15, p. 1923–1929, 2009. Cited in page 96.

Raue, A. et al. Joining forces of Bayesian and Frequentist methodology: a study for inference in the presence of non-identifiability. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society Publishing, v. 371, n. 1984, p. 20110544, 2013. Cited in page 96.

Razavi, S.; Tolson, B. A.; Burn, D. H. Review of surrogate modeling in water resources. *Water Resources Research*, Wiley Online Library, v. 48, n. 7, 2012. Cited in page 85.

Rio-Chanona, E. A. D. et al. Comparison of physics-based and data-driven modelling techniques for dynamic optimisation of fed-batch bioprocesses. *Biotechnology and bioengineering*, Wiley Online Library, v. 116, n. 11, p. 2971–2982, 2019. Cited in page 81.

Robert, C.; Casella, G. A short history of Markov Chain Monte Carlo: Subjective recollections from incomplete data. *Statistical Science*, JSTOR, p. 102–115, 2011. Cited 2 times in pages 82 and 83.

Rocha, H. L. *Continuous-Discrete Approach of Tumor Growth: Multiscale Modeling and Bayesian Inference using Gaussian Process Surrogates*. Thesis (PhD Thesis) — Laboratório Nacional de Computação Científica, 2019. Cited 6 times in pages 81, 89, 93, 94, 96, and 98.

Rockne, R. C. et al. The 2019 mathematical oncology roadmap. *Physical Biology*, v. 16, n. 4, p. 041005, 2019. Cited in page 81.

Rosenthal, J. S. et al. Optimal proposal distributions and adaptive MCMC. *Handbook of Markov Chain Monte Carlo*, Chapman & Hall/CRC Boca Raton, FL, v. 4, n. 10.1201, 2011. Cited in page 83.

Rossum, G. V.; Drake, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697. Cited in page 42.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Cited in page 42.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015. Cited in page 38.

Schütze, H.; Manning, C. D.; Raghavan, P. *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008. v. 39. Cited in page 91.

Shlens, J. Notes on Kullback-Leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014. Cited in page 90.

Simonoff, J. S. Smoothing categorical data. *Journal of Statistical Planning and Inference*, Elsevier, v. 47, n. 1-2, p. 41–69, 1995. Cited in page 91.

Singer, N. F. Y. Efficient bayesian parameter estimation in large discrete domains. *Advances in neural information processing systems*, v. 11, p. 417, 1999. Cited in page 91.

Sisson, S. A.; Fan, Y.; Beaumont, M. A. Overview of approximate bayesian computation. *ArXiv e-prints*, 2018. Cited in page 84.

Stein, M. L. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012. Cited in page 30.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In: *Artificial Intelligence and Statistics*. 2009. p. 567–574. Cited in page 41.

Tong, Y. L. *The multivariate normal distribution*. Springer Science & Business Media, 2012. Cited in page 21.

Trefethen, L. N.; III, D. B. *Numerical linear algebra*. Siam, 1997. v. 50. Cited in page 41.

Tripathy, R. K.; Bilionis, I. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of computational physics*, Elsevier, v. 375, p. 565–588, 2018. Cited in page 85.

Tsoularis, A.; Wallace, J. Analysis of logistic growth models. *Mathematical biosciences*, Elsevier, v. 179, n. 1, p. 21–55, 2002. Cited in page 16.

Velten, K. *Mathematical modeling and simulation: introduction for scientists and engineers*. John Wiley & Sons, 2009. Cited 3 times in pages 15, 16, and 81.

Volpatto, D. T. et al. A generalized SEIRD model with implicit social distancing mechanism: a Bayesian approach for the identification of the spread of COVID-19 with applications in Brazil and Rio de Janeiro state. *medRxiv*, Cold Spring Harbor Laboratory Press, 2020. Available on: <https://www.medrxiv.org/content/early/2020/10/12/2020.05.30.20117283>. Cited in page 16.

Wang, B.; Chen, T. Gaussian process regression with multiple response variables. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 142, p. 159–165, 2015. Cited 2 times in pages 40 and 73.

Wasserman, L. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013. Cited 2 times in pages 83 and 90.

Wilson, A. G. *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. Thesis (PhD Thesis) — University of Cambridge, 2014. Cited 5 times in pages 27, 31, 33, 36, and 38.

Wilson, A. G. et al. Deep kernel learning. In: *Artificial intelligence and statistics*. 2016. p. 370–378. Cited 3 times in pages 38, 39, and 59.

Wolberg, J. *Data analysis using the method of least squares: extracting the most information from experiments*. Springer Science & Business Media, 2006. Cited in page 19.

Yang, Z. et al. A surrogate-assisted particle swarm optimization algorithm based on efficient global optimization for expensive black-box problems. *Engineering Optimization*, Taylor & Francis, v. 51, n. 4, p. 549–566, 2019. Cited 2 times in pages 68 and 85.

Yankeelov, T. E. et al. Multi-scale modeling in clinical oncology: Opportunities and barriers to success. *Annals of Biomedical Engineering*, v. 44, n. 9, p. 2626–2641, 2016. Cited in page 81.

Zhang, J.; Yin, J.; Wang, R. Basic framework and main methods of uncertainty quantification. *Mathematical Problems in Engineering*, Hindawi, v. 2020, 2020. Cited 2 times in pages 15 and 16.