# Process-centric Gaussian Processes (GPs)

**Ann-Kristin Balve**

Supervisor: Prof. Carl Rasmussen

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
*Master of Philosophy in Machine Learning and Machine Intelligence*

Christ's College August 2024

I would like to dedicate this thesis to my loving parents who always support me and made this MPhil possible.

## Declaration

I, Ann-Kristin Balve of Christ's College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. All software was written from scratch using Python 3.12. This dissertation contains 10280 words including tables, footnotes, figure captions and appendices, but excluding declarations, bibliography, photographs, and diagrams.

Ann-Kristin Balve
August 2024

# Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Carl Edward Rasmussen, for his patient guidance, constant feedback, and invaluable insights into Gaussian processes. I also want to thank Dr.John Dudley for his valuable help in developing a small pilot study to test my software and his advice for the user-centred design.

Furthermore, I want to acknowledge Cambridge Trust and DeepMind who gave me the financial resources to pursue my Masters studies in Cambridge. I specifically want to thank my DeepMind mentor Nimrod Gileadi who offered me feedback the whole year long during regular meetings.

Finally, I want to thank specifically my parents who always supported me during this masters and believed in me, and my friends Jack and Neela for making university life much more enjoyable.

# Abstract

Gaussian processes (GPs) are flexible and powerful probabilistic models that quantify uncertainty in predictions. Nevertheless, their abstract nature often makes them challenging to understand. This thesis introduces a new, conceptually simpler 'process-centric' view, which redefines a Gaussian process as an object fully specified by a mean and covariance function (along with its hyperparameters) that permits two main operations: conditioning and marginalization. The process-centric view provides an intuitive framework that simplifies learning and offers high flexibility in practical applications, such as Bayesian online learning. To enhance the utility of the framework, a novel interactive Jupyter Notebook tutorial was developed during this thesis, which introduces Gaussian processes through the process-centric lens. This tutorial is designed to be an effective educational tool, enhancing the accessibility and learning experience of Gaussian processes to a broad audience. While the process-centric view is demonstrated using exact Gaussian processes, future work could extend this work to approximate Gaussian processes.

# Table of contents

# List of figures

# List of tables

# Nomenclature

Vectors are denoted by boldface (**m**), matrices by capital boldface (**K**). Functions are written in italicised font (*m* and *k*). Individual elements of vectors are denoted by subscripts, ($\mathbf{x_i}$). Reference to the test set is made by subscript asterisk ($\mathbf{x}_*$).

| | |
|---|---|
| $\ell$ | characteristic lengthscale |
| $\mathbf{f}$ or $f(\mathbf{x})$ | vectorized latent function values |
| $\mathcal{D}$ | data set: $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, N\}$ |
| $\sigma_f^2$ | signal variance |
| $\sigma_n^2$ | noise variance |
| $D$ | dimension of the input space |
| $f$ or $f(x)$ | a function |
| $k(\mathbf{x}, \mathbf{x}')$ | covariance function evaluated at $\mathbf{x}$ and $\mathbf{x}'$ |
| $m(\mathbf{x})$ | mean function evaluated at $\mathbf{x}$ |
| $N$ | number of training points |

# Chapter 1

# Introduction

Gaussian Processes (GPs) are probability distributions over functions, commonly used for regression and classification problems by incorporating prior knowledge. They provide a probabilistic framework that quantifies uncertainty in predictions by assigning probabilities to an infinite array of potential functions that fit given data points. GPs are applied in various domains, such as error reduction in reinforcement learning (Deisenroth, Fox, and Rasmussen, 2013), the analysis of astronomical time-series data (Foreman-Mackey et al., 2017), or black-box optimization in Machine Learning (Snoek et al., 2012). These applications demonstrate the flexibility of GPs in addressing complex real-world problems. However, Gaussian processes are often perceived as difficult to understand due to their mathematical complexity and abstract nature. Despite the availability of foundational texts that offer deep insights into GPs (Rasmussen, 1997; Williams and Rasmussen, 2006), these resources may be less accessible to practitioners with limited mathematical backgrounds.

We address this challenge by developing a novel view on Gausssian processes that we call 'process-centric'. The process-centric view is introduced by Hensman and Rasmussen (2024) which sets a foundational starting point for this project. This new view defines a GP as an object defined by a mean and covariance function and permits two operations: marginalization and conditioning. The main contributions of this thesis for the Gaussian Process literature include:

- **The Development of the conceptual Framework**: This thesis introduces the process-centric view from a theoretical perspective, highlights key properties, and sets it in relation to previous views.
- **Practical Exposition of Examples**: In addition to a theoretical process-centric framework, this thesis delivers an interactive software that serves as a pedagogical tool to

improve the conceptual understanding of process-centric GPs through visualisations and practical examples.

## 1.1 Thesis Structure

This thesis is structured as follows. Chapter 2 introduces the key concepts: the multivariate Gaussian distribution, its main properties, and stochastic processes to lay the theoretical foundations for the process-centric view. Chapter 3 develops the process-centric view on GPs, focusing on the role of mean and covariance functions, and explaining key operations of marginalization and conditioning. The chapter is concluded by a discussion of the role of the marginal likelihood for model selection. Chapter 4 places the process-centric view within a broader context of the Gaussian process literature, and motivates the need for more accessible educational resources. Chapter 5 describes the design and implementation of the interactive Jupyter Notebook that was developed as part of this research, highlighting its key features and role as an educational tool. Finally, chapter 6 summarizes key findings, discusses the limitations of the current approach, and suggests directions for future work.

# Chapter 2

# Background

This chapter describes the theoretical background for process-centric Gaussian processes. It begins with the problem formulation and an explanation of the multivariate Gaussian distribution, including its properties. It then discusses stochastic processes, concluded with a brief introduction to Gaussian processes.

## 2.1 Regression Problem Formulation

Let $\mathcal{D}$ be a dataset of $N$ data points, $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, N\}$, where $\mathbf{x}_i \in \mathbb{R}^D$ is a $D$-dimensional input vector and $y_i \in \mathbb{R}$ is a scalar output. Suppose that the observed output $y_i$ is a function of the input $\mathbf{x}_i$ and additive Gaussian noise $\varepsilon_i$. The goal is to infer the function $f$ and make a prediction $y_*$ for a new input value $\mathbf{x}_*$.

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma_\text{n}^2) \tag{2.1}$$

We can vectorize the inputs $\mathbf{X}$, observations $\mathbf{y}$, and function values $\mathbf{f} = f(\mathbf{X})$. Then, the likelihood $p(\mathbf{y} \mid \mathbf{f})$ which expresses the probability of the observations under the model can be computed using:

$$p(\mathbf{y} \mid \mathbf{f}) = \prod_{i=1}^{N} p(y_i \mid f(\mathbf{x}_i)) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma_n^2 \mathbf{I}) \tag{2.2}$$

## 2.2 Bayesian Machine Learning

Traditional regression aims to obtain a single best solution to the problem in Equation 2.1 that maximises the fit to the training data. The Bayesian view takes an alternative approach of learning from data by assigning probabilities to different solutions which quantify our uncertainty about the learned relationship. For example, multiple models could explain the data, but a simpler model might be more probable than a complex one. Let us specify our model to be the function $f$. Then $p(f)$ is the prior probability distribution that expresses our prior beliefs about possible functions values. As data is observed, we can update our knowledge which is captured in the posterior distribution $p(f \mid \mathbf{y})$. Using Bayes' rule, the posterior is obtained by combining the prior with the likelihood function which is formalised in the following expression:

$$p(f,\mathbf{y}) = p(f)p(\mathbf{y} \mid \mathbf{f}) = p(\mathbf{y})p(f \mid \mathbf{y}) \tag{2.3}$$

Note that the normalisation constant $p(\mathbf{y})$, also known as the marginal likelihood, ensures that the posterior is a valid probability distribution. In the following, the Gaussian distribution will be discussed which lays the foundation for Gaussian processes.

## 2.3 Multivariate Gaussian Distribution

The multivariate Gaussian distribution of a $D$-dimensional vector of real-valued random variables $\mathbf{x} = (x_1, x_2, \cdots, x_D)^T$ is given as:

$$p(\mathbf{x}|\mathbf{m},\Sigma) = N(\mathbf{x}|\mathbf{m},\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m})\right) \tag{2.4}$$

It is fully specified by a mean vector $\mathbf{m}$ and covariance matrix $\Sigma$:

$$E[\mathbf{x}] = \mathbf{m} \quad \text{and} \quad E[(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T] = \Sigma \tag{2.5}$$

The covariance $\Sigma \in \mathbb{R}^{D\times D}$ is a symmetric positive definite matrix and determines the distribution's shape and orientation while the mean vector $\mathbf{m} \in \mathbb{R}^D$ is the center of the distribution. For the bivariate Gaussian distribution which is a special case of the multivariate Gaussian with $D = 2$, the random variables $x_1$ and $x_2$ are jointly Gaussian:

$$p(x_1,x_2) = p\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} m_1 \\ m_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & c \\ c & \sigma_2^2 \end{bmatrix}\right) \tag{2.6}$$

Figure 2.1 shows the probability density of a bivariate Gaussian distribution, visualised using a 3D surface plot. In the following sections, two important properties of the Gaussian

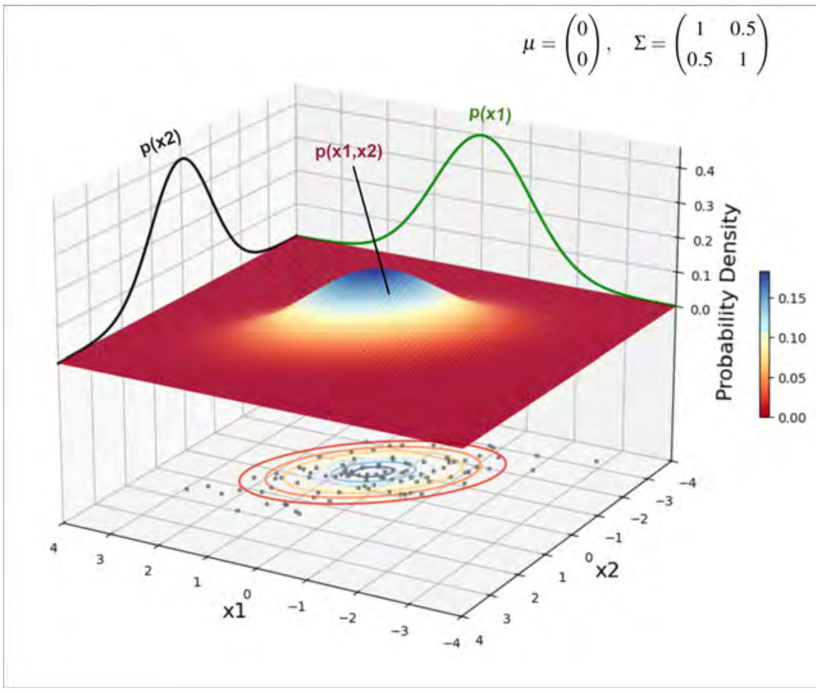


Fig. 2.1 3D visualization of the joint bivariate Gaussian distribution $p(x_1, x_2)$ with zero mean, unit variance, and a correlation coefficient $c = 0.5$. The marginal distributions $p(x_1)$ and $p(x_2)$ are shown in green and black, respectively. Brown dots indicate samples drawn from the distribution.

distribution will be discussed which are closure under marginalisation and conditioning. These properties are vital for Gaussian processes as they make them tractable. We will specifically discuss the bivariate case for ease of visualisation.

### 2.3.1 Property I: Closure Under Marginalisation

If a set of variables $x_1, x_2, \ldots, x_D$ is jointly Gaussian distributed, the marginal distribution of any subset, such as $p(x_1)$, can be obtained by integrating out variables we are not interested in. This process simplifies to considering only the relevant entries in the mean vector **m** and covariance matrix $\Sigma$. For the bivariate case, the marginal distribution of $x_1$ simplifies to a univariate Gaussian distribution, given as:

$$p(x_1) = \int p(x_1, x_2)\, dx_2 = \mathcal{N}(m_1, \sigma_1^2) \tag{2.7}$$

This property, known as closure under marginalization, implies that the resulting marginal distribution remains Gaussian. Thus, the marginalization of a bigger Gaussian distribution does not change the Gaussian nature of the smaller subset. The marginal distributions for a bivariate case are depicted in green and black in Figure 2.1 and Figure 2.2. The latter figure furthermore illustrates that the marginal $p(x_1)$ is independent of the correlation coefficient $c$ which also follows from Equation 2.7.

### 2.3.2 Property II: Closure Under Conditioning

Conditioning updates the Gaussian distribution based on observed values of some variables. The resulting distribution remains Gaussian, which is known as closure under conditioning. For jointly Gaussian variables $x_1$ and $x_2$, the conditional distribution of $x_1$ given $x_2$ is computed as:

$$p(x_1 \mid x_2) = \mathcal{N}\left(m_1 + c\sigma_2^{-2}(x_2 - m_2), \sigma_1^2 - c\sigma_2^{-2}c\right) \tag{2.8}$$

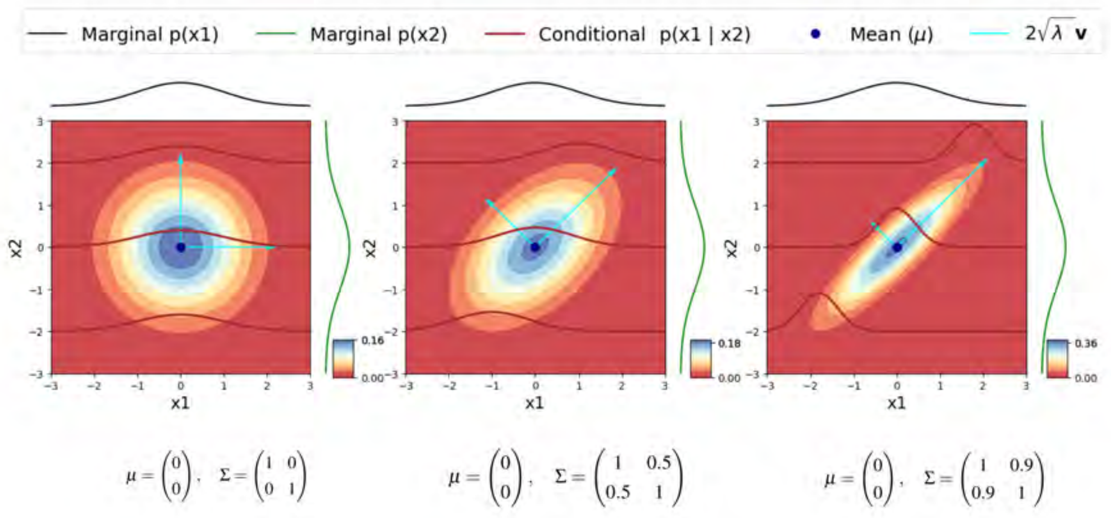


Fig. 2.2 Contour plots of bivariate Gaussian distributions with unit variance and zero mean for correlation coefficient $c = 0$ (left), $c = 0.5$ (center), and $c = 0.9$ (right). The joint density is shown using a red-to-blue heatmap with elliptical contour lines. Marginals for $p(x_1)$ and $p(x_2)$ are shown in black and green, respectively. Each light-blue arrow represents the unit eigenvector $\mathbf{v}$ of the covariance matrix scaled by the factor $2\sqrt{\lambda}$ where $\lambda$ is the corresponding eigenvalue. Brown lines show conditional distributions $p(x_1 \mid x_2)$ for different values of $x_2$

Both marginalizing and conditioning a bivariate Gaussian result in a univariate Gaussian distribution. However the difference is that conditioning fixes the value of one dimension while marginalization simply ignores that dimension. An illustration of conditional distributions for three covariance matrices with different correlation values $c$ can be seen in

Figure 2.2. For uncorrelated variables ($c = 0$), the conditional distribution is equivalent to the marginal in Equation 2.7. In contrast, highly correlated variables ($c = 0.9$) yield a more peaked conditional distribution with a smaller variance. This reflects a reduction in uncertainty because observing $x_2$ provides valuable information about the distribution of $x_1$. Notably, while the observed value of $x_2$ shifts the conditional mean, it does not affect the conditional variance that is solely determined by the covariance of the joint distribution.

## 2.4 Stochastic Process

While probability distributions, such as the multivariate Gaussian, characterise a finite set of random variables, many real-world systems that evolve over a continuous index set, such as time, are best modelled using functions. Such systems can be described using a stochastic process which is a collection of random variables $\{X(t) : t \in T\}$ indexed by parameter $t$ that belongs to some index set $T$. One example is Brownian motion which models the random motion of particles in a fluid medium while randomly bumping into other particles (Uhlenbeck and Ornstein, 1930). It can be described by the Wiener process which samples a random distance $\Delta d$ from a Gaussian distribution $\Delta \mathrm{d} \sim \mathcal{N}(0, \Delta t)$ that determines the position evolution as $d(t + \Delta t) = d(t) + \Delta d$. Ten Brownian motion sample functions can be inspected in Figure 2.3 where each sample function is a possible realization of Brownian motion. Both Gaussian processes and Brownian motion are types of stochastic processes and Brownian motion can in fact be re-expressed as a Gaussian process with zero mean $m(t) = 0$ and a covariance function $k(t,t') = \min(t,t')$ that depends on the minimum of two time points (Pavliotis, 2014).

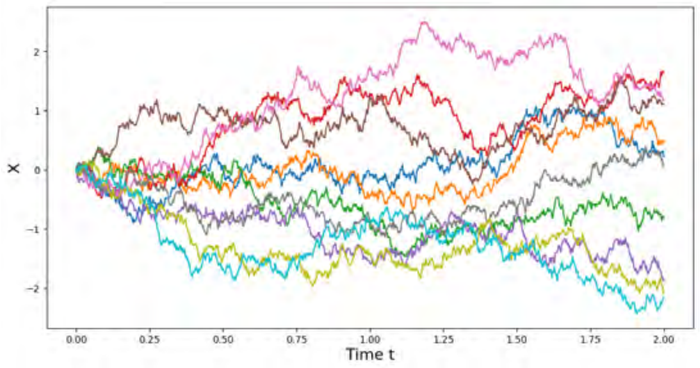


Fig. 2.3 Ten realisations of Brownian motion

## 2.5 Gaussian Process

Gaussian processes combine concepts from both probability distributions and stochastic processes to model functions. Specifically, a Gaussian process is defined as a stochastic process whereas every finite collection of its random variables has a multivariate Gaussian distribution. While the multivariate Gaussian distribution lives on a finite input domain and is specified by vectors or matrices, the Gaussian process lives on an infinite input domain with functions as arguments. In that way, the Gaussian process can be seen as an extension of a multivariate Gaussian distribution with an infinitely large index set. Thus, the two main properties of Gaussians - conditioning and marginalisation - still hold for Gaussian processes which makes their computation practical. In the following section, the process-centric view on Gaussian processes as a novel conceptual framework is introduced.

# Chapter 3

# Process-centric Gaussian Processes (GPs)

In this chapter, the process-centric view on Gaussian processes (GPs) is defined, providing a new framework to implement and think about GPs. This view is conceptually simpler and focuses on the main GP properties (the mean and covariance function) and operations (marginalization and conditioning). We discuss the role of the prior Gaussian process as a way to encode prior knowledge. Specifically, the effect of the prior mean and covariance functions on the shape of prior GP samples is illustrated. Furthermore, the two main GP operations, conditioning and marginalization, are explained and important practical implications are highlighted. This chapter concludes with an interpretation of the log marginal likelihood and its role for hyperparameter and model selection.

## 3.1 The Process-centric Framework

**Definition 1.** According to the **process-centric view**, a Gaussian process is an object, fully specified by a mean function $m$ and covariance function $k$ (and possibly its hyperparameters $\theta$) and allows two main operations: marginalization and conditioning.

Let $p(f) = \mathcal{N}(m,k)$ be a Gaussian process (GP) that encodes prior assumptions about a function $f$. This prior Gaussian process is fully specified by a mean function $m$ and covariance function $k$ and can be marginalized or conditioned. When conditioning a GP on observations $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, N\}$ via the likelihood function, we obtain a new GP that reflects the updated belief about the function $f$ along with an evaluation metric, the marginal likelihood. When marginalizing a GP, we evaluate the Gaussian process at a finite set of input locations $\mathbf{x}$, yielding Gaussian distributed finite function values $\mathbf{f} = f(\mathbf{x})$ which are defined by a mean vector $m(\mathbf{x})$ and covariance matrix $k(\mathbf{x}, \mathbf{x})$. It is important to note the notation difference between bold face letters for vectors and matrices (**m**, **K** respectively) and italicised font for functions (*m* and *k*).

### 3.1.1 Sample Generation

To gain a better intuition on the shape of Gaussian processes, it is useful to generate random functions. Random samples from a joint Gaussian distribution $\mathcal{N}(\mathbf{m},\mathbf{K})$ can be drawn by first computing the Cholesky decomposition of the covariance matrix, yielding the lower triangular matrix $L$:

$$\mathbf{L} = \text{cholesky}(\mathbf{K}) \implies \mathbf{L}\mathbf{L}^\top = \mathbf{K} \tag{3.1}$$

Second, the matrix multiplication of $\mathbf{L}$ with a vector $\mathbf{z}$ of Gaussian distributed independent entries is taken and the mean vector $\mathbf{m}$ is added to obtain a sample vector $\mathbf{y}$

$$z \sim \mathcal{N}(0,I), \quad \mathbf{y} = \mathbf{L}\mathbf{z} + \mathbf{m} \implies \mathbf{y} \sim \mathcal{N}(\mathbf{m},\mathbf{K}) \tag{3.2}$$

We can therefore draw samples from a Gaussian process after having marginalized a GP which will be useful in the following sections where we illustrate the role of prior GPs on the shape of random sample functions. The pseudo algorithm implementing Equation 3.1-Equation 3.2 can be inspected in Algorithm 1.

**Algorithm 1** Gaussian Sample

```
Require: $\mathbf{m}, \mathbf{K}$
1: $\mathbf{L} \leftarrow \text{cholesky}(\mathbf{K})$,
2: $\mathbf{z} \leftarrow \text{rand\_gaussian}(D, 1)$,
3: $\mathbf{y} \leftarrow \mathbf{L}\mathbf{z} + \mathbf{m}$
4: return $\mathbf{y}$
```

## 3.2 Prior Covariance Function

For Gaussian processes, the covariance function $k(x,x')$ (or kernel) is one of the two ways to encode prior knowledge about the function that we wish to model. It is a function of a pair of inputs $\mathbf{x}, \mathbf{x'}$ and describes the change of their corresponding function outputs $f(\mathbf{x})$, $f(\mathbf{x'})$ with varying spatial separation:

$$k(\mathbf{x},\mathbf{x'}) = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x'}) - m(\mathbf{x'})] \tag{3.3}$$

A valid covariance function for Gaussian processes has to be positive definite and symmetric: $k(x,x') = k(x',x)$ (Williams and Rasmussen, 2006). Positive definite means that if a covariance function is evaluated at a finite set of points, the resulting covariance matrix $\mathbf{K}$ is positive definite if and only if it has only positive eigenvalues $\lambda_i > 0$.

Each covariance function encodes different assumptions about the properties and shape of possible functions fitting the data. An overview of four different covariance functions is given in Table 3.1.

| Kernel Name | Formula | Structure Type |
|---|---|---|
| RBF | $k_{SE}(x,x') = \sigma_s^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$ | local variations |
| Periodic | $k_{Per}(x,x') = \sigma_s^2 \exp\left(-\frac{2\sin^2(\pi\lvert x-x'\rvert)/p}{\ell^2}\right)$ | repeating functions |
| Linear | $k_{Lin}(x,x') = \sigma_s^2 x \cdot x'$ | linear functions |
| White Noise | $k_{WN}(x,x') = \sigma_f^2 \delta(x-x')$ | uncorrelated noise |

Table 3.1 Overview of different Prior Covariance Functions commonly used with Gaussian processes

The Radial Basis Function (RBF)[1] and periodic kernel are examples of stationary kernels[2]. The RBF kernel is the most popular kernel function and assigns high similarity values to inputs that are close to each other. It has convenient properties such as infinite differentiability with mean square derivatives and is therefore suitable for modelling smooth and continuous functions (Williams and Rasmussen, 2006). For modelling exactly repeating functions, the periodic kernel, derived by David Mackay in MacKay et al. (1998), is an appropriate choice. An example for a non-stationary kernel is the linear kernel or 'dot product kernel'(Williams and Rasmussen, 2006, Section 4.2). It essentially performs Bayesian linear regression. Finally, the white noise (WN) kernel draws independent samples from a Gaussian random variable which means that it models no covariances but only the variance on the diagonals. In fact, a RBF kernel with a lengthscale approaching zero is equivalent to a white noise kernel.

The interested reader should refer to Duvenaud (2014) for a detailed overview of more kernel functions, such as the Matérn or rational quadratic kernel, as well as composite covariance functions. Since sums or products of two kernels are still valid kernels, new kernel combinations can be constructed by multiplication and addition (Williams and Rasmussen,

[1]The RBF kernel is also known as the Gaussian kernel or squared exponential function

[2]Stationary kernels depend only on the distance between two inputs whereas non-stationary kernels depend directly on the input coordinates.

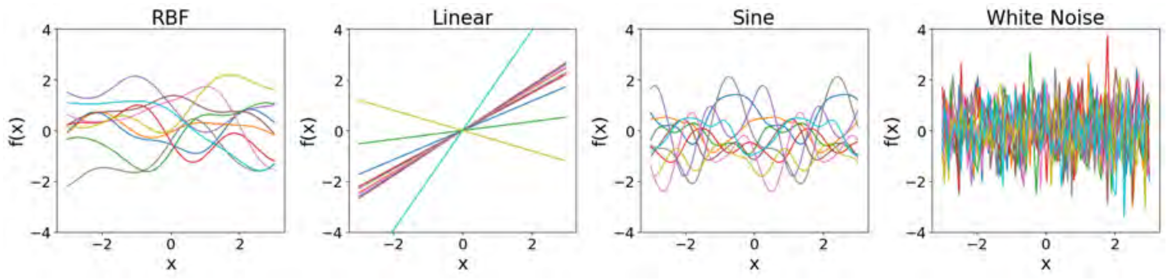


Fig. 3.1 Ten samples drawn from a zero-mean Gaussian process for four different kernels (RBF, Linear, Sine and White noise kernel).

2006, Subsection 4.2.4.). Figure 3.2 shows ten samples drawn from these four covariance functions with zero mean.

### 3.2.1 Covariance Hyperparameters:

Each covariance function is parametrised by hyperparameters $\theta$ that determine its shape. The two most common hyperparameters are:

- the characteristic lengthscale $\ell$ specifying the distance from which two points become uncorrelated (i.e. extrapolation on the x-axis) and
- the signal variance $\sigma_f^2$, modelling the average distance of the function away from its mean (i.e. amplitude on the y-axis).

When modelling noisy training data $\mathbf{y} = f(\mathbf{x}) + \varepsilon$ with i.i.d. Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma_\mathrm{n}^2)$, the noise variance hyperparameter $\sigma_n^2$ is added to the covariance as a diagonal matrix. The covariance of the noisy outputs can now be interpreted as adding a white noise kernel to the 'signal'. This gives us:

$$k_y(y, y') = k(x, x') + \sigma_\mathrm{n}^2 \delta(x - x')$$

where $\delta(x - x')$ is the Kronecker delta function. The hyperparameters can be either manually set or selected by maximising the marginal likelihood (section 3.6).

## 3.3 Prior Mean Function

Another way of incorporating knowledge into the Gaussian process is by defining a prior mean function $m(x)$. The mean function represents the average behaviour of the Gaussian process when drawing many samples and is given by:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

Prior kernel selection often assumes a zero mean $m(x) = 0$ which simplifies the formulation and works well enough in many scenarios. However, in the absence of observations or an uninformative covariance function, the posterior mean function becomes equivalent to the prior mean (subsection 3.5.2). Especially when modelling non-stationary data, the asymptotic behaviour of a function should be specified via a non-zero mean function. This is particularly important for extrapolation when modelling far-away data points. For example, Hwang et al. (2023) shows that zero-mean functions for supernovae reconstructions may result in unphysical estimates, therefore advocating for careful mean function selection for nonstationary signals.

| Mean Function | Formula | Structure Type |
|---|---|---|
| Zero | $m(x) = 0$ | no data trend |
| Linear | $m(x) = x$ | linear data trends |
| Periodic | $m(x) = sin(x)$ | periodic data trends |
| Step | $m(x) = \mathbb{1}(x > 0)$ | step changes |

Table 3.2 Overview of different Prior Mean Functions for Gaussian processes

A variety of mean functions, as shown in Table 3.2, can be used to model linear, periodic, or step data trends respectively. Furthermore, it should be noted that while defining a covariance function requires the fulfillment of constraints on the kernel definition, such as positive definiteness, mean functions can be defined more freely without special requirements. Ten samples drawn from these mean functions can be seen in Figure 3.2.

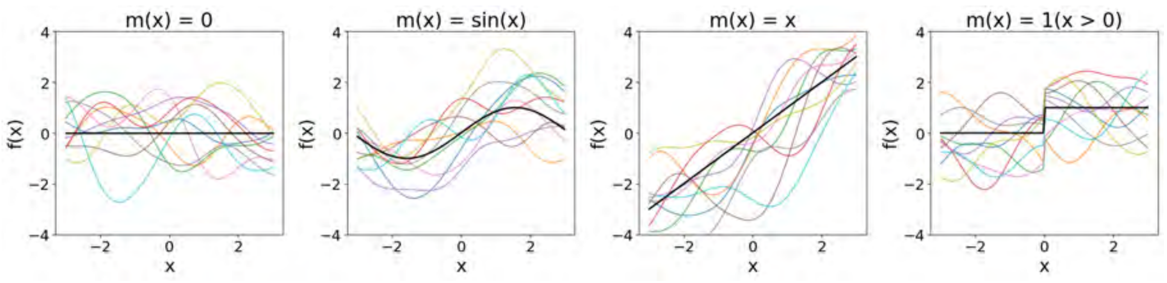


Fig. 3.2 Ten Samples drawn from four different mean functions (Zero, Sine, Linear, Step mean) with RBF-kernel

### 3.3.1 Mean Hyperparameters

Similarly to the covariance function, a mean function can be defined by hyperparameters $\theta$. Specifying hyperparameters for a linear mean function for example might be useful if we assume a certain slope or offset. However, we fixed the hyperparameters due to the unpopularity of defining non-stationary mean functions. Furthermore, optimizing the mean hyperparameters may result in overfitting for reasons mentioned when discussing the log marginal likelihood in subsection 3.6.2.

## 3.4 Conditioning Operation

We discuss now the conditioning operation which involves the updating of prior beliefs about a function based on observed data. We can use Equation 2.3, to express the joint distribution of the observations $\mathbf{y}$ and function $f$ as the product of the prior Gaussian process over functions $p(f) = \mathcal{N}(f \mid m, k)$ and the Gaussian likelihood $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{f}, \sigma_n^2 \mathbf{I}\right)$. This is equivalent to the product of the marginal likelihood $p(\mathbf{y}) = Z_{|\mathbf{y}}$ and the posterior Gaussian process over functions $p(f \mid \mathbf{y}) = \mathcal{N}(f \mid m_{|\mathbf{y}}, k_{|\mathbf{y}})$. The posterior GP has the same properties as the prior GP and is defined by a (posterior) mean $m_{|\mathbf{y}}$ and covariance function $k_{|\mathbf{y}}$. It can be seen as an updated prior after observing data. The likelihood $p(\mathbf{y} \mid \mathbf{f})$ as defined in Equation 2.2 encodes the probability of the observation vector given the function evaluations at the inputs. Gaussian process regression can be summarised with the following main mathematical formulation:

$$
\begin{aligned}
\mathcal{N}(f|\mathrm{m},\mathrm{k})\mathcal{N}(\mathbf{y}|\mathbf{f},\sigma_n^2\mathbf{I}) &= Z_{|\mathbf{y}}\mathcal{N}(f|\mathrm{m}_{|\mathbf{y}},\mathrm{k}_{|\mathbf{y}}), && (3.4)\\
\mathrm{m}_{|\mathbf{y}}(\mathrm{x}) &= \mathrm{m}(\mathrm{x}) + \mathrm{k}(\mathrm{x},\mathbf{x})[\mathbf{K}+\sigma_n^2\mathbf{I}]^{-1}(\mathbf{y}-\mathbf{m}), && (3.5)\\
\mathrm{k}_{|\mathbf{y}}(\mathrm{x},\mathrm{x'}) &= \mathrm{k}(\mathrm{x},\mathrm{x'}) - \mathrm{k}(\mathrm{x},\mathbf{x})[\mathbf{K}+\sigma_n^2\mathbf{I}]^{-1}\mathrm{k}(\mathbf{x},\mathrm{x'}), && (3.6)\\
Z_{|\mathbf{y}} &= \mathcal{N}(\mathbf{y} \mid \mathbf{m}, \mathbf{K}+\sigma_n^2\mathbf{I}). && (3.7)
\end{aligned}
$$

This formulation places all knowns (the prior GP and the likelihood) on the left hand side and all unknowns (the posterior GP and the log marginal likelihood) on the right hand side. While the covariance term depends only on the inputs, the mean also depends on the outputs. A detailed interpretation of each term is provided in subsection 3.5.2.

### 3.4.1 Algorithmic Implementation

**Algorithm 2** GP_conditioning

```
Require: $\mathrm{m}(\cdot \mid \theta), \mathrm{k}(\cdot,\cdot \mid \theta), \mathbf{X}, \mathbf{y}$
1: $\mathrm{K}[n,m] \leftarrow \mathbf{k}(\mathbf{x}_n, \mathbf{x}_m \mid \theta)$
2: $\mathbf{L} \leftarrow \mathrm{chol}(\mathbf{K})$
3: $\mathbf{A}(\cdot) \leftarrow \mathbf{L}^{-1}\mathrm{k}(\mathbf{X},\cdot \mid \theta)$
4: $\mathbf{b} \leftarrow \mathbf{L}^{-1}(\mathbf{y} - \mathbf{m}(\mathbf{X}))$
5: $\mathbf{m}_{|\mathbf{y}}(\cdot) \leftarrow \mathbf{m}(\cdot \mid \theta) + \mathbf{A}(\cdot)^\top \mathbf{b}$
6: $\mathbf{k}_{|\mathbf{y}}(\cdot,\cdot \mid \theta) \leftarrow \mathbf{k}(\cdot,\cdot \mid \theta) - \mathbf{A}(\cdot)^\top \mathbf{A}(\cdot)$
7: $\log Z_{|\mathbf{y}} \leftarrow -\frac{1}{2}\mathbf{b}^\top\mathbf{b} - \sum \log \mathrm{diag}(\mathbf{L}) - \frac{n}{2}\log 2\pi$
8: return $\mathrm{m}_{|\mathbf{y}}(\cdot \mid \theta), \mathrm{k}_{|\mathbf{y}}(\cdot,\cdot \mid \theta), \log Z_{|\mathbf{y}}$
```

The algorithmic implementation of the conditioning operation from Equations 3.4-3.7 can be inspected in Algorithm 2. It takes a prior mean $m$ and covariance function $k$ as input, both parametrised by their function-specific hyperparameters $\theta$, as well as the input matrix $\mathbf{X}$ and observation vector $\mathbf{y}$. Using the Cholesky decomposition for numerical stability and efficient matrix inversion, the posterior mean and covariance functions, as defined in Equations 3.5-3.6 are computed which are lambda functions in our Python implementation. The algorithm also returns the log marginal likelihood as in Equation 3.7.

### 3.4.2 Bayesian Online Updating

A Gaussian process (GP) can either be updated with the whole data set at once or in an online fashion using data batches. Let $\mathbf{y_1}, \mathbf{y_2}$ be vectorised subsets of the data. Here, we use the posterior GP from the previous iteration $p(f \mid \mathbf{y_1})$ as the new GP prior, and combine it with the Gaussian likelihood of the current data batch $p(\mathbf{y_2} \mid \mathbf{f})$. Using Bayes' rule from Equation 2.3, we obtain the posterior GP $p(\mathbf{f} \mid \mathbf{y_1}, \mathbf{y_2})$ and the normalizing constant $p(\mathbf{y_2} \mid \mathbf{y_1})$. This formulation is given as:

$$p(f, \mathbf{y_2} \mid \mathbf{y_1}) = p(f \mid \mathbf{y_1}) p(\mathbf{y_2} \mid \mathbf{f}) = p(\mathbf{y_2} \mid \mathbf{y_1}) p(f \mid \mathbf{y_1}, \mathbf{y_2}) \tag{3.8}$$

The posterior we obtain is equivalent to the formulation in Equation 3.4. However, the normalising constant is now different as it depends on $\mathbf{y_1}$:

$$\mathcal{N}(f \mid \mathrm{m}_{|\mathbf{y_1}}, \mathrm{k}_{|\mathbf{y_1}}) \mathcal{N}(\mathbf{y_2} \mid \mathbf{f}, \sigma^2) = p(\mathbf{y_2} \mid \mathbf{y_1}) \mathcal{N}(f \mid \mathrm{m}_{|\mathbf{y}}, \mathrm{K}_{|\mathbf{y}}) \tag{3.9}$$

To obtain the log marginal likelihood of all observations $\mathbf{y}$, we can simply add the log normalizing constants of both batches:

$$\log p(\mathbf{y}) = \log p(\mathbf{y_1}) + \log p(\mathbf{y_2} \mid \mathbf{y_1}) \tag{3.10}$$

An example for Bayesian online updating for (noise-free) synthetically generated sinusoidal data can be seen in Figure 3.3. In each iteration, the Gaussian process is conditioned on a new data point, using the posterior of the previous iteration as a new prior.

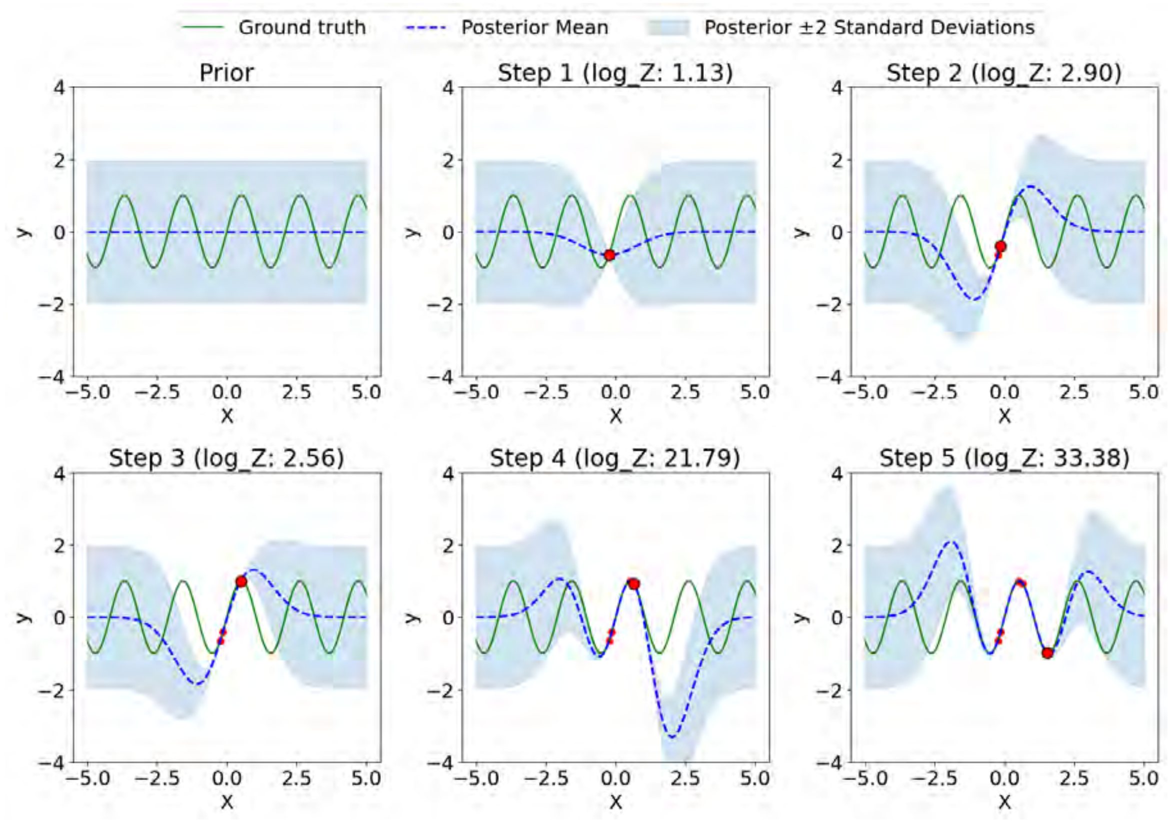


Fig. 3.3 Incremental conditioning of a Gaussian process with a zero mean and RBF covariance function ($\theta = \{\ell = 1, \sigma_n^2 = 0.001, \sigma_f^2 = 1\}$) as a prior. Five data points were sampled from a noise-free sine function, depicted in green. In each iteration, the Gaussian process is conditioned on a new observation (red circles). The new posterior mean and $\pm 2$ standard deviations are depicted as a dotted blue line and filled area. As new data are incorporated into the Gaussian process model, the negative log marginal likelihood (log_Z) is updated.

## 3.5 Marginalization Operation

A Gaussian process is an infinite collection of random variables, but when predicting outputs for certain test locations, we aim to obtain a finite set of random variables that can be worked with. Since a Gaussian process can be seen as a Gaussian distribution with an infinitely long mean vector and infinite by infinite covariance matrix, the marginalization property for multivariate Gaussian distributions, as defined in subsection 2.3.1 also holds for Gaussian processes. This means that the marginal distribution of a Gaussian process is obtained by simply ignoring all random variables we are not interested in.

Marginalization evaluates the Gaussian process $f$ at a finite set of input locations $\mathbf{x}$ which yields a Gaussian distributed finite vector of function values $\mathbf{f} = f(\mathbf{x})$. Thus, marginalization allows to move from an infinite-dimensional object, a stochastic process specified by a mean and covariance function, to a finite-dimensional object, a distribution, specified by a finite mean vector $m(\mathbf{x})$ and covariance matrix $k(\mathbf{x},\mathbf{x})$.

$$f \sim \mathcal{N}(m,k) \implies f(\mathbf{x}) \sim \mathcal{N}(\mathbf{m} = m(\mathbf{x}), \mathbf{K} = k(\mathbf{x},\mathbf{x})) \tag{3.11}$$

### 3.5.1 Algorithmic Implementation

Marginalization from Equation 3.11 is implemented in Algorithm 3. It takes as inputs a mean and covariance function and a set of input points $\mathbf{X}$ to evaluate and it returns a mean vector and covariance matrix which are obtained after evaluating both mean and covariance function at the inputs. The function outputs can then be used for visualisation purposes or for making predictions.

**Algorithm 3** GP_marginal

```
Require: $\mathrm{m}(\cdot \mid \theta), \mathrm{k}(\cdot,\cdot \mid \theta), \mathbf{X}$
1: $\mathbf{m}[n] \leftarrow \mathrm{m}(\mathbf{x}_n)$
2: $\mathbf{K}[n,m] \leftarrow \mathrm{k}(\mathbf{x}_n, \mathbf{x}_m)$
3: return $\mathbf{m}, \mathbf{K}$
```

### 3.5.2 Decomposition of the posterior mean and covariance matrices

Figure 3.4 show two different posterior Gaussian process models, conditioned on 10 training points, sampled from noise-free sinusoidal data. The marginalised mean and $\pm 2$ standard deviation are shown on the left (plot a and d) and were obtained by evaluating the posterior mean and covariance function at the test locations using Equation 3.5 and Equation 3.6 respectively and Equation 3.11. Both models were defined with a sinusoidal prior mean function, but differ in their prior covariance specification. The first model (top) is defined by a RBF prior covariance function, whereas the second model (bottom) is defined by a white noise kernel. On the right side, the decomposition of both covariance matrix and mean vector can be inspected on a red-blue colorscale. Red colors indicate low values whereas blue values indicate high values. Note that the colorscales between covariance matrix and mean vector differ as the mean vector can take negative values, but the covariance matrix has to be positive semi-definite. In the following, these plots will be explained in more detail.

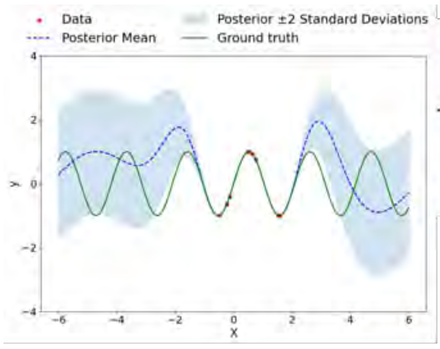
We can obtain the posterior covariance matrix by evaluating the posterior covariance function from Equation 3.6 on test inputs. For a single test point $\mathbf{x}_*$, the posterior covariance can be decomposed into the difference between the following two terms:

- The term $k(\mathbf{x}_*, \mathbf{x}_*)$ represents the variance of the output at the test point $\mathbf{x}_*$ under the prior covariance function.
- The term $k(\mathbf{x}_*, \mathbf{x})[\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} k(\mathbf{x}, \mathbf{x}_*)$ quantifies the contribution of the training data to the variance at the test point $\mathbf{x}_*$.

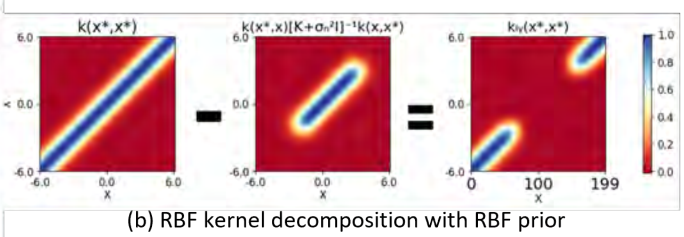
By subtracting the data-dependent term from the prior variance, we obtain the posterior variance $k_{|\mathbf{y}}(\mathbf{x}_*, \mathbf{x}_*)$ reflecting to what extent the uncertainty decreases after observing data. A greater similarity between both terms corresponds to more confident predictions. For example, at test locations near the training points, the posterior variance under the RBF kernel is low since the prior and data highly agree with each other. However, for test locations far away from the training data, the uncertainty increases and reverts to the prior. In the case of the white noise kernel, the posterior equals the prior since the white noise kernel does not model correlations in the data, and thus no additional knowledge is gained from observing data (see plot e)

Using the equation for the posterior mean function from Equation 3.5, we can similarly obtain the posterior mean $\mathbf{m}_{|\mathbf{y}}(\mathbf{x}_*)$ by decomposing it into two terms:

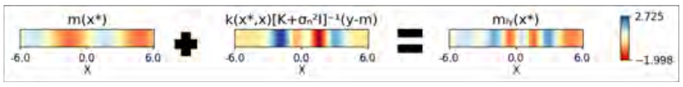
- The prior mean $m(\mathbf{x}_*)$ reflects the expected function value at the test point $\mathbf{x}_*$ under the prior mean function.
- The data contribution term $k(\mathbf{x}_*, \mathbf{x})[\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1}(\mathbf{y} - \mathbf{m})$ describes to what extent the data explains the average function value.



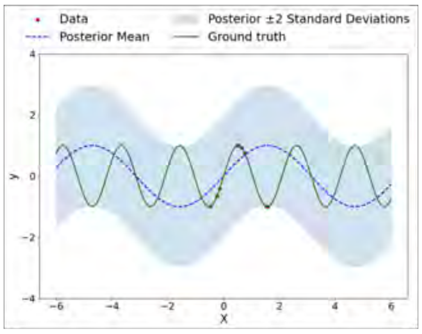
(a) Posterior Gaussian process with RBF kernel and sine mean



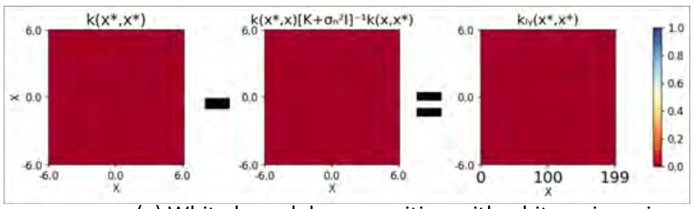
(b) RBF kernel decomposition with RBF prior



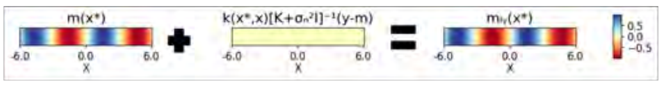
(c) Mean vector decomposition with sine prior



(d) Posterior Gaussian process with white kernel and sine mean



(e) White kernel decomposition with white noise prior



(f) Mean vector decomposition with sine prior

Fig. 3.4 Two Gaussian process Posterior Models for noise-free sinusoidal data after conditioning on 10 test points. Figure (a) and (d) depict the marginalised mean and $\pm$ standard deviation of the posterior Gaussian process using 200 data points. Both models were specified with a prior sinusoidal covariance function but with different kernels (top: prior RBF kernel with $\ell = 1$, $\sigma_f$=1, $\sigma_n = 0.001$ and bottom: white noise kernel with $\sigma_f$=1, $\sigma_n = 0.001$). The covariance matrix and mean vector decompositions can be seen on the right which were obtained by plotting the components of Equation 3.6 and Equation 3.5 respectively, evaluated at the test data. Red colors indicate low values whereas blue colors indicate high values. Note that the colorscales between covariance matrix and mean vector differ as the mean vector can take negative values, but the covariance matrix has to be positive semi-definite. This figure was inspired by an illustration from Schulz et al. (2018)

The sum of both terms yields the posterior mean. While a zero mean function only incorporates data-dependent knowledge about expected function values, a non-stationary mean function includes certain prior assumptions about the expected function values. The effect of the prior mean becomes especially evident for extrapolation. For example, in the case of a sinusoidal prior mean function and in the absence of data, the posterior mean equals the prior mean, exhibiting periodic trends rather than reducing to zero (see Figure 3.4c). Finally, in case of an uninformative covariance function (aka the white noise kernel), the prior and posterior mean vectors are equivalent since the data contribution term from Equation 3.5 is zero. These considerations demonstrate that defining a prior mean can have practical advantages in some cases.

## 3.6 Marginal Likelihood

As shown in Equation 3.4, the conditioning operation returns the logarithm of the marginal likelihood $p(\mathbf{y})$ which reflects how well the posterior Gaussian process explains the data under the prior GP. It marginalizes over the latent function values at a set of locations which justifies the name 'marginal likelihood'.

$$p(\mathbf{y}) = \int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}) d\mathbf{f} \tag{3.12}$$

The marginal likelihood can also be seen as the probability of generating the observations $\mathbf{y}$ when randomly drawing function values $\mathbf{f}$ from the prior. While simple priors can only explain a small number of datasets with high probability, overcomplex priors spread their probability mass on too many datasets. This trade-off between minimizing model complexity and minimizing model misfit is known as Occams razor (Rasmussen and Ghahramani, 2000), and is explained further in the following subsection.

### 3.6.1 Occam's Razor

Occams razor is best described by considering the analytical solution for the log marginal likelihood. We now write the log marginal likelihood as $\log p(\mathbf{y} \mid \mathbf{X}, \theta)$ and explicitly highlight that it is conditioned on the input matrix $\mathbf{X}$ and the hyperparameters $\theta$. The analytical solution for the log marginal likelihood is then given as:

$$\log p(\mathbf{y}|\mathbf{X},\theta) = -\frac{1}{2}(\mathbf{y}-\mathbf{m})^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}(\mathbf{y}-\mathbf{m}) - \frac{1}{2}\log\left|\mathbf{K}+\sigma_n^2\mathbf{I}\right| - \frac{n}{2}\log 2\pi \quad (3.13)$$

Besides a constant term, this expression contains two main terms:

- **The Data Fit Term** ($-\frac{1}{2}(\mathbf{y}-\mathbf{m})^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}(\mathbf{y}-\mathbf{m})$) measures how well the model fits the observed data $\mathbf{y}$. A larger value indicates a better data fit.
- **The Model Complexity Term** ($-\frac{1}{2}\log\left|\mathbf{K}+\sigma_n^2\mathbf{I}\right|$) penalizes more complex models. A smaller value (mind the negative sign) indicates a more complex model.

Maximising the sum of both terms reflect this trade-off between data fit and complexity, preferring the least complex model able to explain the data.

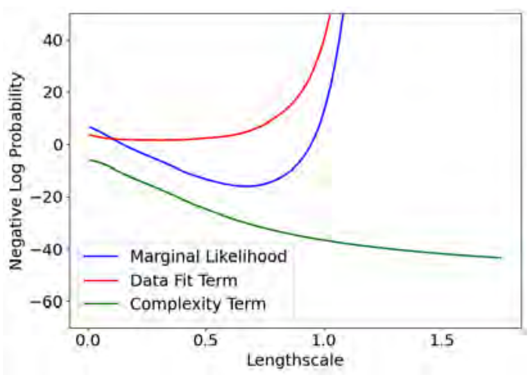
**An example visualisation of Occams Razor**

Figure 3.5a visualises this trade-off for different lengthscales by showing the complexity, data fit term, and marginal likelihood in green, red, and blue respectively for different lengthscales. As we consider a minimisation problem of the negative log marginal likelihood, smaller values are better. With increasing lengthscale, the data (un-)fit term increases while the complexity penalty decreases, corresponding to an over-simplified model. A very small lengthscale improves the fit but at the cost of a higher complexity penalty.

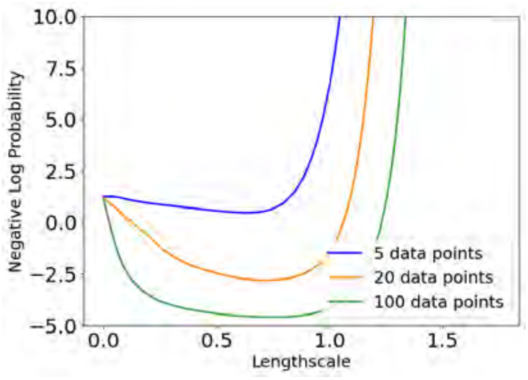
This effect is further illustrated by choosing three lengthscales ($\ell = 0.1$, $\ell = 0.8$, and $\ell = 1.5$) and inspecting their corresponding posterior GP mean and $\pm 2$ standard deviations in Figure 3.6a. We can also see their prior covariance functions evaluated at the test points (yielding the matrix $\mathbf{K}$) in Figure 3.6b. A medium-sized lengthscale ($\ell = 0.8$) minimises the negative log marginal likelihood, optimally balancing fit and complexity. In contrast, a too small ($\ell = 0.1$) or too large ($\ell = 1.5$) lengthscale results in over-, or underfitting. We can also see in Figure 3.6b that overfitting stems from a prior covariance matrix $\mathbf{K}$ that is almost diagonal, whereas an underfitted GP considers too many neighboring points.

Finally, let us discuss the effect of the data set size that can be inspected in Figure 3.5b. The plot visualises the negative log marginal likelihood per data point for three different datasets. As we can see, when increasing the data set size, the log marginal likelihood becomes more peaked and discourages overcomplex models with too short lengthscales stronger.

The important observations made in this section emphasize the importance of hyperparameter selection which is discussed in the following.



(a)



(b)

Fig. 3.5 Figure a) shows the decomposition of the log marginal likelihood (blue) into the model fit (red) and complexity component (green) for different lengthscales and N=8 training inputs. Corresponding visualisations of the posterior GP for different lengthscales can be seen in Figure 3.6a. Figure b) depicts the log marginal likelihood per data point for a small (blue), medium (orange) and large (light-green) dataset.

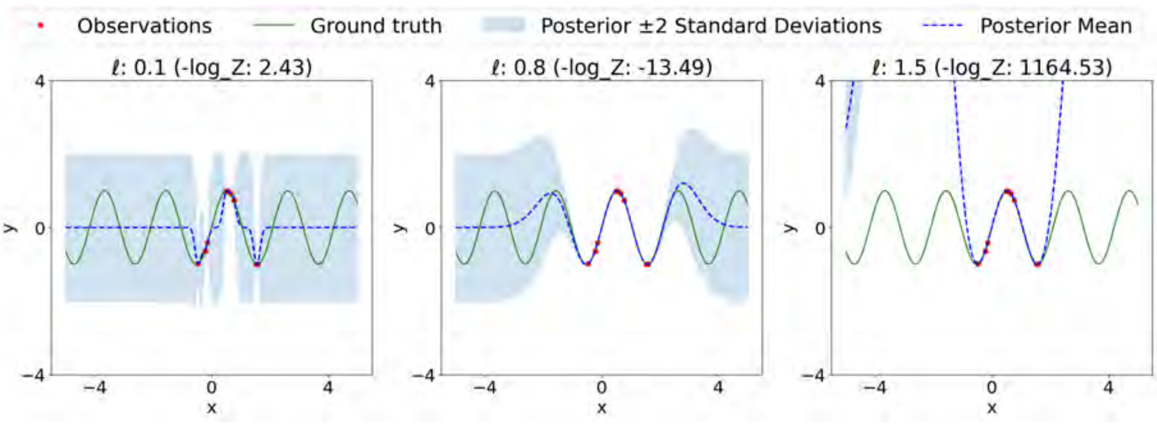
.

### 3.6.2 Hyperparameter Optimisation

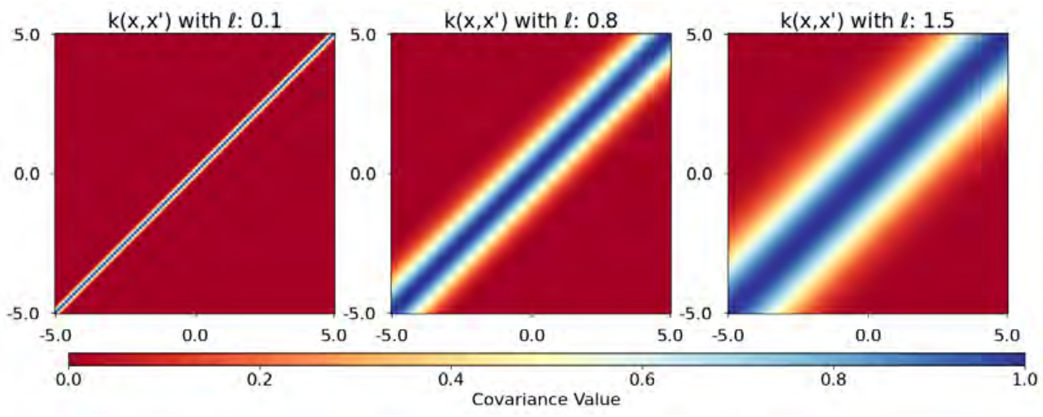
So far, the hyperparameters $\theta$ and model (defined by a mean and covariance function) were fixed. The formula for this level of inference is given in the main GPR formulation in Equation 3.4. However, as discussed in subsection 3.6.1, the kernel hyperparameters strongly affect the complexity and fit of the posterior Gaussian process and should therefore be carefully selected. In practise, the best fitting hyperparameters are learned by maximising the marginal likelihood with respect to the hyperparameters Equation 3.14. This is an approximation, also known as Type II Maximum Likelihood and usually done using gradient-based methods.

$$\begin{aligned} \frac{\partial \log p(\mathbf{y} \mid \mathbf{X}, \theta)}{\partial \theta_j} &= \frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_j} \mathbf{K}^{-1}\mathbf{y} - \frac{1}{2}\operatorname{tr}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \\ &= \frac{1}{2}\operatorname{tr}\left((\alpha\alpha^\top - \mathbf{K}^{-1})\frac{\partial \mathbf{K}}{\partial \theta_j}\right) \text{ where } \alpha = \mathbf{K}^{-1}\mathbf{y}. \end{aligned} \tag{3.14}$$

The contour plots of the log marginal likelihood for different hyperparameter combinations of the RBF kernel can be inspected in Figure 3.7. They were obtained using noisy sinusoidal data (noise=0.5). As visible in plot b) there can be multiple local optima which correspond to different explanations of the data. Keeping the noise fixed, the same data can either be explained by a more complex model (with lower lengthscale), or a a less complex



(a) Posterior Gaussian process regression with mean and ±2 standard deviations.

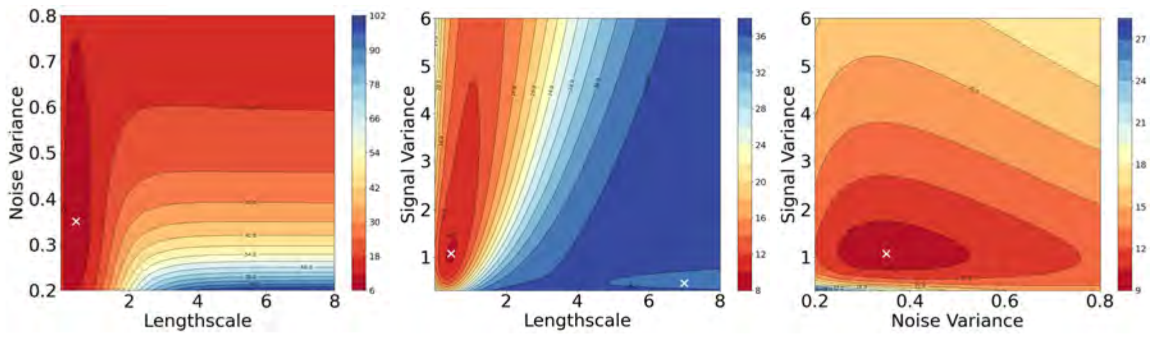


(b) Prior covariance function evaluated on the test points.

Fig. 3.6 Gaussian process regression and corresponding covariance matrices for the length-scales: $\ell = 0.1$, $\ell = 0.8$, and $\ell = 1.5$. The data was drawn from a noise-free sinusoidal function.

model (with higher lengthscale). The global optimum is the more complex model that notably improves the data fit which can be seen by a higher signal variance. Moreover, for the simpler model with a lower signal variance, the model becomes almost invariant of the lengthscale since the model explains the data mainly by noise.

It should be noted that we only considered the optimisation of the covariance hyperparameters. Optimising the hyperparameters of the mean function would improve the data fit term without adding complexity penalties as the complexity term from the log marginal likelihood in Equation 3.13 is independent of the mean. The resulting risk for overfitting might contribute to the unpopularity of hyperparameter optimisation of the mean function.



(a) (b) (c)

Fig. 3.7 Contour plots of the negative log marginal likelihood for different hyperparameter combinations after optimizing the third hyperparameter. Local optima are marked with a white cross. The optimized hyperparameters are $\theta = \{\sigma_f^2 = 1.07, \ell = 0.45, \sigma_n = 0.347\}$. (a) Lengthscale vs. noise variance ($\sigma_f^2 = 1.07$), (b) Lengthscale vs. signal variance (two local optima, $\sigma_n = 0.347$), (c) Noise vs. signal variance ($\ell = 0.45$).

### 3.6.3 Model Selection

Another application of the marginal likelihood is model selection. Let us define model $M$ to be a choice of a covariance function. The goal of model selection is to identify the most suitable covariance function for the given data. Let $p(\mathbf{y} \mid \mathbf{X}, M)$ be the Type II maximum likelihood of the selected model. Using Bayes' rule, this gives us a way to define the posterior probability of a model $p(M \mid \mathbf{y}, \mathbf{X})$ that quantifies how likely the model is given the observed data:

$$p(M \mid \mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y} \mid \mathbf{X}, M)p(M)}{p(\mathbf{y} \mid \mathbf{X})} \tag{3.15}$$

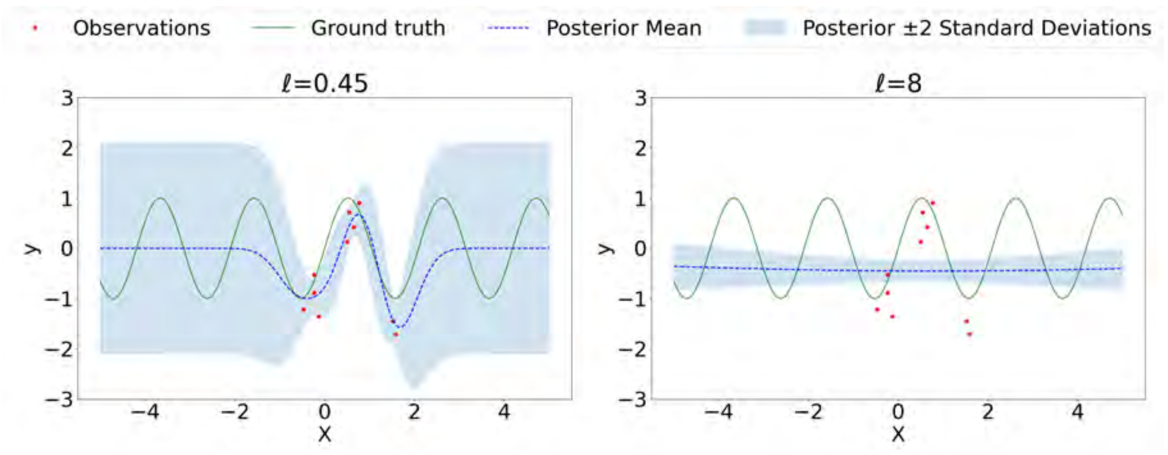
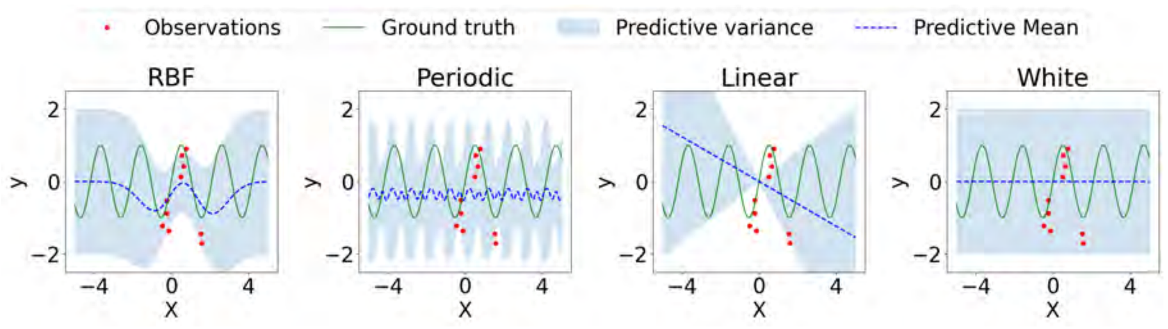


Fig. 3.8 Two explanations of the sinusoidal (noisy) data with a fixed noise level. The left plot corresponds to a more complex model with ($\theta = \{\sigma^2 = 1.07, l = 0.45, \sigma_n = 0.347\}$) (global optimum) and Plot b) corresponds to a simpler model with: ($\theta = \{\sigma^2 = 0.36, l = 8, \sigma_n = 0.347\}$) (local optimum)

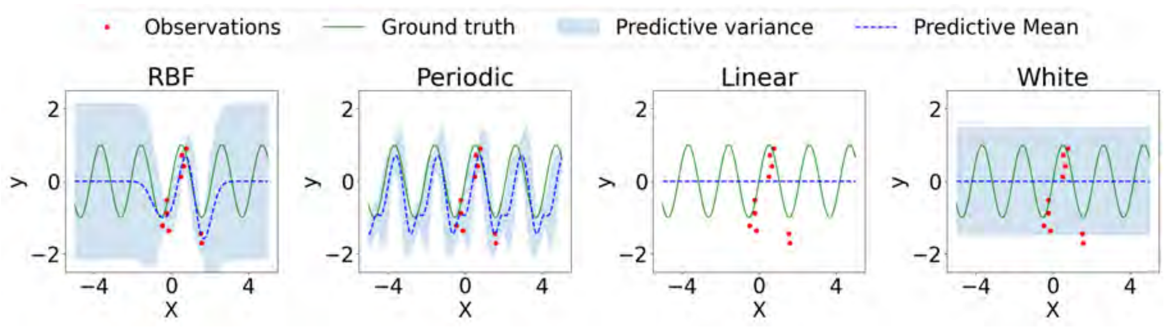
Here, $p(M)$ represents now the prior probability of the model (or 'hyperprior'). Assuming a discrete set of possible models with equal prior probabilities (i.e. a uniform prior), this simplifies the model selection process to choosing the model that maximizes the Type II maximum likelihood.

To illustrate this, we revisit the noisy sinusoidal data example from subsection 3.6.2. The outcomes of hyperparameter optimization are shown in Figure 3.9b, and the corresponding Type II maximum likelihoods for different covariance functions are compared in Figure 3.9d. It is important to note that the preferences for different models can change before and after hyperparameter optimization. Initially, with fixed hyperparameters for all kernels (e.g., $\theta = \{\ell = 1, \sigma_n^2 = 1, \sigma_f^2, p = 1\}$), the RBF kernel minimizes the negative log marginal likelihood, as shown in Figure 3.9a. However, after optimizing the hyperparameters, the marginal likelihood suggests a preference for the periodic kernel over the RBF kernel.

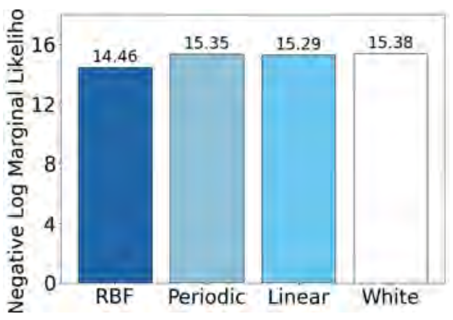
Although the RBF kernel fits the training points well, its variance increases far away from the data, reflecting higher uncertainty for extrapolation. In contrast, the periodic kernel generalizes better and produces lower uncertainty estimates even for data points far from the training set, which aligns with the sinusoidal nature of the underlying function. On the other hand, both the white noise and linear kernels perform poorly, yielding large variances and inaccurate predictions, highlighting their inadequacy in modeling smooth, non-linear data due to their inability to capture non-linear patterns.



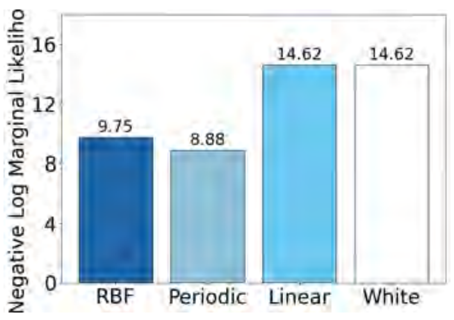
(a) GPR before hyperparameter optimisation



(b) GPR after hyperparameter optimisation.



(c) Log marginal likelihood before optimisation



(d) Log marginal likelihood after optimisation

Fig. 3.9 Comparison of Gaussian process regression (GPR) for different covariance kernels before (Figure a) and after (Figure b) hyperparameter optimisation. The corresponding log marginal likelihood values can be inspected in Figures c and d.

Table 3.3 Optimised Hyperparameters of each model

| Kernel | varSigma | lengthscale | noise | period |
|---|---|---|---|---|
| RBF | 1.07 | 0.46 | 0.35 | - |
| Periodic | 0.95 | 0.91 | 0.32 | 2.17 |
| Linear | 0.00 | - | 1.04 | - |
| White | 0.74 | - | 0.74 | - |

## 3.7 Summary

This chapter provides an in-depth demonstration of the process-centric view on Gaussian processes. It highlights the role of the prior GP and hyperparameters to encode assumptions about a function $f$ and demonstrates the conditioning and marginalization operation through a choice of visualisations and examples. We also show the role of the log marginal likelihood in providing a natural trade-off between model fit and complexity, making it suitable for model selection.

# Chapter 4

# Related Work

In this chapter, a higher-level overview of two common views on Gaussian processes (GPs), the weight-space and function-space views, is provided. This is followed by a discussion on how the process-centric view relates to previous views. The chapter concludes with a literature-based motivation for the development of a novel 'process-centric' interactive Gaussian process software. This software was created during this thesis and is designed to simplify learning through intuitive visualizations, facilitating accessibility of 'process-centric' Gaussian processes to both initial learners and practitioners.

## 4.1 The weight-space view

The weight-space view interprets Gaussian processes as an extension of Bayesian linear regression. In this view, a function $f$ is modelled as a weighted sum of a D-dimensional input vector $\mathbf{x}$ with the weights $\mathbf{w}$ which is given by $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$. Then, a Gaussian prior with zero mean and covariance $\Sigma_p$ is placed on the weights $\mathbf{w} \sim \mathcal{N}(0, \Sigma_p)$. To increase the model's flexibility, the input vector is transformed into a higher-dimensional space using the basis function $\phi(\mathbf{x})$. When using a (possibly) infinite number of basis functions, the Bayesian linear regression model corresponds to a Gaussian process. For a detailed mathematical derivation, the reader should refer to Section 2.1. in Williams and Rasmussen (2006)

This view is advantageous for those familiar with parametric models and Bayesian statistics, as it offers a clear transition from parametric to non-parametric modeling. However, it indirectly tackles Gaussian process regression and involves a lengthy mathematical derivation. Therefore, it might be more challenging to understand for those less familiar with concepts such as parametrics, basis functions, or the kernel trick. The weight-space view is less related to the process-centric view and thus, this thesis does not discuss it in detail.

## 4.2 The function-space view

The function-space view tackles Gaussian processes directly by modeling probability distributions over unknown functions rather than weights (Williams and Rasmussen, 2006, Section 2.2). It specifies a Gaussian process prior over all possible functions $f(\mathbf{x}) \sim \mathcal{GP}(m,k)$ in the same way as described under the process-centric view in Equation 3.4. It then infers the posterior predictive distribution $p(f_* \mid \mathbf{x}_*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\bar{f}_*, \mathbb{V}(f_*))$ for a new test point $\mathbf{x}_*$ which is given by the predicted function value $\bar{f}_* = \mathbf{k}_*^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y}$ and the predictive variance $\mathbb{V}(f_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*$. We can see from this notation that inference involves three evaluations of the prior covariance function: $\mathbf{k}_*$ is the vector of covariances between the test point and $N$ training points, $\mathbf{K}$ contains the covariance matrix between all training points, and $k(\mathbf{x}_*, \mathbf{x}_*)$ is the variance of the test point. Note that for multiple test locations $\mathbf{X}*$, this vector becomes a covariance matrix between all test inputs.

This view aligns closely with the process-centric view that focuses on the interpretation of a Gaussian process as a stochastic process that can also be seen as a single random variable in function space (aka 'random function') (Lamperti, 2012, Chapter 1). Nevertheless, both views focus on different types of objects and thus differ in terms of their interpretation which is discussed in detail in the following section.

## 4.3 Relation of previous views to the process-centric view

The relation between the three different views can be demonstrated by re-expressing the covariance function $k(\mathbf{x},\mathbf{x}')$, evaluated at $\mathbf{x}$ and $\mathbf{x}'$, in terms of the basis functions $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$ respectively. This expression is a dot product with respect to the covariance function $\Sigma_p$:

$$k(\mathbf{x},\mathbf{x}') = \phi(\mathbf{x})^\top \Sigma_p \phi(\mathbf{x}').$$

We write shorthand $\phi(\mathbf{x}^*) = \phi*$ for a test input $\mathbf{x}^*$ and $\Phi = \Phi(\mathbf{X})$ to be an aggregation of all columns $\phi(\mathbf{x})$ of the training data. To make predictions, we can then obtain the following expressions for the covariance matrices: $\mathbf{K} = \Phi^\top \Sigma_p \Phi, \quad \mathbf{k}_* = \phi_*^\top \Sigma_p \Phi, \quad$, and $k(\mathbf{x}_*, \mathbf{x}_*) = \phi_*^\top \Sigma_p \phi_*$. Note that these correspond to the components of the posterior predictive distribution in section 4.2.

Since the posterior predictive distribution is equivalent to a marginalised posterior Gaussian process from the process-centric view, this effectively shows the connection between all three views. In the following, the focus will be on the relation between the function-space and process-centric views, as more direct approaches toward Gaussian processes.

While the function-space view focuses on the posterior distribution of the test outputs, the process-centric view is defined in terms of a joint distribution of the training and test outputs (Equation 2.3). The latter definition has an intuitive interpretation that when using Bayes' rule, this allows placing all known variables on one side (the likelihood and prior) and all unknowns on the other side (marginal likelihood and posterior). In practical terms, both views start by defining a Gaussian process prior but then differ in terms of their computations. While the function-view conditions on both training and test inputs to obtain the posterior predictive distribution which is a multivariate Gaussian distribution, the process-centric view conditions on the training inputs to obtain a posterior Gaussian process. While the latter is an infinite-dimensional object defined in terms of functions, the posterior predictive distribution is a finite-dimensional object defined in terms of a vector and matrix.

Directly obtaining a finite-dimensional object can be more practical by providing predictions for test points without requiring further computations. Nevertheless it might be conceptually less intuitive that prior and posterior are of different types (stochastic process vs. probability distribution). In that way, the process-centric view provides a more unified treatment of the prior and posterior since both are defined in terms of Gaussian processes. The resulting model is more flexible for training as the posterior Gaussian process can now be used as a new prior for online updating of sequentially arriving data (see section 3.4) and for testing as the posterior Gaussian process can be evaluated at any test point section 3.5.

In summary, the process-centric view obtains the same result from the (function-space) posterior predictive but splits the process into conditioning on training points and marginalization at the test points. In that way, the process-centric view provides a new straightforward interpretation of Gaussian processes by focusing on its properties (the mean and covariance function) and operations (marginalization and conditioning).

## 4.4 Educational Gaussian Process Material

Gaussian process models are widely used in a variety of domains, ranging from applications in robotics (Deisenroth, Fox, and Rasmussen, 2013) and astronomy (Foreman-Mackey et al., 2017) to Bayesian optimization (Snoek et al., 2012). They have also been applied in the development of AlphaGo, a program that successfully defeated the Go world champion (Chen et al., 2018). Despite their versatility and the extensive literature on Gaussian processes, they often remain difficult to understand due to their mathematical complexity and abstract nature. This section provides an overview of educational Gaussian process literature, highlighting key resources.

### 4.4.1 Books and Theses

A number of academic textbooks (MacKay et al., 1998; Murphy, 2022; Williams and Rasmussen, 2006) and PhD theses (Duvenaud, 2014; Rasmussen, 1997) written by subject experts offer in-depth theoretical underpinnings into Gaussian processes. These resources include extensive mathematical derivations which require a strong background in calculus, linear algebra, and statistics, While they are invaluable for experienced practitioners, they focus on mathematical detail rather than intuitive understanding which imposes substantial barriers for many readers.

### 4.4.2 Tutorial Articles

Tutorial articles by Schulz et al. (2018) and Wang (2023) offer concise explanations of Gaussian processes, targeting a broader audience, including Gaussian process novices. These articles are more accessible as they provide a high-level overview of key mathematical concepts. Moreover, they emphasize intuitive understanding through visual illustrations and practical examples. For instance, Schulz et al. (2018) explains the posterior predictive distribution by visually decomposing the mean vector and covariance matrix into their components. Likewise, Wang (2023) uses 3D visualisations to illustrate how conditional distributions are obtained by slicing through a bivariate Gaussian distribution.

### 4.4.3 Online Resources

Online tutorials focus on a less academic audience and provide probably the most hands-on explanations as they often include code snippets, (possibly interactive) visualizations, and practical examples. This section covers various formats of online resources available for Gaussian processes. These include: interactive tutorials, static tutorials, and audiovisual materials.

#### Interactive Tutorials

Interactive tutorials include comprehensive step-by-step explanations of Gaussian processes, combining text, mathematical explanations, and interactive graphs with modifiable parameters. For instance using the software from Deisenroth, van der Wilk, and Luo (2020, December) and Görtler et al. (2019), users can explore the behaviour of the covariance matrix by adjusting hyperparameter values via a slider, or they can condition a Gaussian process by clicking on data points. John (2021) provides a single interactive figure that illustrates the main properties and operations of Gaussian processes. This software allows users to more freely explore Gaussian processes by customizing plotting options and specifying data and covariance properties at the same time. However, the extensive availability of interactive elements without accompanying textual explanations may be confusing and overwhelming for Gaussian process novices.

#### Static Tutorials

As static alternatives, Roelants (2019) and Wang (2023) provide Gaussian process tutorials based on Jupyter Notebooks and cover similar content as its interactive counterparts while additionally showing code snippets. This format is particularly targeted towards software developers or data analysts that are mainly interested in the coding implementation of Gaussian processes. Another advantage of Jupyter Notebook-based tutorials is the possibility of direct code execution using web services such as Google Colab or Binder.

#### Audiovisual Materials

An alternative to merely written content are audiovisual resources on social media or online learning platforms, such as YouTube or Moodle. These offer a multi-sensory learning style that might facilitate learning (Fuady and Mutalib, 2018). For example, Richard Turner ((Turner, 2016)) provides an excellent video lecture on Gaussian processes. In addition to verbal explanations, he includes vivid visual animations in his presentation that illustrate how multivariate Gaussians extend to Gaussian processes when the index set becomes infinitely large. A written summary of this lecture is provided by Geten (2019).

## 4.5 Motivation for Interactive Jupyter Notebook

Visual explanations have been consistently shown to enhance conceptual learning in STEM subjects (Bobek and Tversky, 2016), highlighting the value of a visual-based approach in educational tools. Just (2010) demonstrates that interactive visualizations can significantly improve both learning outcomes and student motivation compared to static images. However, interactivity is not guaranteed to enhance learning and its effectiveness depends on several factors, including the student's prior knowledge (Park et al., 2009) and the complexity of the topic being taught (Patwardhan and Murthy, 2015). To optimize learning, Patwardhan and Murthy (2015) suggest incorporating 'interactivity enriching features' to guide the learner's exploration. An example for this would be to restrict the numerical range of sliders.

Given these findings, the integration of interactive visual elements into educational resources should be done thoughtfully, ensuring that such features genuinely enhance the learning experience. Jupyter Notebooks are particularly well-suited for this purpose. As interactive programming tools, they allow for the seamless combination of code, text, and visuals. This format has been widely adopted in education across fields like computer science (Al-Gahmi et al., 2022), artificial intelligence (Nelson and Hoover, 2020), and big data (Yuen and Robbins, 2014), where active engagement with the content is crucial. The ability to execute code in real-time, visualize results, and adjust parameters engages students more, leading to higher learning outcomes (Amoudi and Tbaishat, 2023). This suggests that an interactive tool built within a Jupyter Notebook could effectively support the learning of complex concepts such as the process-centric view of Gaussian processes.

## 4.6 Research Contribution

This thesis contributes to the Gaussian process literature by developing a new process-centric view as introduced in Hensman and Rasmussen (2024). This view is conceptually easier to understand than existing views. An interactive Jupyter Notebook that has been developed during this thesis complements a theoretical explanation with carefully designed interactive visualizations, allowing the users to explore the process-centric view dynamically. By integrating interactive elements, the notebook not only facilitates a deeper understanding of the process-centric approach but also aims to enhance user engagement and learning outcomes, directly addressing the need for a more intuitive explanation of Gaussian processes.

# Chapter 5

# Interactive Jupyter Notebook

This chapter introduces the process-centric view on Gaussian processes through an interactive software based on Jupyter Notebook. The motivation for an interactive notebook stems from the assumption that interactive learning improves both user motivation and learning outcomes (Just, 2010). Therefore, in addition to providing a detailed conceptual framework of the process-centric view in chapter 3, as introduced by Hensman and Rasmussen (2024), an interactive tutorial software was designed with a focus on interactive visualisations that enable the user to dynamically change plot outputs and learn parameter-output relationships which might improve the conceptual understanding of Gaussian processes. This chapter covers a description of the target audience, design considerations, implementation details, as well as a detailed overview of the Jupyter Notebook content. This chapter concludes by discussing deployment and the results of a small user study conducted as part of this project. The notebook can be accessed here: Interactive Gaussian Process Tutorial (or via this URL: https://github.com/annkristinbalve/process-centric-gp_tutorial/blob/main/Tutorial_Colab_Final.ipynb).

## 5.1 Target Audience and Goals

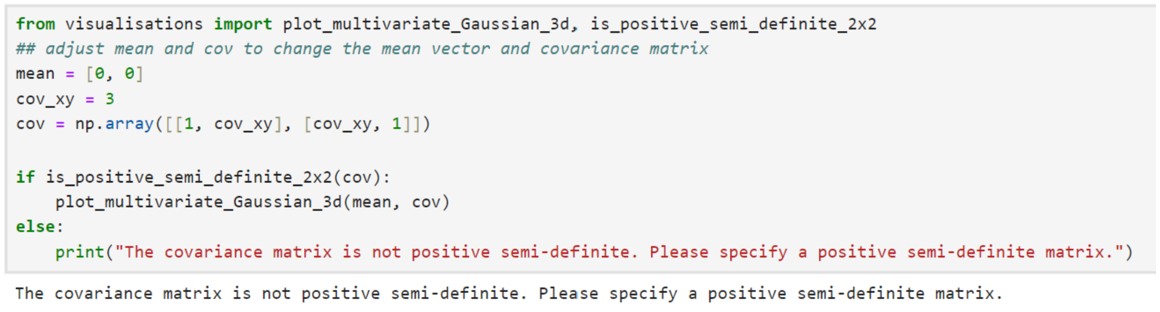
The Jupyter Notebook is designed as educational material for process-centric Gaussian processes. Gaussian processes are flexible models for quantifying uncertainty and are widely used. Nevertheless, they are often poorly understood. The interactive notebook is designed to enhance the learning experience and improve the conceptual understanding of Gaussian processes by introducing a new 'process-centric' view.

The novel software is designed for a wide target audience of both Gaussian process 'newcomers' and more advanced learners. This includes, but is not limited to: final and penultimate undergraduate and postgraduate computer science and engineering students at higher education institutions, as well as Machine Learning researchers and other users interested in learning about Gaussian processes.

## 5.2 Design Considerations

This section covers the key design considerations to improve both the user experience and usability of the 'process-centric GP Jupyter Notebook'. The design of the notebook was guided by principles defined from Nielsen (1994) and Kosslyn (2006). Each of the interactive visualisation, numbered V1-V7, will be introduced in section 5.4. The most important design considerations for the Jupyter Notebook are summarised in the following:

- **Appropriate Background Knowledge**: Even though we assume a university-level mathematics background of the user, basic probability concepts are covered in an initial background section to ensure that the starting point for learning about Gaussian processes is given.
- **Relevant Graphics**: The notebook only introduces one interactive figure per concept aiming to provide a compact summary of each concept.
- **Consistency in the Design**: Different colored boxes for important definitions, formulas, and plot descriptions are given which helps maintaining a structured interface.
- **Pleasurable User Experience**: The notebook includes severeal features that aim to provide a 'fun' and more pleasurable experience, such as renderable 3D plots.
- **Status visibility**: All plots were optimised to reduce running time. Nevertheless, some plots (V5-V6), that involve GP conditioning, may take longer to update and therefore contain status information ('Updating Plot' vs. 'Plot Updated').
- **Constraints**: To maintain a valid interpretation and minimise errors, the correlation slider from the bivariate distributions (V2-V3) were restricted to allow only positive semi-definite matrices. Furthermore, kernel hyperparameters are disabled if not needed (i.e. greying out the period hyperparameter for the white noise kernel).
- **Error Handling**: In case the user nevertheless specified an invalid covariance matrix, a warning message is displayed (see Figure 5.1).



```
from visualisations import plot_multivariate_Gaussian_3d, is_positive_semi_definite_2x2
## adjust mean and cov to change the mean vector and covariance matrix
mean = [0, 0]
cov_xy = 3
cov = np.array([[1, cov_xy], [cov_xy, 1]])

if is_positive_semi_definite_2x2(cov):
    plot_multivariate_Gaussian_3d(mean, cov)
else:
    print("The covariance matrix is not positive semi-definite. Please specify a positive semi-definite matrix.")
```

```
The covariance matrix is not positive semi-definite. Please specify a positive semi-definite matrix.
```

Fig. 5.1 Example for error handling when the user enters a correlation value resulting in a non-positive semi-definite covariance matrix. A short error message is displayed stating the problem and resolution which might be more solution-oriented than the lengthy scipy default error messages.

It should be noted that there there are trade-offs between user experience and usability (Sharp and Rogers, n.d.) in the sense that some features perceived as more ’pleasurable’ might be less clear for concept learning. A 2D plot might be preferred over a 3D plot when summarising information such as local or global optima, although being aesthetically less pleasing. A best trade-off between both objectives was aimed to achieve, however it is not guaranteed to work for everyone.

## 5.3 Implementation Details

This section provides an overview of the technical implementation details of the Jupyter Notebook, including package versions, notebook structure, and deployment.

### 5.3.1 Packages

The software is entirely Python-based, utilizing Python 3.12.4. The starting point for the interactive figures was the `matplotlib` visualizations, as shown in Chapter3. To transition from static to interactive visualizations, these were reimplemented using `plotly`, which generates highly interactive plots. `plotly` allows users to inspect individual data points, hide plot traces via mouse clicks on the legend, or render 3D plots. Another advantage of `plotly` is its seamless integration with Jupyter widgets (`ipywidgets`), which are designed to create dynamic controls for Jupyter Notebooks, such as dropdown menus, tabs, and sliders.

To create a more immersive learning experience for users, both tools were combined, yielding highly interactive plots with modifiable plot displays through `plotly` and adjustable input parameters through `ipywidgets`. Additional packages used include `numpy` for matrix manipulations, `scipy` to draw samples from a multivariate Gaussian, and `autograd` for hyperparameter optimization.

Specific package versions can be accessed in the `requirements.txt` file on GitHub (here).

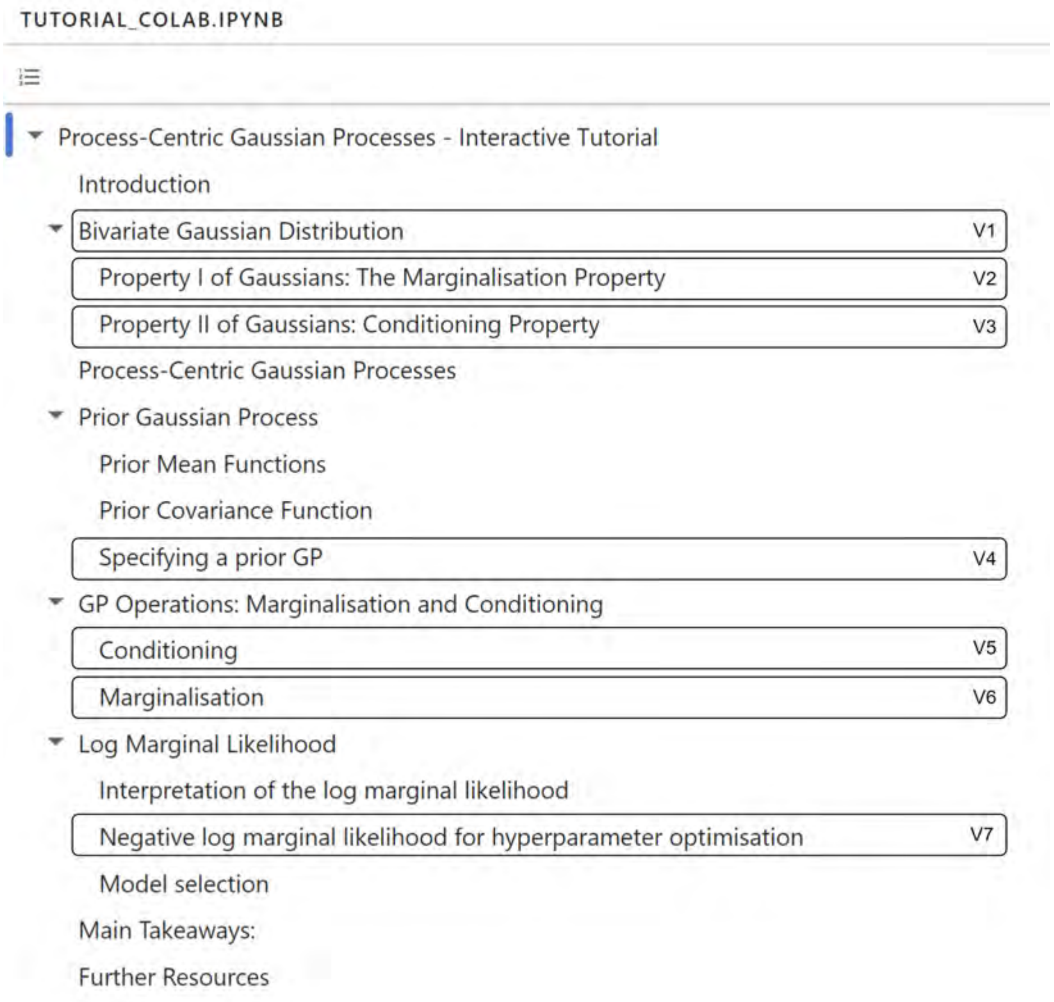
### 5.3.2 Code Structure

The notebook is organized with a modular architecture to ensure clarity and maintain separation of content. The main components are as follows:

- `tutorial.ipynb`: The primary Jupyter Notebook file containing the core instructional content.
- `means.py`: Contains the mean functions for Gaussian processes (`zero_mean`, `lin_mean`, `sine_mean`, `step_mean`).
- `kernels.py`: Contains the covariance functions for Gaussian processes (`rbf_kernel`, `lin_kernel`, `periodic_kernel`, `white_kernel`).
- `gp_functions.py`: Implements key Gaussian process operations, including `GP_marginal`, `GP_conditional`, `GP_conditional_optimised`, and `draw_samples`.
- `data.py`: Contains functions to generate synthetic sinusoidal data and testing points.
- `widgets_helper.py`: Provides helper functions for managing interactive widgets.
- `visualisations.py`: Includes plotting functions for both static and interactive visualizations.

## 5.4 Jupyter Notebook Content

In this section, an in-depth illustration of each interactive visualisation is given in the order of their appearance in the notebook. The Notebook contains both text, main formulas, and code to generate the visualisations. An overview of the content can be found in Figure 5.2. Sections including interactive visualisations (V1-V7) are highlighted with a box. We also discuss the purpose of each visualisation for Gaussian process learning.



TUTORIAL_COLAB.IPYNB

- Process-Centric Gaussian Processes - Interactive Tutorial
  - Introduction
  - Bivariate Gaussian Distribution V1
    - Property I of Gaussians: The Marginalisation Property V2
    - Property II of Gaussians: Conditioning Property V3
  - Process-Centric Gaussian Processes
  - Prior Gaussian Process
    - Prior Mean Functions
    - Prior Covariance Function
    - Specifying a prior GP V4
  - GP Operations: Marginalisation and Conditioning
    - Conditioning V5
    - Marginalisation V6
  - Log Marginal Likelihood
    - Interpretation of the log marginal likelihood
    - Negative log marginal likelihood for hyperparameter optimisation V7
    - Model selection
  - Main Takeaways:
  - Further Resources

Fig. 5.2 Overall Notebook Structure: Sections that include interactive visualisations (V1-V7) are highlighted with boxes.

### 5.4.1 Bivariate Gaussian Distribution (V1)

This interactive 3D plot (Figure 5.3) is a vivid depiction of a bivariate Gaussian distribution for two random variables $x_1$ and $x_2$ . Their marginal distributions are shown in green $p(x_1)$ and black $p(x_2)$ respectively. 100 samples drawn from the joint are visible as brown scatter points. The plot can be rendered with mouse interaction or touch.

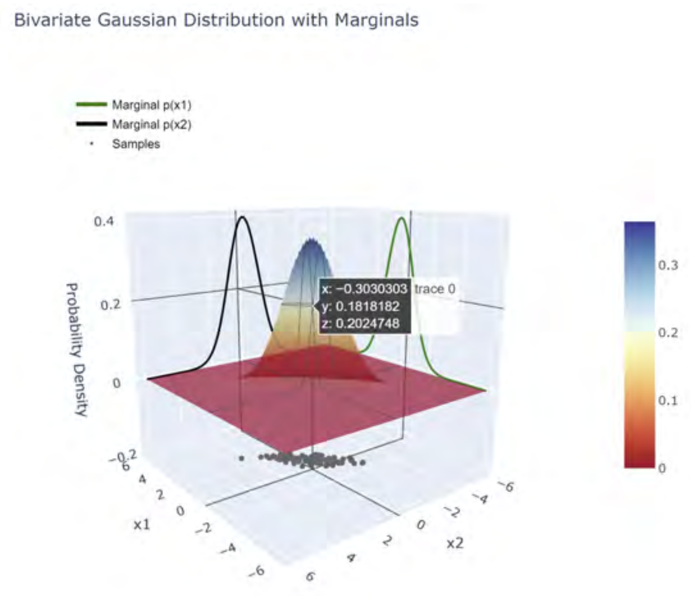


Fig. 5.3 Interactive 3D Bivariate Gaussian Plot (V1): The plot shows the probability density for a bivariate Gaussian. The marginal distributions for $x_1$ and $x_2$ are shown in green and black respectively. Samples drawn from this multivariate distribution are shown as brown dots. The user can render the plot through mouse interaction or touch. Furthermore, individual probability density values of the surface plot can be inspected by hovering over the plot.

Table 5.1 Features and Learning Goals of Figure V1

| **Feature** | **Learning Goal** |
|---|---|
| Renderable 3D plot of a bivariate Gaussian distribution | Understand the shape of a multivariate Gaussian distribution |

### 5.4.2 Property I of Gaussians: Marginalization Property (V2)

This interactive 2D contour plot demonstrates the marginalization property for two random variables. The plot displays the marginal distributions for each variable alongside a red-to-blue heatmap representing probability density. Users can adjust the means $m_1$ and $m_2$, as well as the covariance parameter $c$ to observe their effects on the joint and marginal distribution.

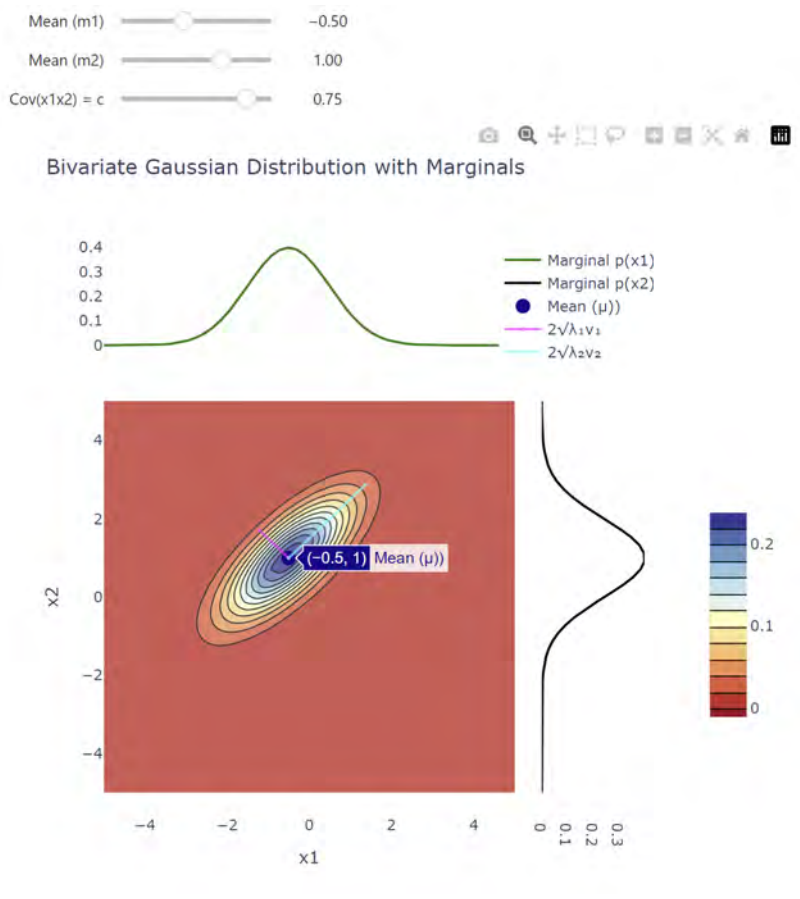


Fig. 5.4 Interactive 2D Bivariate Gaussian Plot (V2): This figure shows the joint probability density on a red-to-blue colorscale and the corresponding marginal distributions along each axis . The user can adjust the means $m_1$, $m_2$ and covariance parameter $c$ between two random variables $x_1$ and $x_2$. Furthermore, the mean (dark-blue) and scaled eigenvectors $2\sqrt{\lambda}\mathbf{u}$ (cyan and magenta) show the center and shape of the distribution.

Table 5.2 Features and Learning Goals of Figure V2

| **Feature** | **Learning Goal** |
|---|---|
| Adjustable mean and covariance sliders | Understand relations between joint and marginal distributions. |

## 5.4.3 Property II of Gaussians: Conditioning Property (V3)

The interactive 2D contour plot in Figure 5.5 demonstrates the conditioning property of a bivariate Gaussian distribution. This plot allows users to explore how conditional distributions change for different covariance matrices and observed values. The user can dynamically adjust these parameters to observe their impact on the conditional distribution.

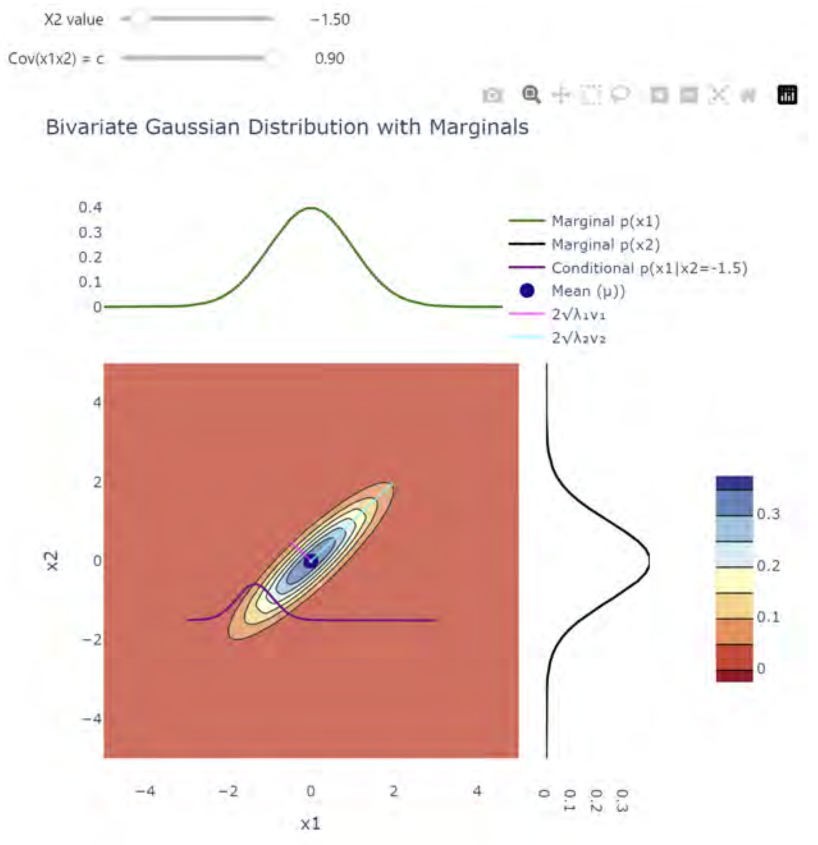


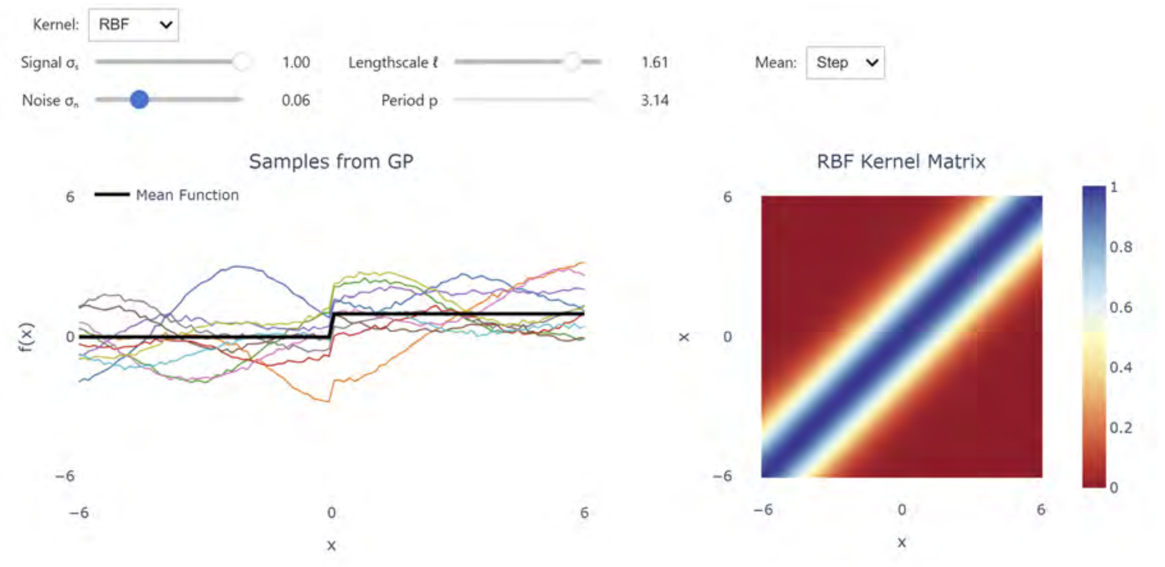
Fig. 5.5 Interactive 2D Bivariate Gaussian Plot (V3): This figure illustrates the concept of conditioning by showing how the conditional distribution $p(x_1 \mid x_2$ changes with different correlation values $c$ and observations $x_2$. The user can adjust these parameters interactively.

Table 5.3 Features and Learning Goals of Figure V3

| Feature | Learning Goal |
|---|---|
| Inputs $x_2$ and correlation $c$ slider | Understand how observed values and covariance affect the conditional distribution. |

### 5.4.4 Specifying a prior GP (V4)

Figure 5.6 demonstrates how different prior assumptions, specified through mean functions, covariance functions, and covariance hyperparameters, influence the resulting Gaussian process. Users can interactively explore how these parameters shape sample functions drawn from a prior GP. Additionally, the effect of covariance hyperparameters on the GP's prior covariance matrix is shown.



Kernel: RBF
Signal $\sigma_s$ 1.00
Lengthscale $\ell$ 1.61
Mean: Step
Noise $\sigma_n$ 0.06
Period p 3.14

Fig. 5.6 Interactive Plot V4: This figure illustrates how prior mean functions, covariance functions, and hyperparameters influence samples drawn from a Gaussian process. Users can explore how different prior assumptions shape the sample functions (left) and observe how covariance-specific hyperparameters affect the GP's prior covariance matrix (right).

Table 5.4 Features and Learning Goals of Figure V4

| Feature | Learning Goal |
|---|---|
| Kernel and mean selection menus | Understand how different GP priors influence sample functions. |
| Covariance hyperparameter sliders | Explore the impact of hyperparameters on the prior covariance matrix. |

### 5.4.5 Conditioning a GP (V5)

Figure 5.7 illustrates the conditioning operation for a Gaussian process. The figure shows the posterior mean and $\pm 2$ standard deviations at different steps of conditioning. Initially, the prior (step-0) is visualized. Users can condition the GP prior on new data points by moving the slider from left to right, with each step corresponding to a batch update.

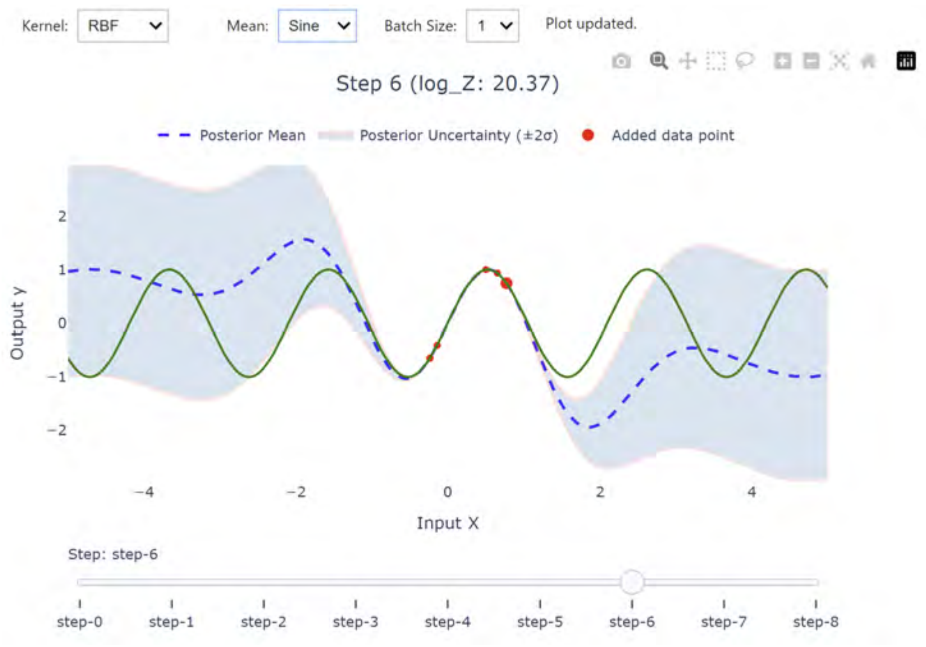


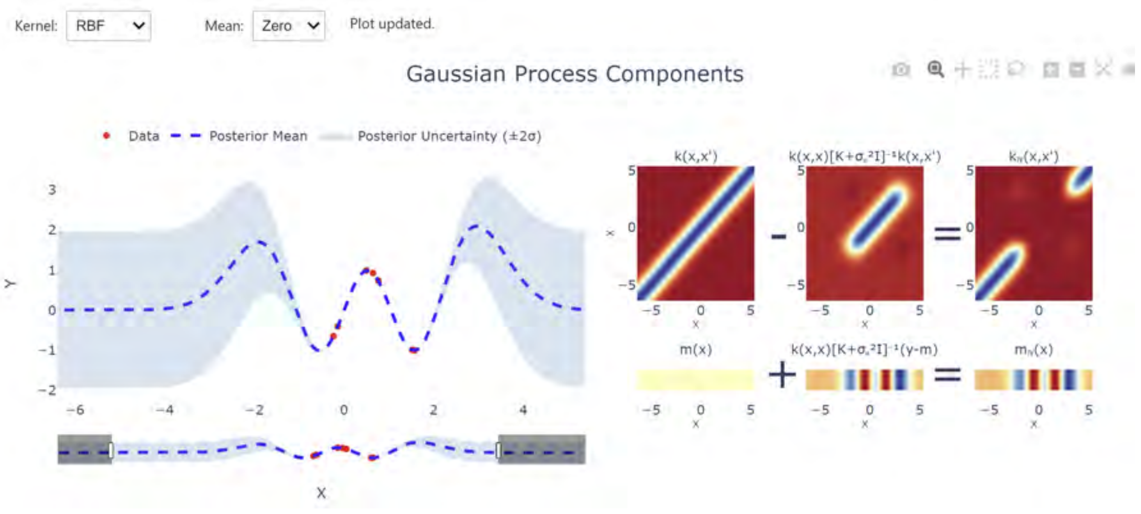
Fig. 5.7 Interactive Plot V5: This plot visualizes the effect of conditioning a prior Gaussian process (GP) on observations. It shows the evolution of the posterior mean and uncertainty ($\pm 2\sigma$) as new data points are incorporated step by step. Users can adjust the batch size to explore scenarios involving streaming data. The plot also displays the negative log marginal likelihood (log_Z) at each step, reflecting the model fit as conditioning progresses.

Table 5.5 Features and Learning Goals for Figure V5

| **Feature** | **Learning Goal** |
|---|---|
| Step-slider | Understand the role of conditioning in reducing uncertainty. |
| Covariance and mean drop-down menus | Explore how prior assumptions affect the posterior mean and covariance. |
| Batch-size | Understand the implications of batch size on online updates in GPs. |

## 5.4.6 Marginalization of a GP (V6)

Figure 5.8 illustrates the role of marginalizing a posterior Gaussian process (GP) in making predictions. This interactive figure allows users to observe the GP's fit at selected test points, along with a detailed visualization of the covariance matrix and mean vector components. The figure helps foster an intuitive understanding of how posterior mean and covariance are calculated and how the posterior distribution changes at different test locations.


Kernel: RBF
Mean: Zero
Plot updated.

Gaussian Process Components
Data
Posterior Mean
Posterior Uncertainty (±2σ)
k(x,x')
k(x,x)[K+σ$_n^2$I]$^{-1}$k(x,x')
k$_N$(x,x')
m(x)
k(x,x)[K+σ$_n^2$I]$^{-1}$(y−m)
m$_N$(x)
y
x

Fig. 5.8 Interactive Plot V6: This plot illustrates the concept of marginalization by illustrating how a posterior Gaussian process can be used to obtain predictions at certain test locations. The plots on the right also provide an interpretation of the prediction process by visualizing the components of the covariance matrix and mean vector.

Table 5.6 Features and Learning Goals for Figure V6

| **Feature** | **Learning Goal** |
|---|---|
| Input-value slider | Effect of different test locations on marginalisation for both extrapolation and interpolation |
| Covariance and mean drop-down | Gain an intuitive understanding of how the posterior mean and covariance are calculated and how the prior GP affects predictions |

### 5.4.7 Hyperparameter Optimisation (V7)

The interactive figure V7 features three 3D plots of the (negative) log marginal likelihood surface as a function of different RBF kernel hyperparameters. Each plot can be rendered, zoomed in, and rotated. Furthermore, in the first tab (see Figure 5.9) , two local optima can be inspected when fixing the signal variance to $\sigma_n = 0.35$.

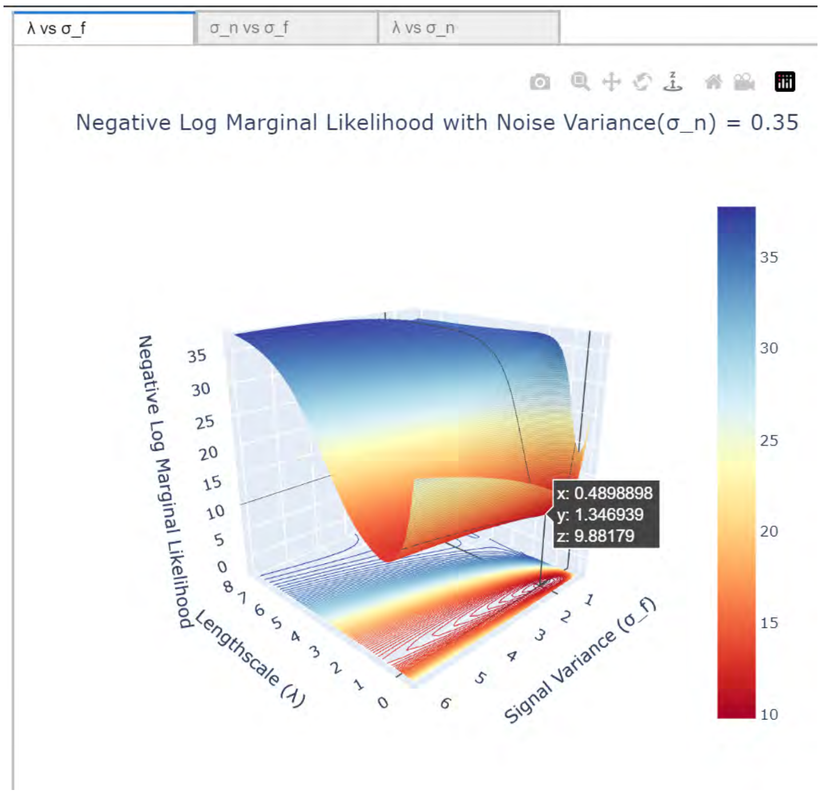


Fig. 5.9 The negative log marginal likelihood surface can be inspected on a 3D surface that can be rendered and rotated via touch or mouse interaction. Furthermore, two local optima can be seen when setting the signal variance to $\sigma_n = 0.35$.

Table 5.7 Features and Learning Goals for Figure V7

| **Feature** | **Learning Goal** |
|---|---|
| Render 3D plots of log marginal likelihood | Understand role of log marginal likelihood for hyperparameter selection |

## 5.5 Deployment

For the Notebook deployment, Google Colab was chosen as it provides free computing resources, alleviates dependency problems, and can be run directly on the browser. This makes the notebook more accessible as the user does not have to install new packages or locally store files. The notebook can be directly loaded using GitHub. Initially, alternatives to Colab were considered, specifically Render, a cloud application that can similarly be connected to Github, but unlike Colab it automatically installs packages. Due to a very slow loading time for our Jupyter Notebook (>5min) which mainly involved package conflicts with Plotly, deployment through Render was not further pursued. Furthermore, alternatives to Notebooks were tested such as Plotly Dash which creates Python-based Dashboards. However, deployment outside of a local server usually involves commercial platforms (e.g. Heroku, Dash Enterprise). Moreover, the Dash-based notebook lacked efficiency and most plot updates took much more than 200ms. These practical considerations as well as the proven advantages of using Jupyter Notebooks for educational purposes, as discussed in Chapter 4, subsection 4.4.3, led to the choice of using Jupyter Notebooks.

## 5.6 User Testing and Evaluation

A very small qualitative pilot study with 9 participants was conducted to gather feedback on the interactive tutorial. The results can be inspected in Appendix A. Even though not being representative due to a small sample size and convenience sampling, they give some indications on the usability of an interactive Jupyter Notebook tutorial as well as the perception of the process-centric GP view. Interestingly, 7 out of 9 users reported that they found the process-centric view definition more intuitive than the weight-space view or function-space view definition of Gaussian processes. 5 participants furthermore self-evaluated an improved understanding of the marginalisation operation after completing the tutorial. The interactive elements and visualisations were in general perceived as helpful and participants reported that they enjoyed completing the tutorial. Nevertheless, it should be noted that the notebook might be more suitable for experienced learners as some ’GP novices’ reported that they felt like being ’thrown into the deep’. Furthermore, some participants would have preferred a different format over a Jupyter Notebook, such as a web-page or video which should be considered in future research.

## 5.7 Summary

Overall, this chapter introduces a novel software for Gaussian processes which might be a useful educational tool for learning about Gaussian processes. Considering its length and depth, it should be used as a supplementary material to lectures instead of a stand-alone tool.

# Chapter 6

# Discussion and Conclusion

The thesis introduced the process-centric view as a new framework for understanding and implementing Gaussian Processes (GPs). This view is introduced in Hensman and Rasmussen (2024) and describes Gaussian processes as stochastic processes specified by a mean and covariance function with two main operations: conditioning and marginalisation. This chapter summarises the main contributions of this thesis for the Gaussian process literature and discusses limitations and potential directions for future research.

## 6.1 Main Contributions

The main contribution of this thesis is the exposition of the process-centric view, both with respect to theory and practise.

**Development of a novel 'process-centric' GP framework**: This thesis develops a conceptually simpler 'process-centric' view of Gaussian processes. Furthermore, the small user survey shows some indications that the process-centric definition might be helpful in understanding GPs. By building on intuition rather than mathematical derivation, this new framework provides a more accessible entry point for newcomers to the field of Gaussian processes, particularly those with less mathematical backgrounds. The process-centric view might therefore also encourage a deeper engagement with and application of Gaussian processes.

**Demonstrating the inherent flexibility of GPs**: Moreover, we demonstrate how the process-centric framework introduces a highly flexible framework. Clearly distinguishing between conditioning and marginalisation implies a natural separation between training and testing which has several practical advantages. Notably, the conditioning operation can handle any number of data points and allows a posterior GP to become a prior GP for a following iteration. This feature highlights the suitability of the process-centric framework for Bayesian online learning with streaming data. Learning at real-time is for example crucial to guarantee safety of physical systems such as autonomous vehicles (Lederer et al., 2021; McAllister et al., 2017). Moreover, online learning reduces memory usage since a GP can discard incoming data after conditioning as it retains all information.

**Highlighting the role of the underappreciated mean function**: We highlight the role of the mean function as a way to encode prior knowledge. The mean function has been historically underappreciated and the majority of the scientific literature assumes zero-mean, focusing instead on the covariance function. However, specifying a non-zero prior mean function has important implications for solving real-world problems with non-stationary data trends, such as in cosmology (Hwang et al., 2023). We demonstrate the key role of the mean function for specifying asymptotic behaviour of functions, particularly for extrapolation problems using simple examples.

**Encouraging Gaussian Process Learning through interactive software**: The development of interactive visualizations represents another key contribution of this dissertation. These visualizations are integrated into an interactive Jupyter Notebook, which is accessible on Github. The notebook serves as a practical demonstration of the theoretical discussion of the process-centric view. This makes such rather complex concepts more tangible and easier to grasp. The small user study indicates that especially the Notebook's visualisations may improve learning of Gaussian processes. This software is also well suited for integration into a lecture environment by serving as a supplementary educational tool.

## 6.2 Limitations

While the current demonstration of process-centric Gaussian processes is promising, it has certain shortcomings. These include the accessibility constraints of the interactive software, a limited variety of data, and a lack of extensive empirical evidence for the effectiveness of the process-centric view as a superior framework for Gaussian processes.

**Accessibility of Notebook**: The Jupyter Notebook requires code execution, which might pose challenges for users without programming proficiency or for smartphone users. Although Google Colab provides a convenient cloud-based environment, the need for a Google account may limit accessibility of the tutorial. Another important limitation is running time which is especially reduced in the cloud environment. These factors can potentially impair the usability of the notebook and should be addressed in future work.

**Data Choice**: The thesis primarily utilizes a dataset of 10 synthetically generated sinusoidal data points. While this is effective for illustrative purposes, it limits the exploration of Gaussian processes to small and periodic data. In real-world scenarios with larger datasets, exact Gaussian process regression faces scalability issues, as it requires inverting an $N \times N$ covariance matrix. Thus, future work should demonstrate the application of the process-centric view to more varied real-world examples.

**Limited user studies**: Moreover, while we postulate that the new process-centric framework for GPs is easier to understand, this claim is based on mainly theoretical considerations rather than empirical validation. The lack of comprehensive user testing means that the effectiveness of the process-centric view in enhancing conceptual understanding of Gaussian processes remains to be conclusively demonstrated.

## 6.3 Directions for Future Work and Open Questions

In this section, we outline key directions for future research that arise from the limitations and contributions of this thesis. The main areas for further investigation include extending the process-centric view, exploring hyperparameter learning, and enhancing educational tools and empirical validation.

**Approximate Gaussian processes**: This thesis focuses on exact Gaussian processes, which scale poorly with large datasets due to the $\mathcal{O}(N^3)$ complexity of inverting the covariance matrix $K$ (see Equation 3.6). Approximate methods, which introduce $M$ pseudo data points ($M \ll N$), offer a more scalable alternative by reducing the computational cost to $\mathcal{O}(N^2M)$. For a detailed discussion of sparse approximation methods, see Snelson and Ghahramani (2005), Bui et al. (2017), and Williams and Rasmussen (2006, Chapter 8). The process-centric view can be extended to approximate Gaussian processes by introducing an approximation operation that results in a more efficient GP. As the approximated GP is still Gaussian, the key operations of marginalization and conditioning retain which is due to the closure (or consistency) property. Additionally, extending this framework to non-Gaussian likelihoods, such as in Gaussian process classification, is a promising avenue for future research.

**Hyperparameter Considerations**: The process-centric view naturally supports online learning, but the current implementation assumed fixed hyperparameters. Further work could extend online learning to sequential hyperparameter learning, especially in the streaming data context to further enhance GP flexibility. Methods for combining hyperparameter optimization in a streaming data setting together with sparse approximate Gaussian processes are for example showcased in Huber (2014) and Bui et al. (2017). Additionally, the role of the mean function hyperparameters, which was not discussed in this thesis, could be explored. While being cautious with overfitting, future work should extend the consideration of hyperparameters to the mean function.

**Further Development of Educational Material and User Testing**: Finally, to increase the accessibility and impact of the educational tools developed in this project, future work could focus on transforming the Jupyter Notebook into more user-friendly formats, such as instructional videos or a standalone web application. These resources would address the limitations of the Colab environment and make the material available to a broader audience. Furthermore, empirical studies are needed to validate the effectiveness of the process-centric framework. For example, comparative studies could assess the learning outcomes of students using the process-centric view versus traditional approaches, providing concrete empirical evidence of its educational benefits.

## 6.4 Conclusion

In conclusion, this thesis has developed a theoretical and practical framework for process-centric Gaussian processes, providing simple, intuitive explanations and interactive plots. A summary of this work, containing both theory and practical examples, is provided in a Colab-hosted Juypter Notebook.

# References

Al-Gahmi, A., Zhang, Y., & Valle, H. (2022). Jupyter in the classroom: An experience report. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*, 425–431.

Amoudi, G., & Tbaishat, D. (2023). Interactive notebooks for achieving learning outcomes in a graduate course: A pedagogical approach. *Education and Information Technologies*, *28*(12), 16669–16704.

Bobek, E., & Tversky, B. (2016). Creating visual explanations improves learning. *Cognitive research: principles and implications*, *1*, 1–14.

Bui, T. D., Yan, J., & Turner, R. E. (2017). A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, *18*(104), 1–72.

Chen, Y., Huang, A., Wang, Z., Antonoglou, I., Schrittwieser, J., Silver, D., & de Freitas, N. (2018). Bayesian optimization in alphago. *arXiv preprint arXiv:1812.06855*.

Deisenroth, M. P., Fox, D., & Rasmussen, C. E. (2013). Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, *37*(2), 408–423.

Deisenroth, M. P., van der Wilk, M., & Luo, Y. (2020, December). A practical guide to gaussian processes. https://infallible-thompson-49de36.netlify.app/

Duvenaud, D. (2014). *Automatic model construction with gaussian processes* [Doctoral dissertation].

Foreman-Mackey, D., Agol, E., Ambikasaran, S., & Angus, R. (2017). Fast and scalable gaussian process modeling with applications to astronomical time series. *The Astronomical Journal*, *154*(6), 220.

Fuady, R., & Mutalib, A. A. (2018). Audio-visual media in learning. *Journal of K6 Education and Management*, *1*(2), 1–6.

Geten, Y. (2019). Gaussian processes: A comprehensive introduction [Accessed: 2024-08-08]. https://yugeten.github.io/posts/2019/09/GP/

Görtler, J., Kehlbeck, R., & Deussen, O. (2019). A visual exploration of gaussian processes [https://distill.pub/2019/visual-exploration-gaussian-processes]. *Distill*. https://doi.org/10.23915/distill.00017

Hensman, J., & Rasmussen, C. E. (2024). *Gaussian process modelling* [Unpublished draft].

Huber, M. F. (2014). Recursive gaussian process: On-line regression and learning. *Pattern Recognition Letters*, *45*, 85–91.

Hwang, S.-g., L'Huillier, B., Keeley, R. E., Jee, M. J., & Shafieloo, A. (2023). How to use gp: Effects of the mean function and hyperparameter selection on gaussian process regression. *Journal of Cosmology and Astroparticle Physics*, *2023*(02), 014.

John, T. (2021). Interactive visualization of gaussian processes [Accessed: 2021-01-01]. http://www.infinitecuriosity.org/vizgp/

Just, G. A. (2010). *The effect of online interactive visuals on undergraduate mathematics learning*. Northern Illinois University.

Kosslyn, S. M. (2006). *Graph design for the eye and mind*. OUP USA.

Lamperti, J. (2012). *Stochastic processes: A survey of the mathematical theory* (Vol. 23). Springer Science & Business Media.

Lederer, A., Conejo, A. J. O., Maier, K. A., Xiao, W., Umlauft, J., & Hirche, S. (2021). Gaussian process-based real-time learning for safety critical applications. *International Conference on Machine Learning*, 6055–6064.

MacKay, D. J., et al. (1998). Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, *168*, 133–166.

McAllister, R. T., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., & Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning.

Murphy, K. P. (2022). *Probabilistic machine learning: An introduction*. MIT press.

Nelson, M. J., & Hoover, A. K. (2020). Notes on using google colaboratory in ai education. *Proceedings of the 2020 ACM conference on innovation and Technology in Computer Science Education*, 533–534.

Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.

Park, S. I., Lee, G., & Kim, M. (2009). Do students benefit equally from interactive computer simulations regardless of prior knowledge levels? *Computers & Education*, *52*(3), 649–655.

Patwardhan, M., & Murthy, S. (2015). When does higher degree of interaction lead to higher learning in visualizations? exploring the role of 'interactivity enriching features'. *Computers & Education*, *82*, 292–305.

Pavliotis, G. A. (2014). Stochastic processes and applications. *Texts in applied mathematics*, *60*.

Rasmussen, C., & Ghahramani, Z. (2000). Occam's razor. *Advances in neural information processing systems*, *13*.

Rasmussen, C. E. (1997). *Evaluation of gaussian processes and other methods for non-linear regression* [Doctoral dissertation, University of Toronto Toronto, Canada].

Roelants, P. (2019). Gaussian processes (1/3) - from scratch. https://peterroelants.github.io/posts/gaussian-process-tutorial/

Schulz, E., Speekenbrink, M., & Krause, A. (2018). A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, *85*, 1–16.

Sharp, H. R., & Rogers, Y. (n.d.). Y. & preece, j.(2007). *Interaction design: beyond human-computer interaction*.

Snelson, E., & Ghahramani, Z. (2005). Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, *18*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Turner, R. (2016). *Machine learning tutorial: Gaussian processes* [Accessed: 2024-08-08]. https://www.youtube.com/watch?v=92-98SYOdlY

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of brownian motion. *Physical Review*, *36*, 823–841.

Wang, J. (2023). An intuitive tutorial to gaussian processes regression. *Computing in Science & Engineering*.

Williams, C., & Rasmussen, C. (2006). *Gaussian processes for machine learning* (Vol. 2). MIT press Cambridge, MA.

Yuen, T. T., & Robbins, K. A. (2014). A qualitative study of students' computational thinking skills in a data-driven computing class. *ACM Transactions on Computing Education (TOCE)*, *14*(4), 1–19.

# Appendix A

# Results of the small user survey

A small user survey with nine computer-affine undergraduate and postgraduate STEM students was conducted. A written participant consent form was obtained prior to the study. The most interesting results are shown in the following. Although not being representative due to a very small sample size (N=9), they give some indications of the usability of an interactive Jupyter Notebook.

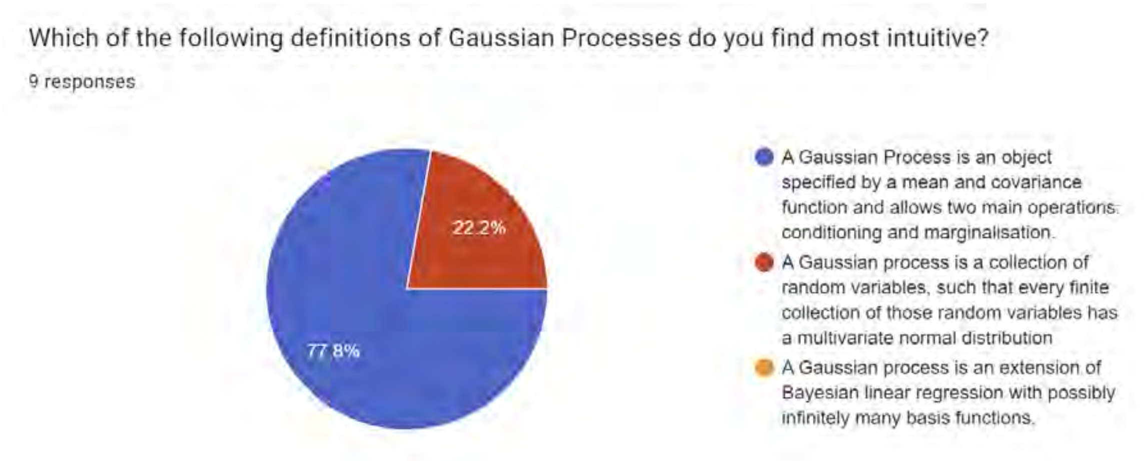


Fig. A.1 7 out of 9 Users reported after completing the tutorial to prefer the process-centric definition for being most intuitive.

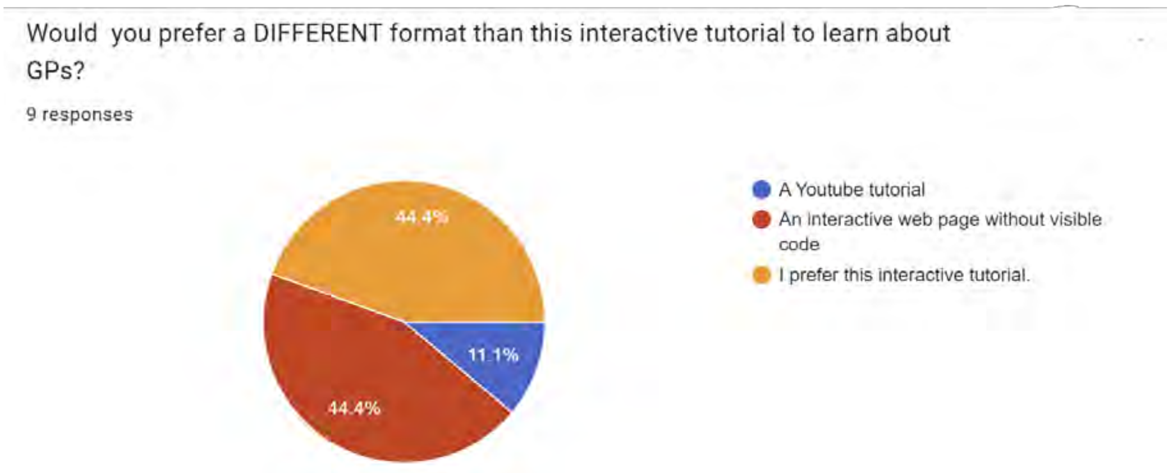


Fig. A.2 4 out of 9 Participants would prefer an interactive web page over a Notebook.

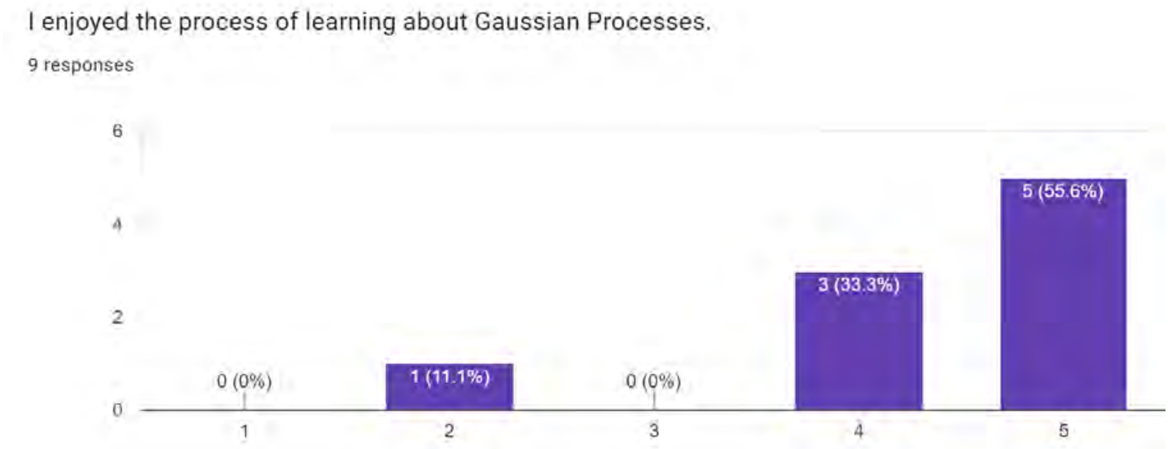


Fig. A.3 8 out of 9 Participants agree or fully agree that they enjoyed the process of learning about GPs.

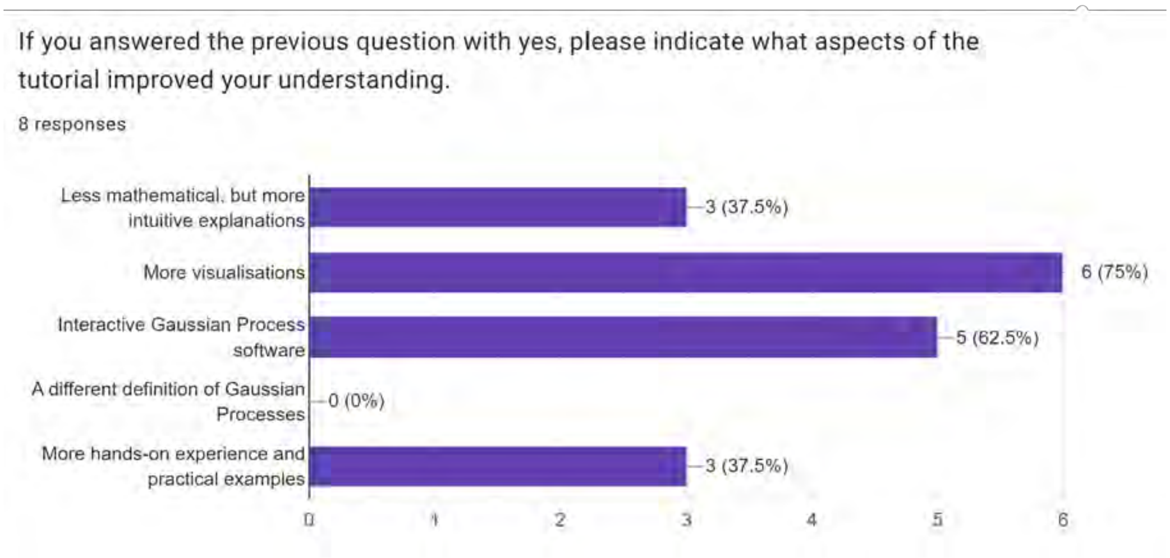


Fig. A.4 7 out of 9 participants reported that the visualisations helped with the understanding.

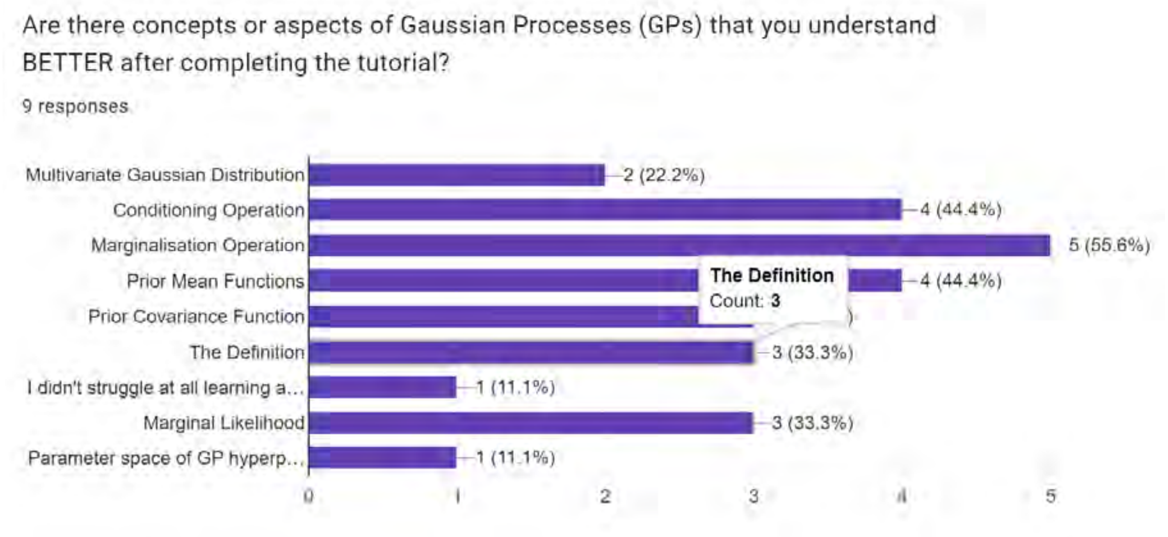


Fig. A.5 The tutorial was most helpful in improving participant's understanding of the marginalization operation.