AGH UNIVERSITY OF KRAKOW

**FIELD OF SCIENCE ENGINEERING AND TECHNOLOGY**

SCIENTIFIC DISCIPLINE AUTOMATION, ELECTRONICS, ELECTRICAL ENGINEERING, AND SPACE TECHNOLOGIES

# DOCTORAL THESIS

## Generative Gaussian Process Models in Functional Phenomena Analysis and Fault Detection

Author: Adrian Dudek

Supervisor: Prof. Jerzy Baranowski, PhD, DSc

Completed in: Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering

Kraków, 2024

AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**DZIEDZINA NAUK INŻYNIERYJNO-TECHNICZNYCH**

DYSCYPLINA AUTOMATYKA, ELEKTRONIKA, ELEKTROTECHNIKA I TECHNOLOGIE KOSMICZNE

# ROZPRAWA DOKTORSKA

## Generatywne modele procesów Gaussowskich w analizie zjawisk funkcjonalnych i wykrywaniu usterek

Autor: Adrian Dudek

Promotor rozprawy: Prof. dr hab. inż. Jerzy Baranowski

Praca wykonana: Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Kraków, 2024

*I would like to express my heartfelt gratitude to my supervisor, Professor Jerzy Baranowski, for his invaluable guidance and support throughout the preparation of this dissertation. Your encouragement, patience, and the time you've devoted over the years have been deeply appreciated.*

*To my colleagues from the LISZ Group at KAiR, AGH, thank you for sharing your extensive knowledge and offering your assistance time and time again.*

*My deepest thanks go to my fiancée, Marta, whose unwavering belief in me, constant motivation, and deep understanding have been the cornerstone of my journey. Your support means the world to me, and I'm grateful for your presence in my life.*

*To my parents, Krystyna and Jerzy, my sister Angelika, and my entire family, thank you for your care, for being models of perseverance, and for all the kind words and gestures that have carried me forward.*

*Finally, I extend my gratitude to everyone who contributed in any way to the completion of this dissertation – thank you!*

# Abstract

My doctoral dissertation focuses on the application of Gaussian Processes (GP) for modeling and analyzing complex functional phenomena, as well as fault detection in technical systems. In my research, I aimed to extend the use of Gaussian Processes in technical sciences, particularly in the context of system diagnostics characterized by nonlinear dependencies and sudden changes in state. The main challenge I address involves effectively modeling these systems while accounting for prediction uncertainty and use of generative models in case of lack of data.

The aim of this dissertation was to prove two theses: 1. *Gaussian Process models can fill a gap in functional phenomena analysis and diagnostics* by modeling complex nonlinear dependencies and providing uncertainty quantification, which is crucial for accurate predictions and interpretable results; and 2. *generative Gaussian Process models can be useful in diagnostics when data is scarce*, as they allow for the creation of synthetic datasets that reflect the functional properties of the system, enhancing the effectiveness of diagnostic algorithms.

To prove these theses my work focuses on several key research problems. First and foremost, I investigated the use of GP for anomaly detection, especially in industrial processes where instability and malfunctions can have serious operational consequences. Due to their probabilistic nature, GP models not only allow for the prediction of future system behavior but also enable the estimation of the uncertainty of these predictions. In the context of fault detection, GP models are particularly effective in identifying transient states and unexpected anomalies, which are often difficult to detect using traditional diagnostic methods.

GP models can be employed to generate synthetic data for further analysis, especially in situations where real-world data is limited or difficult to obtain. This approach is crucial when collecting sufficient data in real conditions is costly or time-consuming. By generating reliable data, GP models can support other diagnostic algorithms, improving their effectiveness and enhancing fault detection accuracy.

My research also involved developing methods to improve the computational efficiency of GP models. Traditionally, Gaussian Processes require complex computations, particularly when working with large datasets. To minimize these challenges, I applied Chebyshev approximation and Fast Fourier Transform (FFT). This led to significantly reduced computational burdens without sacrificing the quality of the results, as well as maintain the functional behavior of the modelled data. These solutions made GP models more practical for real-time applications, especially in industrial environments.

In my doctoral dissertation, GP models were applied in various technical contexts. I used them for the diagnostics of lithium-ion battery health, where accurate predictions of battery wear and the remaining useful life are crucial for resource management. These models proved particularly effective in predicting battery health under different operating conditions, providing better insights into the degradation process.

Additionally, I used GP models for spatial modeling of air pollution, allowing for more precise predictions even in regions with limited monitoring stations. Another area of application was the detection of anomalies in industrial process signals, where GP enabled the identification of anomalies in transient states that were previously difficult to detect with traditional methods. Additionally, I analyzed the possibilities of classifying devices faults using GP-based Gaussian mixture models.

In conclusion, my doctoral research introduces innovative approaches to using Gaussian Processes in technical system diagnostics, both in the context of prediction and fault detection. My research provides practical solutions to computational challenges associated with GP and demonstrates how they can be effectively applied in various fields, such as electrical engineering, spatial modeling, and industrial process diagnostics.

# Streszczenie

Praca doktorska skupia się na wykorzystaniu Procesów Gaussowskich (GP) do modelowania i analizy złożonych zjawisk funkcjonalnych oraz detekcji usterek w systemach technicznych. W pracy podejmuję próbę rozszerzenia zakresu zastosowań procesów gaussowskich w naukach technicznych, zwłaszcza w kontekście diagnostyki systemów, które charakteryzują się nieliniowymi zależnościami oraz nagłymi zmianami stanu. Główne wyzwanie, z jakim się mierzę, dotyczy efektywnego modelowania tych systemów przy jednoczesnym uwzględnieniu niepewności predykcji oraz wykorzystania modeli generatywnych w przypadku niewystarczających danych.

Celem tej pracy doktorskiej było udowodnienie dwóch tez: 1. *Procesy Gaussowskie mogą wypełnić lukę w analizie zjawisk funkcjonalnych i diagnostyce* poprzez modelowanie złożonych nieliniowych zależności i dostarczanie kwantyfikacji niepewności, co jest kluczowe dla dokładnych prognoz i interpretowalnych wyników; oraz 2. *generatywne modele Procesów Gaussowskich mogą być użyteczne w diagnostyce*, gdy brakuje danych, ponieważ pozwalają na tworzenie syntetycznych zbiorów danych odzwierciedlających właściwości funkcjonalne systemu, co zwiększa skuteczność algorytmów diagnostycznych.

Aby udowodnić te tezy w pracy tej koncentruję się na kilku kluczowych problemach badawczych. Przede wszystkim badałem zastosowanie GP do wykrywania anomalii, szczególnie w procesach przemysłowych, gdzie niestabilności i usterki mogą mieć poważne konsekwencje operacyjne. Modele GP, dzięki swojej probabilistycznej naturze, oferują nie tylko możliwość predykcji przyszłego zachowania systemu, ale również umożliwiają szacowanie niepewności tych przewidywań. W kontekście detekcji usterek, GP pozwalają na lepsze wykrywanie stanów przejściowych oraz niespodziewanych anomalii, które mogą być trudne do zidentyfikowania za pomocą tradycyjnych metod diagnostycznych.

Modele GP mogą być stosowane do tworzenia syntetycznych danych na potrzeby dalszych analiz. Jest to efektywne szczególnie w przypadkach, gdy rzeczywiste dane są ograniczone lub trudno dostępne. Takie podejście ma ogromne znaczenie w sytuacjach, gdzie zebranie wystarczającej ilości danych w rzeczywistych warunkach jest kosztowne lub czasochłonne. Dzięki możliwości generowania wiarygodnych danych, GP mogą wspomagać inne algorytmy diagnostyczne, poprawiając ich skuteczność i zwiększając dokładność detekcji usterek.

Moje badania obejmowały również rozwój metod poprawiających efektywność obliczeniową modeli GP. Tradycyjnie procesy gaussowskie wymagają skomplikowanych obliczeń, zwłaszcza przy pracy z dużymi zbiorami danych. W celu zminimalizowania tych trudności, zastosowałem metody takie jak aproksymacje Czebyszewa oraz szybką transformata Fouriera (FFT). To pozwoliło na znaczną redukcję obciążeń obliczeniowych, bez utraty jakości wyników, a co więcej na zachowaniu funkcjonalnych własności modelowanych danych.

W ramach mojej pracy doktorskiej modele GP zostały zastosowane w różnych kontekstach technicznych. Zastosowałem je do diagnostyki stanu zdrowia baterii litowo-jonowych, gdzie precyzyjne prognozowanie zużycia i ocena pozostałej żywotności mają kluczowe znaczenie dla zarządzania zasobami energetycznymi. Modele te okazały się szczególnie skuteczne w przewidywaniu stanu zdrowia baterii w różnych warunkach operacyjnych, oferując lepsze zrozumienie procesu degradacji. Ponadto, za pomocą GP modelowałem przestrzenne zmiany poziomu zanieczyszczeń powietrza, co pozwoliło na bardziej precyzyjne przewidywania, nawet w regionach o ograniczonej liczbie stacji pomiarowych. Kolejnym obszarem zastosowania była detekcja anomalii w sygnałach procesów przemysłowych, gdzie GP pozwoliły na identyfikację nieprawidłowości w stanach przejściowych, które wcześniej były trudne do wykrycia za pomocą tradycyjnych metod. Ponadto analizowałem możliwości klasyfikacji usterek urządzeń przy użyciu modeli mikstur Gaussowskich opartych na GP.

Podsumowując, moja praca doktorska wprowadza nowatorskie podejścia do wykorzystania procesów gaussowskich w diagnostyce systemów technicznych, zarówno w kontekście predykcji, jak i wykrywania usterek. Moje badania dostarczają praktyczne rozwiązania dla problemów obliczeniowych związanych z GP oraz pokazują, jak można je efektywnie zastosować w różnych dziedzinach, takich jak inżynieria elektryczna, modelowanie przestrzenne i diagnostyka procesów przemysłowych.

# List of Abbreviations

| | | |
|---|---|---|
| GP | Gaussian Process | see page 2 |
| RBF | Radial Basis Function | see page 6 |
| MCMC | Markov Chain Monte Carlo | see page 11 |
| HMC | Hamiltonian Monte Carlo | see page 11 |
| INLA | Integrated Nested Laplace Approximation | see page 14 |
| GMRF | Gaussian Markov Random Field | see page 14 |
| MLE | Maximum Likelihood Estimator | see page 10 |
| ML-II | Maximum Likelihood Type II | see page 10 |
| GMM | Gaussian Mixture Model | see page 19 |
| SoH | State of Health | see page 19 |
| FFT | Fast Fourier Transform | see page 18 |

# List of Mathematical Symbols

| | | |
|---|---|---|
| $S$ | Set of data points in a spatial or time domain | see page 4 |
| $z_n$ | Random variable with normal marginal distribution | see page 4 |
| $\mathbb{N}$ | Set of natural numbers | see page 4 |
| $\mathbb{R}$ | Set of real numbers | see page 4 |
| $\mathrm{Normal}$ | Normal (Gaussian) distribution | see page 4 |
| $K$ | Covariance matrix | see page 4 |
| $\mu_{x_1,\ldots,x_n}$ | Mean vector | see page 4 |
| $\mu(\cdot)$ | Mean function of a Gaussian Process | see page 5 |
| $k(\cdot,\cdot')$ | Covariance function of a Gaussian Process | see page 5 |
| $p(f,\tilde{f})$ | Joint probability distribution of $f$ and $\tilde{f}$ | see page 6 |
| $\mathrm{E}[\cdot]$ | Expected value | see page 5 |
| $l$ | Characteristic length scale | see page 6 |
| $d(x_i,x_j)$ | Euclidean distance between points $x_i$ and $x_j$ | see page 6 |
| $\Gamma(\nu)$ | Gamma function | see page 6 |
| $K_\nu(\cdot)$ | Modified Bessel function of the second kind | see page 6 |
| $\nu$ | Smoothness parameter in Matérn kernel | see page 6 |
| $\mathbf{w}$ | Model parameters (weights) | see page 10 |
| $\theta$ | Hyperparameters of the model | see page 10 |
| $X$ | Input data (features) | see page 10 |
| $y$ | Observed data | see page 10 |
| $p(\mathbf{w} \mid \mathbf{y}, X, \boldsymbol{\theta})$ | Posterior probability of parameters | see page 10 |
| $\mathcal{H}_i$ | Model structure | see page 10 |
| $p(\mathbf{y} \mid X, \mathbf{w}, \mathcal{H}_i)$ | Likelihood of the data given model parameters | see page 10 |

| | | |
|---|---|---|
| $H(\rho, \Theta)$ | Hamiltonian function | see page 12 |
| $T(\rho \mid \Theta)$ | Kinetic energy | see page 12 |
| $V(\Theta)$ | Potential energy | see page 12 |
| $\rho$ | Momentum variable | see page 12 |
| $\Theta$ | Parameters (position) | see page 12 |
| $M$ | Mass (metric) matrix | see page 12 |
| $\mathrm{Multinormal}$ | Multinormal (Gaussian) distribution | see page 12 |
| $H(\rho^*, \Theta^*)$ | Hamiltonian at the proposed new state | see page 12 |
| $P_m$ | Metropolis acceptance probability | see page 12 |
| $\mathrm{Cholesky}(\cdot)$ | Cholesky decomposition of matrix | see page 11 |
| $\mathcal{K}$ | Laplace operator | see page 13 |
| $S(\cdot)$ | spectral density of covariance matrix | see page 14 |
| $\mathbf{Q}(\theta)$ | Precision matrix dependent on hyperparameters | see page 15 |

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Modern industry relies on innovative technologies that enable automation, enhance efficiency, and increase precision in production processes. Examples of devices that perform such tasks include electric motors, pumps, assembly lines, and cooling systems. Industrial batteries also play a crucial role by powering various equipment, while bearings ensure the smooth operation of moving machinery, guaranteeing reliability and durability in demanding working conditions. However, over time, every machine experiences wear and tear or even failures, leading to production downtime and increased operational costs. Unfortunately, manufacturers of industrial equipment rarely provide tools for self-diagnostics, making it difficult to quickly detect and resolve issues. As a result, machine diagnostics, based on advanced analytical methods, is gaining importance. It allows for continuous monitoring of the technical condition of devices, predicting potential malfunctions, and preventing production stoppages, which is essential for modern production management.

Traditional approaches, which primarily rely on statistical models and machine learning techniques based on historical data, are commonly used to predict system behavior. However, in the face of complex systems, these classical methods can sometimes fall short, especially when dealing with unexpected changes or anomalies in system performance. This presents an opportunity to explore new approaches, which could lead to innovative solutions that better address the challenges faced by these systems.

In the diagnostics of technical systems, classical mathematical models such as linear regression or ARMA (AutoRegressive Moving Average) models are often used, assuming system stability under certain conditions. Judging by Box and Jenkins (1970), these methods are effective for linear time series forecasting, but they often struggle in handling non-stationarity and nonlinear relationships. Technical systems, characterized by nonlinear dependencies and fluctuating behavior, present challenges that these methods are not always able to address. The limitations of these approaches become particularly evident in situations where atypical transitory states or unforeseen anomalies occur.

The introduction of machine learning techniques, such as Random Forests or neural networks, has led to advancements in the analysis of large datasets and the identification of patterns. These algorithms handle nonlinearities and complex dependencies effectively, which are particularly valuable in the context of industrial systems. However, their biggest limitation remains the difficulty in interpreting results – many of these models operate as "black boxes," meaning that although they can generate precise predictions, users have

limited insight into how these results are produced. Neural networks like the Multilayer Perceptron (MLP) described by Bishop (1995) are commonly known and used, for instance in (Bikku 2020), where we can find that these neural networks approaches are known for their efficiency, but their structure makes it difficult to understand the influence of individual variables on the final model decisions. Additionally, many algorithms lack tools to assess the uncertainty of predictions, which can lead to potentially risky decisions in critical applications.

That highlights the need for more advanced functional analysis methods. These methods should account for both the dynamics of transitory states and the complex interactions between individual system components. The existing gap in the diagnostics of industrial systems calls for modern tools that not only predict system behavior but also allow for a better understanding of internal dependencies and real-time anomaly detection.

In response to these challenges, Gaussian Processes (GP) offers an innovative approach to modeling technical systems. Due to their ability to model complex nonlinear dependencies and estimate confidence intervals, GPs not only facilitate a better understanding of system dynamics, but also support precise anomaly detection. Their flexibility makes them a competitive tool for analyzing technical systems, especially in cases where uncertainty modeling and real-time data variability are required.

In light of the mentioned challenges, I focused on solutions which could be applied to the functional diagnostics of engineering systems, aiming to address the identified gaps in current methodologies. In my review paper (Dudek and Baranowski 2022), which aligns with the common trend observed across multiple reviews in the field, such as (Gonzalvez, Lezmi, Roncalli, and Xu 2019), it was established that GP models have been gaining wider recognition, largely driven by advancements in computational power. This is crucial because GPs, as kernel-based methods, rely heavily on intensive computations. The growing availability of more powerful computing tools and resources has made it feasible to apply GPs to more complex and large-scale problems.

My involvement in a project funded by the NCN, titled 'Prediction and Detection of Process Faults' driven and motivated me to focus on problems related to anomaly detection, especially in industrial processes. Anomalies in these environments can lead to severe disruptions and early detection is essential for ensuring system stability and operational efficiency. GPs offer a unique advantage in this domain due to their probabilistic nature, which allows for not only prediction but also the quantification of uncertainty. This capability is particularly beneficial for detecting rare or unexpected events that deviate from normal operating conditions.

Moreover, another aspect of GPs is their ability to serve as a generative tool. In cases where there is a scarcity of data, a common issue across various domains, GPs can generate synthetic data that is critical for training models and enhancing the robustness of anomaly detection algorithms. Data scarcity refers to situations where there is an insufficient amount of data to properly train models, leading to unbalanced datasets. In many real-world applications, the available data may not evenly represent all possible states of a system, especially rare or abnormal conditions. This can cause models to overfit to the dominant class or fail to generalize to unseen scenarios. The lack of data also necessitates the use of statistical models, such as GPs, which can incorporate prior knowledge and uncertainty into the learning process.

GPs can model data with high depth and accuracy, creating artificial datasets that mirror the functional behavior of the system, whether in its healthy or abnormal state. This capability is particularly valuable in industrial settings where obtaining vast amounts of real-world data can be costly or impractical. Additionally, in fields such as healthcare or aerospace, where collecting real-time data can be expensive, risky, or constrained by privacy regulations, the ability of GPs to simulate realistic data distributions becomes even more crucial. By supplementing limited datasets with reliable synthetic data, GPs can help mitigate the risks of overfitting, enhance the performance of models on unbalanced datasets, and improve model generalization across a broader range of operational conditions.

The generated data are highly reliable because they are derived from the underlying functional analysis, a core strength of GP models. By leveraging this property, GPs not only assist in detecting anomalies but also act as a supplementary source of data that can improve the performance of other algorithms, especially in scenarios where data scarcity limits the applicability of traditional models. This dual role of GPs—as both a diagnostic and generative tool—positions them as a powerful approach for advancing the field of industrial process diagnostics.

## 1.2 Gaussian Processes

GPs form a framework for modeling and making predictions about uncertain systems, widely used in many fields. At their core, GPs provide a principled approach to learning functions from data by treating function values as random variables that exhibit a joint Gaussian distribution. Before exploring the details of GPs, it is crucial to understand the fundamental building block upon which they are constructed: the normal (or Gaussian) distribution. The normal distribution is a continuous probability distribution characterized by a symmetric, bell-shaped curve. It is defined for a random variable $X$ with a mean $\mu$ and variance $\sigma^2$, and is given by the probability density function:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{1}$$

The parameters $\mu$ and $\sigma^2$ describe the center and the spread of the distribution, respectively. When $\mu = 0$ and $\sigma^2 = 1$, the distribution is referred to as the standard normal distribution. The normal distribution's significance lies in its ubiquity in natural phenomena and its key properties, such as being completely determined by its first two moments (mean and variance) and the fact that linear combinations of normal random variables remain normally distributed.

The multivariate normal distribution generalizes the normal distribution to multiple dimensions, describing a vector of jointly normally distributed variables. It is characterized by a mean vector $\mu$ and a covariance matrix $K$, defining the central location and shape of the distribution, respectively. The probability density function for a multivariate normal distribution is given by:

$$p(x) = \frac{1}{(2\pi)^{n/2}|K|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T K^{-1}(x-\mu)\right), \tag{2}$$

where $x$ is an $n$-dimensional random vector, $\mu \in \mathbb{R}^n$ is the mean vector, and $K = K^{\mathsf{T}} \in \mathbb{R}^{n\times n}$ is the positive definite covariance matrix. These properties are essential in defining GP.

### 1.2.1 General definition

The definition of GPs can be expressed in multiple equivalent ways. Various definitions are used depending on their application, whether for practical understanding or for exploring the mathematical properties of GP. Additionally, GPs are based on the Bayesian statistics. In Bayesian inference, there is a distinction between the prior and the posterior. Prior is a distribution over the possible functions before observing any data and the posterior, on the other hand, is the updated distribution over functions after incorporating observed data. This prior reflects assumptions about properties like smoothness, continuity, or periodicity of the function. The posterior takes into account both the observed data and prior assumptions, providing not only predictions but also uncertainty estimates. Bayesian inference in GPs allows for natural incorporation of uncertainty in the predictions, giving a full distribution of possible functions based on the data.

The simplest and most commonly encountered method of defining GP itself, particularly in applied sciences such as machine learning, can be found in works like C. Rasmussen and Williams (2006).
**Definition:** Let $S$ be a set of data $t$ in a spatial or time domain (or both). Then Gaussian Process on this set $S$ is defined as a collection $Z_t$ of $z_n$ elements, where each element $z_n$ is a random variable with a normal marginal distribution. Joint distribution of the entire set $Z$ is also normally distributed.
It can be also presented in the form shown below:

$$
\begin{aligned}
S &= \{t_1, t_2, \ldots, t_n\} \\
Z_t &= \{z_{t_1}, z_{t_2}, \ldots, z_{t_n}\} \\
&\quad \text{where } t \in S \\
&\quad \text{then} \\
\forall n \in \mathbb{N}, \forall t_1, \ldots, t_n &: (Z_{t_1}, \ldots, Z_{t_n}) \\
&\quad \text{is multivariate normal distribution}
\end{aligned}
$$

Above definition is underlaying for many applied science approaches, demonstrating its broad relevance and utility across various fields and it was based on (C. Rasmussen and Williams 2006) approach to GP where they came up with following definition, which captures the essence of GP:

> *"A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution."*

The interpretation presented by Rasmussen is straightforward and can be used to define a stochastic model for an unknown function, where the model base on Gaussian distribution. A stochastic model for an unknown function $f(x)$ can be constructed by ensuring that it exhibits Gaussian properties. This involves modeling the joint distribution of $f$ at various points $x_1,\ x_2,\ \ldots,\ x_n$ within a domain $X$. For example, when modeling a variable observed at discrete locations, the joint distribution $(f(x_1), \ldots, f(x_n))^\mathsf{T}$ is assumed to follow a multivariate normal distribution:

$$
(f(x_1), \ldots, f(x_n))^\mathsf{T} \sim \text{Normal}(\mu_{x_1,\ldots,x_n}, K_{x_1,\ldots,x_n}), \tag{3}
$$

where $\mu_{x_1,\ldots,x_n} \in \mathbb{R}^n$ represents the mean vector and $K_{x_1,\ldots,x_n} \in \mathbb{R}^{n\times n}$ denotes the covariance matrix (represents covariance between points). The mean $\mu$ can be any function, while the covariance matrix elements

$K = k(x_i, x_j)$ must satisfy stringent conditions to ensure that $k(\cdot, \cdot)$ defines a valid covariance function. These conditions are crucial for maintaining the model's consistency and tractability.

### 1.2.2 Covariance and mean functions

The formalization of probability theory done by Kolmogorov (1956) led to significant contribution called the Kolmogorov extension theorem, which provides the precise conditions under which finite-dimensional distributions, such as $f(x_1), \ldots, f(x_n))^\mathsf{T}$, can be extended to describe a valid random function $f(x)$. As shown, for example by Simpson (2021a) there are two critical conditions:

1. The order of the observations does not affect the results in any substantial way. In practice, changing the order merely permutes the rows and columns of the mean vector and covariance matrix, which is permissible.

2. There exists a consistent way to map between the distributions of $(f(x_1), \ldots, f(x_n), f(x_{n+1}))^\mathsf{T}$ and $(f(x_1), \ldots, f(x_n))^\mathsf{T}$. This consistency imposes a strict constraint on the covariance function.

In essence, it is crucial to ensure that, as we expand our covariance matrix by adding rows and columns, it remains positive definite (i.e., all eigenvalues must remain non-negative). This requirement is essential for maintaining the validity of the multivariate normal distribution. The specific condition that must be met is that for any positive integer $n$ and any set of points $x_1, \ldots, x_n$, and for any non-zero $a_1, \ldots, a_n$, the following inequality must hold:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) \geq 0. \tag{4}$$

This condition is notoriously difficult to verify, which is why the choice of a covariance function is typically limited to a small set of well-understood functions commonly found in GP literature.

Assuming that the chosen (or constructed) covariance function meets the required conditions, it can be used to define our GP. The covariance function establishes the dependency structure between different points in the input space, and its correctness ensures that the GP conforms to the properties such as positive definiteness and the proper definition of marginal distributions for any subset of data. With known covariance function, we can create the covariance matrix, where each entry represents covariance between two points calculated by the covariance function. By adding a mean function, which represents the expected value of the GP at each point in the input space, we can fully define our GP. The mean function specifies the central tendency of the process. It can be a constant function, representing a fixed mean across the input space, or a more complex function that varies with its variable, allowing the GP to model more intricate trends within the data. Based on (C. Rasmussen and Williams 2006; Simpson 2021a) the definition of GP containing mean function $\mu(x)$ and the covariance function $k(x, x')$ of a real process $f(x)$ can be written as follows:

$$\mu(x) = E[f(x)] \tag{5}$$

$$k(x, x') = E[f(x) - \mu(x))(f(x') - \mu(x'))] \tag{6}$$

Then we can define GP as:

$$f(x) \sim GP(\mu(x), k(x, x')) \tag{7}$$

The mean and covariance functions completely characterize the GP prior, which controls the full behavior of funtion $f$ prior. If we assume that $f = \{f(x_n)\}_{n=1}^{N}$ then the prior distribution of $f$ is a multivariate Gaussian distribution $f \sim \text{Normal}(\mu, K)$, where mean is $\mu = \{\mu(x_n)\}_{n=1}^{N}$, and $K$ is, as denoted earlier, covariance matrix, where $K_{i,j} = k(x_i, x_j)$ the joint distribution of $f$, which represents original function (observations) and a new $\tilde{f}$, which represents values from GP, is also a multivariate Gaussian with assumption of mean equal 0 is presented by the formula:

$$p(f, \tilde{f}) = \text{Normal}\left( \begin{bmatrix} f \\ \tilde{f} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} K_{f,f} & k_{f,\tilde{f}} \\ k_{\tilde{f},f} & k_{\tilde{f},\tilde{f}} \end{bmatrix} \right) \tag{8}$$

where $f$ are the observed values, $\tilde{f}$ are predicted values from model, $k_{f,\tilde{f}}$ is the covariance between $f$ and $\tilde{f}$ , and $k_{\tilde{f},\tilde{f}}$ is the prior variance of $\tilde{f}$. Covariance functions may also depends on hyperparameters, but I omitted them here to ease the notation. On such a prior distribution, we then make an inference from GP model, by adding the data from the process we would like to model.

As established, for instance by C. Rasmussen and Williams (2006) the GP can be fully specified by defining the mean and covariance functions. The mean function is often set to zero, as this simplifies calculations and is typically sufficient. However, there are cases where adjusting the mean is beneficial, such as for improving model interpretability or better reflecting prior knowledge. The covariance function (also known as the kernel function) represents the similarity between data points. As mentioned, we generally select them from a set of predefined functions, and that choise should ideally reflect the prior beliefs about the problem. In principle established in (Schölkopf and Smola 2002), any function that generates a positive definite covariance matrix can be used, however, as I mentioned before, designing a new covariance function that is both mathematically valid and practically useful is often challenging.

The most basic and commonly used kernel is the Radial Basis Function (RBF), also known as the Squared Exponential (SE), defined by Powell (1985) and can be formulated with example formula from (C. Rasmussen and Williams 2006):

$$k(x_i, x_j) = \exp\left( -\frac{d(x_i, x_j)^2}{2l^2} \right) \tag{9}$$

Its primary characteristic is that its value depends only on $d(x_i, x_j)$ which is the Euclidean distance between points, calculated by measuring the length of the shortest path connecting them. The formula for calculating such distance between points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ can be formulated as follows:

$$d(P, Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{10}$$

RBF kernel's parameter (kernels' parameters are also called hyperparameters) $l$ is known as the *characteristic length scale*. The RBF is infinitely differentiable, meaning that the GP with this kernel has mean square derivatives of all orders. Despite the smoothness of the RBF, strong smoothness assumptions are often unrealistic, and it is recommended to use a kernel from the Matérn family.

Also in research by Matérn (1960) the Matérn kernel family was defined as generalization of the RBF function. The general formula for this covariance functions is given by:

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left( \frac{\sqrt{2\nu}}{l} d(x_i, x_j) \right)^{\nu} K_\nu \left( \frac{\sqrt{2\nu}}{l} d(x_i, x_j) \right) \tag{11}$$

where $K_\nu$ is a modified Bessel function, $l$ is the length-scale parameter, $\Gamma(\cdot)$ is gamma function. The Gamma function is a generalization of the factorial function to complex and real number arguments, defined as:

$$\Gamma(n) = \int_0^\infty t^{n-1} e^{-t}\, dt \quad \text{for} \quad n > 0. \tag{12}$$

and for positive integers, $\Gamma(n) = (n-1)!$. The Bessel function of the first kind, denoted as $J_\nu(x)$, is a solution to Bessel's differential equation:

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - \nu^2)y = 0, \tag{13}$$

where $\nu$ is a real or complex number representing the order of the Bessel function. The modified Bessel function of the second kind, used in Matérn kernel function, denoted as $K_\nu(x)$, is a solution to the modified Bessel equation:

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} - (x^2 + \nu^2)y = 0. \tag{14}$$

Unlike the regular Bessel functions, which oscillate, the modified Bessel function $K_\nu(x)$ decays exponentially as $x \to \infty$. It is widely used in problems where the solution exhibits exponential decay or growth. Commonly used values in this kernel for $\nu$ include $3/2$, which models functions that are at least once differentiable, and $5/2$, for functions that are at least twice differentiable.

For different purposes we can use, for instance, the Exp-Sine-Squared kernel function, also known as the periodic kernel, which was proposed by Mackay (1998). This kernel captures the periodicity of functions and is defined by the formula:

$$k\left(x_i, x_j\right) = \exp\left(-\frac{2\sin^2\left(\pi d\left(x_i, x_j\right)/p\right)}{l^2}\right) \tag{15}$$

In addition to the length-scale parameter $l$, this kernel includes a periodicity parameter $p$, which adds flexibility in modeling periodic functions. This kernel is particularly useful for modeling seasonality or other periodic patterns in the data.

Based on my research on state of art of GP, the most commonly used and widely applied kernel functions were the basic kernels such as RBF and the Matérn family. However, in various research studies, I have also spotted approaches based on a variety of different kernels. Additionally, methods that combine features from multiple kernels (by f.e. multiplying two kernels) known as multikernels, have also been utilized. Multikernels allow for more flexible modeling of complex dependencies within data, making them particularly valuable in advanced research and analysis.

### 1.2.3 Other ways to define Gaussian Process

It is worth noting that there are possibilities to define GP within a more mathematically rigorous context. However, my research focused more on diagnostics, fault detection and generative modeling of GP so in my work I mainly referred to the previously mentioned definition as in the current state of art it is used notoriously. This other ways of defining GPs have been well described and compared in (Simpson 2021b). These definitions are mostly based on field theory by Adler and Taylor (2007) and Gaussian measures by Bogachev (2015). Here, due to after mentioned reasons, I intend to only sketch the topic, indicating that there are other possible ways to define GPs.

### Linear Functionals

We can have a GP defined as a collection of random variables $u(s)$, where $s \in S$ and $S$ is a specified index set that is appropriately well-behaved from a topological standpoint. The defining characteristic of a GP is that for every continuous linear functional $\ell(\cdot)$ applied to the process, the resulting distribution is Gaussian. In this context, a linear functional is a mapping $\ell(\cdot)$ that takes a GP and an input, returning a real number, and it satisfies two key properties: linearity and boundedness.

To provide a concrete example, point evaluation $u(s_j)$ (evaluating the function at a specific point) is a linear functional. Similarly, the definite integral of $u(s)$ over a set is also a linear functional. Importantly, if $\ell_1(\cdot), \ldots, \ell_J(\cdot)$ are continuous linear functionals, then the vector $(\ell_1(u), \ldots, \ell_J(u))^\mathsf{T}$ is multivariate Gaussian.

### Abstract Wiener space

The last way of defining a GP, which I would like to present, uses the Reproducing Kernel Hilbert Space (RKHS) as the fundamental construct, particularly when dealing with infinite-dimensional spaces. This construction is often referred to as an abstract Wiener space. It provides a comprehensive framework for some GPs encountered in practical applications.

Concept is based on a fact that any stochastic process can be uniquely defined by its characteristic function (Adler and Taylor 2007). A Gaussian Process $u(s)$ can be defined on a separable Hilbert space $H$. The Hilbert space structure, with an inner product $\langle \cdot, \cdot \rangle_H$, is central to this definition because it allows us to define the covariance structure of the GP. The goal is to construct a GP supported on the $H$, but instead of focusing directly on the covariance function, we focus on the characteristic function of the process. We start by considering a separable Hilbert space $H$ with an inner product $\langle \cdot, \cdot \rangle_H$. So, a GP is defined on any finite-dimensional subspace $V_n \subset H$, with the characteristic function:

$$\phi_{\tilde{u}_n}(h) = \exp\left(-\frac{1}{2}\langle h, h \rangle_H\right). \tag{16}$$

The characteristic function $\phi_{\tilde{u}_n}(h)$ defines the Gaussian nature of the process. A GP is fully characterized by its mean and covariance, and the characteristic function expresses this directly. For a zero-mean GP, the characteristic function tells us that any linear combination of function evaluations follows a normal distribution. The inner product $\langle h, h \rangle_H$ in the Hilbert space $H$ is linked to the covariance of the process. The characteristic function ((16)) reflects the covariance between different function values of the GP. Specifically, this formula encapsulates how much variation the GP has across the space $H$, and the covariance operator determines this variation. In this context, the Hilbert space $H$ can be seen as encoding the smoothness properties of the GP. The characteristic function is tightly connected to the covariance operator of the GP, which operates within this Hilbert space structure.

This approach works well in finite dimensions, but issues arise when attempting to extend the process to the entire Hilbert space $H$. The covariance operator of the GP must be a trace-class operator, defined as:

$$\mathrm{tr}(C_u) = \sum_{n=1}^{\infty} C_u(e_i, e_i) = \mathrm{E}\left[\langle u, u \rangle_H\right] < \infty \tag{17}$$

where $\{e_i\}_{i=1}^{\infty}$ is an orthonormal basis for $H$. However, if the space is infinite-dimensional, the identity operator (which the covariance operator represents) does not satisfy the trace-class condition, resulting in

an infinite trace. This renders the process non-Gaussian on $H$, demonstrating the complexities of extending finite-dimensional GP definitions to infinite-dimensional settings.

To resolve this, a larger Hilbert space $H' \supset H$ is introduced, defined by relaxing the admissibility condition for the sequences $\{a_n\}$:

$$H' = \left\{ \sum_{n=1}^{\infty} a_n e_n : \sum_{n=1}^{\infty} \frac{a_n^2}{n^2} < \infty \right\}. \tag{18}$$

The space $H'$ is larger because it allows sequences that would not satisfy the norm condition in $H$. We can then define a linear embedding $j : H \to H'$, allowing elements of $H$ to be considered in $H'$. If $\{e'_n\}$ is an orthonormal basis for $H'$, then we map elements $h \in H$ to $H'$ using:

$$j\left( \sum_{n=1}^{\infty} \alpha_n e_n \right) = \sum_{n=1}^{\infty} \frac{\alpha_n}{n} e'_n, \tag{19}$$

where $e'_n = n e_n$, making $\|e'_n\|_{H'} = 1$. This scaling ensures $e'_n$ forms an orthonormal basis for $H'$.

Thus, while a GP defined on the original space $H$ encounters difficulties due to the trace condition, by extending to a suitable larger space $H'$, we can effectively model a GP. The characteristic function in the extended space becomes:

$$\phi_u(h) = \exp\left( -\frac{1}{2} \langle C_u h, h \rangle_{H'} \right), \tag{20}$$

where $C_u$ is now a trace-class operator on $H'$, fulfilling the conditions for a GP on $H'$.

In summary, characteristic function from formula (20) describes the GP by encoding its covariance structure in terms of the inner product on the Hilbert space $H$. The covariance operator, reflected in the inner product, governs how the GP behaves over its domain, and the characteristic function ensures that any finite-dimensional projection of the GP is still Gaussian. This abstract approach allows the GP to be defined without needing an explicit covariance function but instead relying on the properties of the Hilbert space. The trace-class operator condition in the covariance ensures that the GP behaves well (i.e., has finite variance) across the space.

### 1.2.4 Computation of Gaussian Processes

GPs are rooted in the hierarchical Bayesian model framework, which provides a structured way to model uncertainty at various levels—ranging from model parameters to hyperparameters and the overall model structure. However, analytically solving the hierarchical GP model is highly challenging due to the complexity of the posterior distributions involved. As a result, various computational techniques have been developed to approximate or compute GPs efficiently, which make the use of GPs feasible for practical applications by overcoming the intractability of the full Bayesian approach. In this section, I will describe the hierarchical model for GP as well as some commonly used ways to compute GP models.

#### Hierarchical model

One of the primary challenges in implementing GPs involves determining the hyperparameters of the chosen kernel. Setting these parameters typically involves complex Bayesian inference within a hierarchical model with three levels: the posterior over parameters, the posterior over hyperparameters, and, at the top, the

posterior over the model structure. While defining the hierarchical model, we should acknowledge that: $w$ are the parameters of the model (often weights in linear models), $\theta$ are the hyperparameters of the model, $X$ is the input data (or features), $\mathcal{H}_i$ represents the model structure (defines how the data and parameters relate) and $y$ is the observed data. More on that can be found in (C. Rasmussen and Williams 2006).

The first level, the posterior over parameters, is described by:

$$p\left(w \mid y, X, \theta, \mathcal{H}_i\right) = \frac{p\left(y \mid X, w, \mathcal{H}_i\right) p\left(w \mid \theta, \mathcal{H}_i\right)}{p\left(y \mid X, \theta, \mathcal{H}_i\right)}, \tag{21}$$

where $p\left(y \mid X, \theta, \mathcal{H}_i\right)$ is the marginal likelihood, defined as:

$$p\left(y \mid X, \theta, \mathcal{H}_i\right) = \int p\left(y \mid X, w, \mathcal{H}_i\right) p\left(w \mid \theta, \mathcal{H}_i\right) dw. \tag{22}$$

The second level, the posterior over hyperparameters, is described by:

$$p\left(\theta \mid y, X, \mathcal{H}_i\right) = \frac{p\left(y \mid X, \theta, \mathcal{H}_i\right) p\left(\theta \mid \mathcal{H}_i\right)}{p\left(y \mid X, \mathcal{H}_i\right)}, \tag{23}$$

where the normalizing constant is:

$$p\left(y \mid X, \mathcal{H}_i\right) = \int p\left(y \mid X, \theta, \mathcal{H}_i\right) p\left(\theta \mid \mathcal{H}_i\right) d\theta. \tag{24}$$

The top level, the posterior over the model structure, is given by:

$$p\left(\mathcal{H}_i \mid y, X\right) = \frac{p\left(y \mid X, \mathcal{H}_i\right) p\left(\mathcal{H}_i\right)}{p(y \mid X)}. \tag{25}$$

This hierarchical Bayesian approach is challenging due to its complexity, requiring both a deep understanding of the problem and Bayesian inference expertise. Although the fully Bayesian approach can yield highly accurate models, especially with limited data, it is computationally demanding. The integrals involved in model may not be analytically tractable, necessitating approximations or advanced techniques, which can be computationally intensive.

### Maximum Likelihood Estimator

The most popular approach is the Maximum Likelihood Estimator (MLE). MLE comes in two variants: Type I and Type II. Type I optimizes the model parameters by maximizing the likelihood of the data, whereas Type II focuses on optimizing the model's hyperparameters by maximizing the marginal likelihood of the data, which allows for a better fit of the model to the data. In my research I mostly focus on Type II approach, referred as ML-II (Type II maximum likelihood method). ML-II simplifies the process by maximizing the marginal likelihood from equation (22), which is the likelihood of the observed data, integrated over the distribution of the GP functions. This method often employs approximations such as the Laplace approximation around the maximum of the marginal likelihood. This technique aims to optimize the model's hyperparameters in order to better fit model to the data. Moreover, while using this approach we have to be aware of the possibility of multiple local optima, so we need to use optimizers with subsequent runs while using different start values. Most applications are based on the algorithm presented for example by C. Rasmussen and Williams (2006). The input to this algorithm includes the data points $X$, the target values $y$ corresponding

to the input data, the covariance function $k$, the noise level $\sigma_n^2$, and the test input $x^*$ for which we wish to make predictions. The required steps are:

1. Compute the covariance matrix as $K = k(X, X) + \sigma_n^2 I$.
2. Perform the Cholesky decomposition of $K$ to get $L = \text{Cholesky}(K)$.
3. Solve the linear system to compute $\alpha$ as $\alpha = L^{-\top}(L^{-1}y)$.
4. Compute the predictive mean for the test input $x^*$ as $\bar{f}_* = k(x^*, X)\alpha$.
5. Compute the predictive variance as $v = L^{-1}k(X, x^*)$, and then compute $\text{Var}(f_*) = k(x^*, x^*) - v^\top v$.
6. Calculate the log marginal likelihood as:

$$\log p(y|X) = -\frac{1}{2}y^\top \alpha - \sum \log(L_{ii}) - \frac{n}{2}\log 2\pi$$

In algorithm the $\text{Cholesky}(K)$ is Cholesky decomposition, which is a matrix factorization technique used for symmetric, positive-definite matrices. Given a positive semi-definite matrix $K$, the Cholesky decomposition expresses $K$ as:

$$K = LL^T \tag{26}$$

where $L$ is a lower triangular matrix, meaning all the entries above the main diagonal are zero and $L^T$ is the transpose of $L$. The Cholesky decomposition is useful in numerical methods, such as solving linear systems, optimizing algorithms, and in GP regression for computing the inverse of the covariance matrix.

The output of the algorithm includes the predictive mean $\bar{f}_*$, the predictive variance $\text{Var}(f_*)$, and the log marginal likelihood $\log p(y|X)$. In general, while ML-II is less accurate compared to methods involving MCMC, it is easier to implement. It also requires less computational power, making it more suitable for practical applications, especially when dealing with limited computational resources.

### Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is advanced computational technique used for approximating complex probability distributions in Bayesian data analysis. As we can see in review by Margossian and Gelman (2023) MCMC is a strongly developing class of algorithms that generate samples from a target distribution by constructing a Markov chain, where the states of the chain represent possible values of the random variables. Specifically, MCMC enables the estimation of distribution characteristics, such as the expected value or variance, by sampling in the parameter space. The primary advantage of MCMC lies in its ability to handle high-dimensional spaces and complex dependency structures, making it a preferred tool in various applications, from machine learning to computational statistics.

Hamiltonian Monte Carlo (HMC) is an enhanced version of MCMC that incorporates concepts from Hamiltonian mechanics to explore the parameter space more efficiently. HMC is also the basis of framemork Stan, that enables flexible statistical modeling and Bayesian inference using advanced algorithms. Based on the description and implementation of this algorithm provided by Stan Development Team (2024) HMC simulates the dynamics of a physical system using Hamilton's equations, allowing for more effective movement through the sample space compared to standard MCMC methods. It is using derivatives of

the target density to guide transitions. It introduces momentum variables $\rho$, sampling from a joint density $p(\rho, \Theta) = p(\rho \mid \Theta)p(\Theta)$, where $\rho \sim \text{MultiNormal}(0, M)$ and $M$ is the mass (metric) matrix defined in (M. Betancourt 2017). The Hamiltonian is defined as:

$$H(\rho, \Theta) = T(\rho \mid \Theta) + V(\Theta), \tag{27}$$

where $T(\rho \mid \Theta) = -\log p(\rho \mid \Theta)$ (kinetic energy) and $V(\Theta) = -\log p(\Theta)$ (potential energy). The system evolves using Hamilton's equations:

$$\frac{d\Theta}{dt} = \frac{\partial H}{\partial \rho} \quad \frac{d\rho}{dt} = -\frac{\partial H}{\partial \Theta}. \tag{28}$$

Since HMC relies on the derivatives of the potential energy $V(\theta)$ to update both position and momentum, we need a way of efficient computation of gradients required for simulating the Hamiltonian dynamics. To solve this, we can use the autodiff (automatic differentiation), which is an essential tool for efficient computation of gradients in HHC. It automates the calculation of these gradients with high precision. This allows the algorithm to evolve the system through Hamilton's equations without manual derivative computation, ensuring both accuracy and computational efficiency. In frameworks like Stan, autodiff is seamlessly integrated, enabling HMC to efficiently navigate complex, high-dimensional posterior distributions. Then the system is numerically integrated using the leapfrog algorithm, which alternates updates to the momentum and position (parameters) in discrete time steps $\epsilon$. A transition between states is generated by simulating the joint dynamics of $\Theta$ and $\rho$, followed by a Metropolis acceptance step to correct for any numerical errors in the integration. The Metropolis step accepts the proposal $(\Theta^*, \rho^*)$ with probability $P_m$:

$$P_m = \min(1, \exp(H(\rho, \Theta) - H(\rho^*, \Theta^*))). \tag{29}$$

If the proposal is not accepted, the current state is retained for the next iteration.

As a result, HMC can better explore the distribution space, avoiding the slow convergence issues that often plague traditional MCMC approaches. As stated in (M. J. Betancourt and Girolami 2013) this method is particularly effective in high-dimensional parameter spaces, where classical MCMC methods may struggle with efficiency.

### Sparse approximation

Frequently, GP computation can be challenging due to the necessity of inverting matrices, which grow with the size of the dataset. Modern computers can handle reasonably sized GP models, but for large-scale problems, computation becomes a burden. To mitigate this, various approximation methods such as sparse methods are employed. These methods treat only a subset of latent variables exactly, while approximating the rest, reducing computational demands. Authors of (Bottou, Chapelle, DeCoste, and Weston 2007; Quiñonero-candela, C. E. Rasmussen, and Herbrich 2005) interpret this as exact inference with an approximate prior.

Specifically, when performing inference with a GP model, by placing a joint GP prior on the training data $f$ and test latent values $\tilde{f}$ and combining it with the likelihood $p(y|f)$, the joint posterior is given by:

$$p\left(f, \tilde{f} \mid y\right) = \frac{p\left(f, \tilde{f}\right) p(y \mid f)}{p(y)}. \tag{30}$$

To compute the posterior predictive distribution, we marginalize out the training set latent variables:

$$p\left(\tilde{f} \mid y\right) = \int p\left(f, \tilde{f} \mid y\right) \mathrm{d}f = \frac{1}{p(y)} \int p(y \mid f) p\left(f, \tilde{f}\right) \mathrm{d}f, \tag{31}$$

resulting in the posterior predictive distribution:

$$p\left(\tilde{f} \mid y\right) = \mathrm{Normal}\left(K_{*,f}\left(K_{f,f} + \sigma_n^2 I\right)^{-1} y, K_{*,*} - K_{*,f}\left(K_{f,f} + \sigma_n^2 I\right)^{-1} K_{f,*}\right). \tag{32}$$

However, this equation requires matrix inversion, which is computationally expensive for large datasets.

To reduce computational complexity, the joint prior $p(\tilde{f}, f)$ can be modified using inducing variables $u$, which are latent variables corresponding to a set of input locations (inducing inputs). The GP's consistency allows recovery of the joint prior $p(\tilde{f}, f)$ by marginalizing out $u$ from the joint GP prior $p(\tilde{f}, f, u)$:

$$p\left(\tilde{f}, f\right) = \int p\left(\tilde{f}, f, u\right) \mathrm{d}u = \int p\left(\tilde{f}, f \mid u\right) p(u) \mathrm{d}u, \quad \text{where} \quad p(u) = \mathrm{Normal}\left(0, K_{u,u}\right). \tag{33}$$

This joint prior can then be approximated by assuming that $f_*$ and $f$ are conditionally independent given $u$:

$$p\left(\tilde{f}, f\right) \simeq q\left(\tilde{f}, f\right) = \int q\left(\tilde{f} \mid u\right) q(f \mid u) p(u) \mathrm{d}u. \tag{34}$$

The choice of inducing variables and additional assumptions for equation (34) depend on the specific sparse algorithm used.

### Hilbert space approximation

Another approach to GP approximation is the Hilbert space approximate GP model, which provides a low-rank representation of stationary GPs. This method is based on s based on interpreting the covariance function as the kernel of a pseudo-differential operator (Shubin 1987) and approximating it using Hilbert space methods (Courant and Hilbert 1966) and was presented for instance in (Solin and Särkkä 2014) work. It uses basis function approximations, such as Laplace eigenfunctions for stationary covariance functions. The GP is approximated using a linear model via basis function expansion, significantly speeding up computation and inference. The linear structure simplifies approximation computation and facilitates the use of GPs as latent functions in non-Gaussian models, providing greater overall flexibility.

The Laplace eigenfunctions are independent of kernel choice and hyperparameters and can be computed analytically. The idea is to construct the covariance operator of a stationary covariance function as a pseudo-differential operator represented as a series of Laplace operators. This operator $\mathcal{K}$ over function $f(x)$ is formulated as:

$$\mathcal{K} f(x) = \int k\left(x, x'\right) f\left(x'\right) \mathrm{d}x', \tag{35}$$

where $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ is the covariance function. This operator is approximated using Hilbert space methods on a compact subset, subject to boundary conditions, leading to the approximation of a stationary covariance

function as a series expansion of eigenvalues and eigenfunctions of the Laplace operator. The resulting covariance is given by:

$$k\left(x, x'\right) = \sum_{j} S\left(\sqrt{\lambda_j}\right) \phi_j(x)\phi_j\left(x'\right), \tag{36}$$

where $S(\cdot)$ is the spectral density of the covariance function, $\lambda_j$ is the $j$th eigenvalue, and $\phi_j(\cdot)$ is the eigenfunction of the Laplace operator. Calculating this formula is relatively straightforward, as it involves evaluating the spectral density at the square roots of the eigenvalues and multiplying them with the eigenfunctions of the Laplace operator. This approximated GP is referred to as Hilbert Space Gaussian Process (HSGP). One of the significant works was done by researchers Riutort-Mayol, Bürkner, Andersen, Solin, and Vehtari (2020), which have developed a framework for working with HSGP, including integration with tools like Stan and MCMC sampling methods. They have also extended their methods to GPs with periodic covariance functions.

### Integrated Nested Laplace Approximation

A GP with a Matérn kernel can be interpreted as the solution to a corresponding partial differential equation. When modeling this equation using the finite element method, the GP can be approximated as a Gaussian Markov Random Fields (GMRFs). Specifically, according to the theory presented by Lindgren, Rue, and Lindström (2011), a GP with a Matérn kernel can be related to a GMRF through the solution of the equation:

$$(\kappa^2 - \Delta)^{\alpha/2} u(x) = W(x), \tag{37}$$

where $\Delta$ is the Laplace operator, $\kappa$ is the scale parameter, $\alpha$ controls the regularity of the process, and $W(\mathbf{x})$ represents spatial white noise. Solving this equation using the finite element method leads to a discrete representation in the form of a GMRF. This is important because GMRFs exhibit sparse covariance structures, which allow for efficient numerical implementation. If we already have a representation in the form of a GMRF, we can use the Integrated Nested Laplace Approximation (INLA) method, which is an efficient technique for approximating posteriors in the context of Bayesian modeling. INLA leverages the locality properties of GMRFs, allowing for a significant reduction in computational complexity compared to traditional Monte Carlo methods such as MCMC. In practice, this means that a GP with a Matérn kernel can be effectively modeled and approximated using INLA. That makes it an prominent alternative to traditional MCMC methods which can approximate Bayesian inference.

INLA, as described in (Rue and Held 2005; Rue, Martino, and Chopin 2009), is particularly suited to models that can be expressed as latent Gaussian models, due to their computational efficiency. In the INLA framework, the observed variables $y = (y_1, \ldots, y_N)$ are modeled using an exponential family distribution. The mean $\mu_i$ for each observation $y_i$ is linked to the linear predictor $\eta_i$ through a link function, where $\eta_i$ incorporates covariate terms and various types of random effects. For the model to be latent Gaussian effects $x$ are assumed to follow a GMRF with zero mean and a precision matrix $\mathbf{Q}(\theta)$ dependent on hyperparameters $\theta$. The likelihood for the observations, given the latent effects and hyperparameters, is:

$$\pi(y \mid x, \theta) = \prod_{i \in \mathcal{I}} \pi\left(y_i \mid \eta_i, \theta\right), \tag{38}$$

where $\eta_i$ is part of the vector $x$, containing all latent effects, and $\mathcal{I}$ includes the indices of all observed $y$ values.

INLA's primary objective is to estimate the posterior marginal distributions of the model's effects and hyperparameters. Model effects capture the underlying data structure and relationships, including fixed effects related to covariates and random effects accounting for spatial or temporal correlations. The joint posterior distribution of the effects and hyperparameters is given by Krainski, Gómez Rubio, Bakka, Lenzi, Castro-Camilo, Simpson, Lindgren, and Rue (2018):

$$\pi(x, \theta \mid y) \propto \pi(\theta)|Q(\theta)|^{1/2} \exp\left\{-\frac{1}{2}x^\top Q(\theta)x + \sum_{i\in\mathcal{I}} \log\left(\pi\left(y_i \mid x_i, \theta\right)\right)\right\}. \tag{39}$$

Here, $Q\left(\theta\right)$ denotes the precision matrix, and $|Q\left(\theta\right)|$ its determinant. The marginal distributions for the latent effects and hyperparameters are computed by integrating over the hyperparameter space, which requires a reliable approximation of the joint posterior distribution. This is achieved by approximating $\pi(\theta \mid y)$ as $\tilde{\pi}(\theta \mid y)$ and using it to approximate the posterior marginal distribution of the latent parameter $x_i$:

$$\tilde{\pi}\left(x_i \mid y\right) = \sum_k \tilde{\pi}\left(x_i \mid \theta_k, y\right) \times \tilde{\pi}\left(\theta_k \mid y\right) \times \Delta_k, \tag{40}$$

where $\Delta_k$ are weights corresponding to hyperparameter values $\theta_k$ on a grid.

INLA focuses on the distribution of the random variables, approximating the posterior distribution using Laplace approximation methods. This approach is computationally efficient but offers less direct insight into specific instances of the model's output.

### 1.2.5 Conclusion

In summary, a GP is a stochastic process utilized for modeling data observed over time, space, or both. It generalizes normal probability distributions, where each distribution characterizes a random variable (scalar or vector in the multivariate case). For the rest of the work we should focus on the definition where, given an index set $S$, a GP on $S$ is a collection of random variables indexed by a continuous variable:

$$\{f(t) : t \in S\} \tag{41}$$

This collection has the property that for any $n \in \mathbb{N}$ and any $t_1, \ldots, t_n \in S$, the marginal distribution of any finite subset:

$$\{f\left(t_1\right), f\left(t_2\right), \ldots, f\left(t_n\right)\} \tag{42}$$

is a multivariate Gaussian distribution.

As mentioned, each selected covariance function in a GP model has a set of free parameters called hyperparameters that need to be determined. Properly estimating these parameters is one of the main challenges in using GPs. Currently, the ML-II approach is widely adopted, where hyperparameters are optimized via marginal likelihood. This method is advantageous due to its simplicity and computational ease, but it can suffer from issues such as multiple local maxima, leading to less accurate results. Alternative approaches, such as MCMC and INLA, also exist for hyperparameter estimation and GP model creation but they need more attention.

The equation (7) is widely referenced across the literature, sometimes in different forms, but it mostly comes down to this formula. GPs have a wide range of applications, particularly in fields where machine learning and statistical methods are relevant. GPs are especially useful in challenging scenarios where data is difficult to obtain, and constructing a reliable model is challenging.

## 1.3 Theses and research problems

The goal of this dissertation is to provide a comprehensive solution for data diagnostics and fault detection with a focus on functional analysis as well as address the problem on insufficient data for modeling with use of generative GP. As a result, the following theses have been formulated:

**Thesis 1: Gaussian Processes can fill a gap in functional phenomena analysis and diagnostics.** GP models address a critical need in the field by offering the ability to model complex nonlinear dependencies that are difficult for traditional methods to handle. Their use of kernel functions allows them to capture relationships between data points in a flexible manner. Moreover, GP models are particularly advantageous due to their capacity for uncertainty quantification, enabling the user to not only predict outcomes but also to assess the confidence of these predictions. This feature is crucial in high-stakes applications where accurate diagnostics and the ability to handle uncertainty are paramount. Additionally, GPs models are providing interpretable results that demonstrate how input variables influence the outputs, which contrasts with the "black-box" nature of many modern machine learning models. By offering both predictive power and interpretability, GP models fill a gap in the current literature and practice of functional diagnostics, providing a valuable tool for real-time analysis and decision-making.

**Thesis 2: Generative Gaussian Processes models can be useful in the context of diagnostics with a lack of data.** In many diagnostic contexts, the availability of data is often insufficient or incomplete, limiting the effectiveness of traditional diagnostic models. Generative GP models can play a pivotal role in addressing this challenge by creating synthetic data that accurately reflects the functional properties of the system under investigation. These generative models allow for the augmentation of scarce datasets, improving the robustness and reliability of diagnostic algorithms. One of the key advantages of GP-based generative models is their ability to maintain the original functional relationships present in the data, ensuring that the synthetic data is both representative and reliable for training and testing purposes. This capability is particularly valuable in fields where collecting large amounts of real-world data is costly, time-consuming, or impractical. By filling the gaps in data availability, GP generative models provide a solution that enhances diagnostic processes, making it possible to perform accurate fault detection even in data-constrained environments. However, to fully leverage the potential of generative GP models, further research is needed to optimize their computational efficiency and ensure their applicability in large-scale industrial systems.

The formulated theses provide a foundation for understanding the potential of GPs in diagnostics and data augmentation, particularly in contexts where data is scarce or incomplete. However, while these models offer promising solutions, their practical application raises several critical challenges that need to be addressed. Specifically, the implementation of GPs in technical sciences and the efficiency of generative models demand further exploration.

In the following sections, six key research problems are defined, which solutions are able to support the theses. The main body of this doctoral thesis consists of the following papers, with each paper providing a solution to one of the defined research problems:

[1] A. Dudek and J. Baranowski. "Gaussian Processes for Signal Processing and Representation in Control Engineering". In: *Applied Sciences* 12.10 (2022).

[2] A. Dudek and J. Baranowski. "Efficient Gaussian Process Calculations Using Chebyshev Nodes and Fast Fourier Transform". In: *Electronics* 13.11 (2024).

[3] J. Baranowski, A. Dudek, and R. Mularczyk. "Transient Anomaly Detection Using Gaussian Process Depth Analysis". In: *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2021, pp. 221–226.

[4] A. Dudek, K. Jarzyna, and J. Baranowski. "Mixture Based Classifier Using Gaussian Processes for Induction Motor Diagnosis". In: *Proceedings of the IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*. 3-6 November, Chicago, USA, 2024.

[5] A. Dudek and J. Baranowski. "Modelling of Li-Ion battery state-of-health with Gaussian processes". In: *Archives of Electrical Engineering* vol. 72, No 3 (2023), pp. 643–659.

[6] A. Dudek and J. Baranowski. "Spatial Modeling of Air Pollution Using Data Fusion". In: *Electronics* 12.15 (Aug. 2023), p. 3353.

**Problem 1. Adoption of the GP in technical sciences**

GPs have become versatile tools in technical sciences, especially in control engineering and signal processing, because of their ability to model complex, non-linear relationships and handle uncertainty. However, despite their potential, the broad application of GPs is often limited by significant computational challenges, especially when applied to large datasets or in real-time scenarios. The problem, therefore, is understanding how GPs can be effectively applied across various technical domains, identifying both the advantages they offer and the obstacles they present. This includes examining the methods used to implement GPs, how they are adapted to different problems, and the ways in which these methods can be optimized for efficiency without compromising the quality of the results.

**Contribution and novelty [1]**

My contribution to this problem provides a comprehensive analysis of the wide-ranging applications of GPs in technical sciences, focusing on control engineering and signal processing. My work provides identifying and categorizing the different methods used to implement GPs, highlighting both their strengths and the challenges they pose, particularly in terms of computational demands. It also uncovers innovative approaches that have been developed to make GPs more computationally feasible, such as MCMC sampling, ML-II, sparse approximations and advanced kernel selection strategies. The key value of this contribution

lies in its ability to extract the essential elements of complex technical applications into clear insights, offering practical guidance and a state-of-art description, on how to leverage GPs more effectively across various domains, thus enhancing their applicability and efficiency in real-world technical scenarios.

**Problem 2. Efficiency of GP generative model computing**

GP models are powerful tools for modeling complex, non-linear relationships in data, but their application is often hindered by significant computational challenges. The complexity arises particularly in large datasets, where the need to compute and manipulate large covariance matrices can make GPs computationally expensive and time-consuming. This issue is intensified when GPs are used as generative models, which require the generation of numerous data points based on the model, further increasing the computational burden. The central problem is how to make GP computations more efficient while maintaining accuracy, especially in scenarios where the model needs to generate large amounts of data.

**Contribution and novelty [2]**

To address this problem I proposed solution, which introduces an innovative method to enhance the computational efficiency of GPs by leveraging Chebyshev nodes and the Fast Fourier Transform (FFT). The key contribution lies in reducing the computational domain by strategically selecting computation points based on the properties of Chebyshev nodes. This approach not only reduces the number of points required for computation but also significantly increases the speed of calculations. Additionally, by representing the results as a Chebyshev series, my method allows for the recovery of more detailed information about the original function or process. The Chebyshev series representation provides a more comprehensive view of the underlying data, capturing nuances that might be lost in more conventional approaches. This dual advantage of faster computations and enhanced information retrieval makes this method particularly valuable for improving the performance of GP-based generative models, making them more feasible for large-scale applications without sacrificing the quality of the results.

**Problem 3. Detecting transient anomalies in signals**

In process control systems, technical issues often manifest as time-localized events, such as single spikes or transient trends. These faults can be difficult to detect using traditional feature-based analysis methods, as these approaches typically involve averaging data over a time window, which can obscure or completely miss the transient anomalies. The core problem lies in the challenge of analyzing entire measured signals for anomalies, without relying on feature extraction that might weaken the impact of these localized events. This creates a critical need for methods that can effectively identify and analyze these brief but significant anomalies within the full context of the signal.

**Contribution and novelty [3]**

My proposed approach addresses the challenge of detecting transient, time-localized anomalies in process control systems by utilizing a combination of GPs and data depth analysis. Unlike traditional feature-based methods, my solution models the entire measured signal as a function of time, enabling the detection of

anomalies that might otherwise be averaged out or overlooked. By applying GPs to create a probabilistic model of both healthy and faulty transient states, and then using data depth functions based on generated data to use it in real world scenarios, the method provides a precise and reliable way to identify even subtle anomalies. This contribution is particularly valuable because it offers a robust tool for analyzing the full spectrum of the signal, ensuring that critical transient events are not missed, so it can enhance the overall reliability and effectiveness of fault detection in complex systems.

**Problem 4. Challenges in induction motor diagnostics**

Induction motor diagnostic is crucial for maintaining operational efficiency and preventing costly failures. Traditional diagnostic methods often struggle with noisy data and the complexity of detecting subtle faults, especially during motor startup when transient behaviors are difficult to analyze. Accurately identifying motor faults, such as broken rotor bars or other anomalies, requires a more sophisticated approach that can handle data uncertainty and provide reliable classifications under varying conditions. The key challenge is to develop a diagnostic method that can effectively process noisy, sparse, or limited data, ensuring accurate fault detection and classification. Moreover, each datapoint in such an analysis represents a functional profile over time, adding another layer of complexity in ensuring accurate diagnostics, thereby enabling proactive maintenance and reducing operational downtime.

## Contribution and novelty [4]

My proposed solution presents a novel approach to induction motor fault diagnostics by combining GPs with Gaussian Mixture Models (GMMs). The contribution lies in the integration of GPs as a Bayesian inference tool within the GMM framework, allowing for a more accurate classification of motor faults based on functional profiles and frequency characteristics. This method leverages the strengths of GPs in handling noisy and sparse data while enhancing the probabilistic classification capabilities of GMMs. By using synthetic data generation to augment real data, the approach not only improves computational efficiency but also provides a more robust diagnostic tool capable of clasification both healthy and faulty motors with higher precision. This combination is particularly effective in scenarios with limited or imbalanced data, making it a significant step forward in proactive motor maintenance and fault detection.

**Problem 5. Li-Ion battery diagnostics**

The degradation of lithium-ion batteries over time presents a significant challenge in various applications, as it leads to a reduction in battery capacity and an increase in internal resistance. Predicting the state of health (SoH) of these batteries is crucial for ensuring reliability, safety, and efficiency in their usage. However, this task is complex due to the multifaceted nature of battery degradation, which is influenced by numerous variables, including operating conditions and environmental factors. Traditional methods for SoH prediction often struggle to balance accuracy with computational feasibility, as they rely on either overly simplified models or highly complex electrochemical simulations. This creates a pressing need for more robust and adaptable predictive models that can accurately assess battery health under diverse conditions.

**Contribution and novelty [5]**

My research on the presented problem aimed to advance the ability to predict the SoH of lithium-ion batteries by applying GPs, coupled with both ML-II methods and MCMC sampling. This methodology brings a perspective to the problem where GPs, particularly when combined with MCMC sampling, offer a more detailed and probabilistically informed understanding of battery health. Moreover, by integrating a multi-kernel approach, this solution enhances prediction accuracy, allowing it to adapt more effectively to the varying conditions under which batteries operate. This contribution is also valuable because it addresses the shortcomings of traditional methods, providing a flexible and precise tool (with possibility of catching the uncertainty) that can be widely applied across different battery systems to improve their reliability and longevity.

**Problem 6. Spatial modeling of air pollution**

Air pollution is a significant environmental and health concern, characterized by its variability across different locations and times. Traditional air quality monitoring systems, which typically rely on a limited number of professional monitoring stations, often struggle to provide a comprehensive understanding of pollution levels due to their sparse coverage. This limitation is especially problematic because air pollutants can exhibit complex, spatially-dependent, functional behaviors, changing rapidly in response to environmental factors. The central challenge is how to accurately model air pollution across a broad area, accounting for these dynamic and localized changes, with the data constraints imposed by the limited number of monitoring stations.

**Contribution and novelty [6]**

My approach to the problem firstly enhances air pollution modeling by data fusion, especially integrating data from multiple sources, including low-cost sensors and traditional monitoring stations. Then by employing the INLA method within the R-INLA framework, the model is able to effectively capture the complex and variable nature of air pollution across different scenarios. This integration allows for the creation of more detailed and accurate pollution maps, even in areas where data from professional stations are sparse. The novelty of this solution lies in its ability to combine diverse data types, improving the precision and reliability of pollution predictions but also by using the GP methods of complex spatial of air pollution. This approach not only fills gaps left by traditional monitoring methods but also provides a more robust tool for environmental monitoring and public health protection.

In the next chapter, I will present expanded summaries of the articles to highlight their respective areas of focus.

# Chapter 2

# Detailed overview of conducted research

This chapter provides a summary of the research papers conducted to address the key problems identified in support of the proposed theses. The studies reviewed here are closely linked to the research questions, offering insights and findings that contribute to validating the theses. By examining relevant articles and experimental results, this section highlights the most important methodologies employed, the challenges encountered, and the solutions proposed, thereby laying the groundwork for the conclusions drawn in this doctoral thesis. Appendix A includes the full texts of all the referenced papers, and when described, appropriate page of this thesis is provided for the purpose of clarity. Citation in this chapter is harmonized with the one from relevant papers and not chapter 1 of this thesis.

## 2.1 Gaussian Processes in technical sciences

Concluding research detailed in article [1] *"Gaussian Processes for Signal Processing and Representation in Control Engineering"* (see page 43) we confirmed out that GPs are a powerful tool in the field of technical sciences, especially in control engineering and signal processing. There are primarily used in regression, classification, and optimization tasks. My research provided an overview of key applications of GPs and highlights areas where their use is highly relevant as well as methods popularly used across applications.

GPs are widely used in control engineering for system identification, fault detection, and predictive control, effectively handling uncertainty in system dynamics. In signal processing, they excel at modeling time series and extracting patterns from noisy data. Especially in cases with sparse or irregular measurements GP can outperform traditional methods due to their robustness to missing or noisy data.

### Main Technical Applications

Several key areas in technical sciences benefit significantly from the application of GPs due to their flexibility, ability to model complex non-linear relationships, and their probabilistic nature, which allows for handling uncertainty. In figure 2.1 we can see diagram of highlated areas for GP applications. Below are some of the most prominent examples in the field where GPs have demonstrated their effectiveness:

GP find extensive applications in regression tasks, where modeling complex, nonlinear dependencies is necessary. An example is battery health monitoring, particularly in the context of electric vehicles, consumer electronics, and renewable energy storage systems. As noted in (Garay, Huaman, and Vargas-Machuca

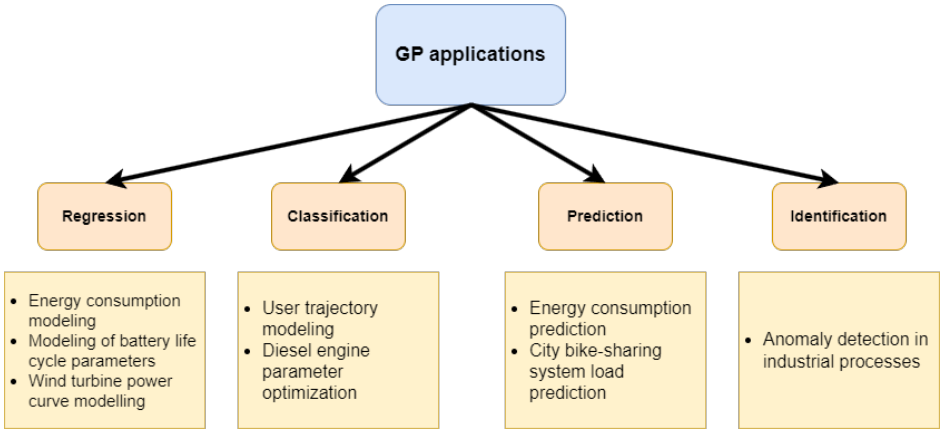



Figure 2.1: The diagram highlights GP applications in four areas: regression (e.g., energy and battery modeling), classification (user trajectories, engine optimization), prediction (energy use, system load), and identification (anomaly detection in industrial processes).

2021b; Tagade, Hariharan, Ramachandran, Khandelwal, Naha, Kolake, and Han 2020), GPs enable modeling the degradation processes to predict the remaining useful life (RUL) and state of health (SoH) of batteries. Their ability to capture the complexity and non-linearity of degradation processes offers more accurate predictions compared to traditional methods, allowing for the optimization of maintenance schedules, improvement of reliability, and reduction of operational costs.

In the context of prediction, GPs demonstrate their effectiveness in forecasting energy consumption, particularly in smart grids and building energy management systems. The authors Zeng, Ho, and Yu (2020) employed GPs to model and predict electricity consumption based on historical usage patterns and external factors, such as weather conditions or building occupancy rates. Accurate energy forecasts are critical for optimizing energy distribution, reducing waste, and implementing low-energy strategies. The ability of GPs to model uncertainty is especially beneficial in this context, as it enables probabilistic forecasting, which supports better decision-making in energy supply and demand management.

GPs are applied in classification tasks where predicting uncertainty is crucial, such as in diesel engine modeling related to post-injection processes done by Gönül, Kutlar, Calik, and Orcun Parlak (2021). The goal is to optimize engine parameters to minimize oil dilution, which can occur due to factors like frequent DPF regeneration, catalyst wear, and cold starts. GPs model the relationships between engine settings and oil dilution rates, effectively capturing complex interactions and providing uncertainty estimates. This approach has proven more accurate than polynomial regression, aiding in optimal parameter selection and risk management for diesel engines.

GPs play a significant role in identification, such as in anomaly detection, where they are used to identify and predict faults or unusual behavior in industrial processes. As described by Wang and Mao (2019), this is particularly important in sectors where continuous operation is critical, such as manufacturing, power generation, and chemical processing. By modeling the expected behavior of a system and identifying deviations from the norm, GPs enable early fault detection, preventing system failures and costly downtimes. The probabilistic nature of GPs allows not only for the detection of faults but also for estimating the confidence in these detections, which is essential for decision-making in high-risk environments.

### Relevance to technical sciences

GPs are highly relevant to control engineering, automation, and signal processing due to their ability to manage uncertainty, model non-linear dynamics, and support data-driven approaches. In control systems, uncertainties in model parameters and external disturbances are common. GPs provide a probabilistic framework that naturally incorporates these uncertainties, offering not only predictions but also confidence intervals, which improve decision-making in critical systems such as autonomous vehicles or industrial automation.

GPs are also effective in modeling complex, non-linear relationships that traditional linear models struggle with. This makes them invaluable in fields like robotics, aerospace, and process control, where non-linear dynamics are prevalent. Their ability to learn from data without needing a predefined model allows GPs to excel in system identification and adaptive control, adjusting to changing system conditions in real-time.

In the area of fault detection and diagnostics, GPs can model expected system behavior and identify deviations, enabling early detection of potential failures. This is crucial in industries such as manufacturing and energy, where undetected faults can lead to costly downtime or safety hazards.

Finally, GPs enable adaptive control strategies, where the system evolves based on real-time data. This is particularly useful in dynamic environments, such as renewable energy systems and robotics, where conditions change frequently.

In summary, GPs offer robust tools for uncertainty management, non-linear modeling, and adaptive control, making them essential in advancing modern control engineering and automation systems.

### Summary and Conclusions

The increasing computational power available today has made GPs a practical tool for a wide range of applications in technical sciences. Their flexibility, combined with the ability to model uncertainty, makes them particularly valuable in areas such as control engineering, signal processing, and optimization.

My future research will also focus on exploring the potential of generative GP models, particularly in scenarios involving missing or limited datasets, where traditional approaches may fall short. Additionally, I discovered that there is a need to investigate alternative computational techniques for GPs that are not yet widely utilized but could offer significant improvements in performance and scalability. Further analysis of less-explored approaches to GPs, along with the development and proposal of alternative optimization algorithms, would help to improve the efficiency of GP applications, making them more practical and effective in solving complex real-world problems.

## 2.2 Efficient Gaussian Process calculations

Our work described in article [2] *"Efficient Gaussian Process Calculations Using Chebyshev Nodes and Fast Fourier Transform"* (see page 68) investigates the development of a novel approach to GP modeling that significantly reduces computational complexity and retains information about modeled function by utilizing Chebyshev nodes and FFT. Traditional GP models, while powerful in capturing complex, non-linear relationships, are often computationally expensive, especially with large datasets. Our study proposes a method that leverages the properties of Chebyshev polynomials to optimize the placement of computational

nodes, thereby minimizing the degree of the Chebyshev series as well as using the proprieties of Chebyshev series enhancing the efficiency of GP calculations.

At the core of our approach is the use of Chebyshev nodes, which are strategically selected points that enable efficient function interpolation. The values at these nodes are transformed into Chebyshev series coefficients using FFT. The Chebyshev series, known for its convergence properties, especially towards differentiable functions, provide a robust framework for approximating functions with minimal loss of accuracy. The application of FFT in this context allows for the rapid computation of these coefficients.

The primary contribution of this study lies in the innovative combination of Chebyshev nodes and FFT for GP calculations. This method not only reduces the dimensionality of the problem but also enhances the performance of GP as a generative model. By focusing on functional series representations, the approach ensures that the generative model maintains high accuracy with a lower computational cost.

## Algorithm

We proposed methodology, which centers around the generation of functions through a GP model, represented as a functional series specifically, the Chebyshev series. The goal is to efficiently transform points generated by the GP into a Chebyshev series, which involves determining the coefficients of the series. This is not a trivial task, as each polynomial can be uniquely expressed as a finite Chebyshev series. The critical insight lies in the unique linear relationship between the function values at Chebyshev nodes and the coefficients of the Chebyshev expansion. This relationship allows for an efficient mapping of values to coefficients, executable in $O(n \log n)$ operations using the FFT.

In the context of generative modeling, the use of the Chebyshev series is particularly advantageous due to its strong convergence properties, especially for differentiable functions generated by the model. As stated in (Trefethen 2012) the convergence of the Chebyshev series to the function generated by the GP model can be mathematically described by:

$$\|f - f_n\| \leq \frac{2V}{\pi\nu(n-\nu)^\nu}, \quad \text{for integer } \nu \geq 1, \tag{1}$$

where $f$ and its derivatives up to $f^{(\nu-1)}$ are of bounded variation $V$. For any $n > \nu$, this inequality illustrates the rate at which the Chebyshev series converges to the true function.

Given the computational demands of directly calculating the Chebyshev series, the Chebyshev interpolant is often employed. The interpolant closely resembles the series and provides a similar convergence behavior, represented as:

$$\|f - p_n\| \leq \frac{4V}{\pi\nu(n-\nu)^\nu}. \tag{2}$$

The minimal difference in accuracy between the series and its interpolant, amounting to only one bit of precision, underscores the efficiency and numerical stability of the method. This is critical in many engineering and scientific applications.

An important aspect of the methodology involves obtaining the Chebyshev coefficients, which is crucial for the effectiveness of the approach. Leveraging the work of Ahmed and Fisher (1968), it is established that the use of interpolation formulas combined with FFT on the values of Chebyshev polynomials can accurately determine these coefficients with minimal loss of precision.

Another crucial thing is to know where to apply FFT based algorithms to obtain Chebyshev coefficients in the optimal way. The idea is to calculate the GP at certain selected points based on Chebyshev nodes and polynomial properties, and then apply the rest of the algorithm to the values obtained from the GP model. These selected points are crucial in reducing the degree of the polynomial required to achieve accurate interpolation, thereby enhancing the computational efficiency of the method, because this selection directly influence the order of Chebyshev series.

The approach is particularly powerful in the realm of generative models, where it is used to generate new functions with high accuracy and efficiency. The rapid convergence of the Chebyshev series, combined with FFT's computational efficiency, makes this method ideal for generating high-quality samples in machine learning and simulations.

Figure 2.2 illustrates the overall process, starting from the selection of Chebyshev nodes, through function evaluation, to the application of FFT for obtaining the Chebyshev coefficients, and finally the construction of the GP model.

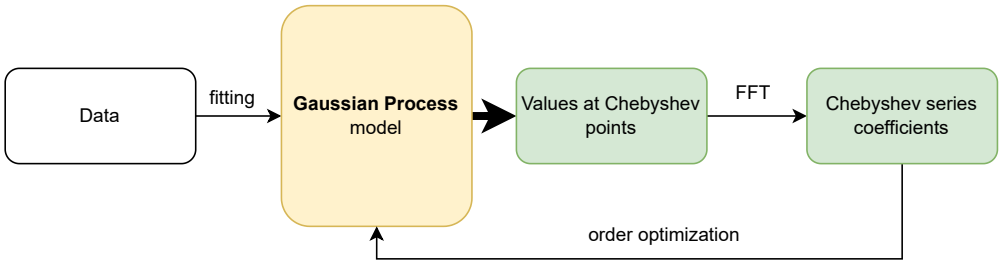



Figure 2.2: Flowchart of the algorithm for efficient GP modeling using Chebyshev nodes and FFT.

This methodology ensures that the GP model is computationally efficient while maintaining the accuracy necessary for practical applications in generative modeling.

### Example Application of the Algorithm

In our study, we provided an example application of the proposed algorithm. Method was applied to two types of functions: one that is at least twice differentiable and another that is infinitely differentiable, using both MCMC sampling and the INLA framework. Here we will only focus on a fraction of experiment in order to give a context of practical application.

The experiment involved generating a small number of noisy samples from the selected functions, followed by fitting a GP model to these samples. The functions were: $f(x) = |\sin(x)|^3$ and $f(x) = e^x + e^{x^2}$ for Matérn and RBF kernel respectively. The GP model was then used to generate values at the Chebyshev nodes, which were subsequently transformed into Chebyshev series coefficients using FFT. The degree of the Chebyshev series was optimized by evaluating the behavior of the coefficients, ensuring that the series accurately represented the original function with minimal computational cost.

Example in Figure 2.3 first row shows the function $f(x) = e^x + e^{x^2}$ for RBF kernel with niosy samples generated from it, as well as example usage of algorithm in second row. Example of usage shows in the left column the fitted mean values and 95% confidence intervals for the model using a RBF kernel. Right column displays the corresponding Chebyshev coefficient values for MCMC case, with the red dashed line

indicating the selected threshold for optimization. This example shows that optimized order of Chebyshev series should be around 25. Based on that we could use only 25 nodes for GP model in order to retrieve enough information for GP generative model. This should lead to significant computations time decrease without scarification accuracy of GP model.

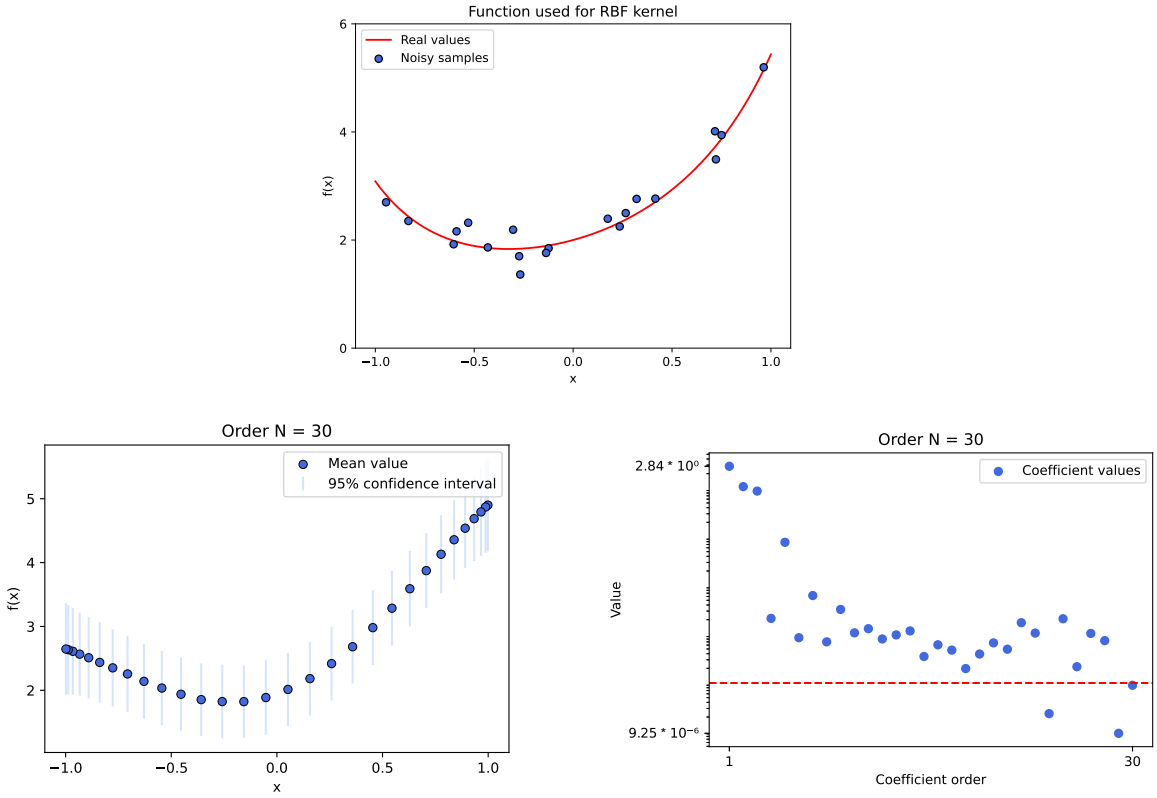



Figure 2.3: Top: Function $f(x) = e^x + e^{x^2}$ with noisy samples for the RBF kernel fit. Bottom left: GP model calculations with 95% confidence interval for order $N = 30$. Bottom right: Chebyshev coefficient values for the GP model, with a red dashed line marking the optimization threshold.

## Summary and Conclusions

Conclusion of this study presents a significant advancement in the field of GP modeling by introducing an efficient computational approach that leverages Chebyshev nodes and FFT. The primary theoretical contribution is the innovative use of these tools to optimize the placement of computational nodes, thereby minimizing the computational complexity of GP models. The example application demonstrates the practical benefits of this approach, particularly in scenarios requiring heavy computational resources. The proposed algorithm successfully reduces computation time while maintaining high accuracy, making it a valuable tool for researchers and practitioners working with GP models. Future work will explore the extension of this approach to more complex, multi-dimensional problems, further enhancing its applicability in various fields of study.

## 2.3 Transient anomaly detection

In recent years, the monitoring and diagnostics of process control installations have gained significant importance due to their critical role in the global economy. Fault detection within these systems is particularly challenging, especially during transient states, where most existing methods struggle. Here, I summarize the key contributions of work [3] *"Transient anomaly detection using Gaussian Process depth Analysis"* (see page 95) in developing a novel method for detecting anomalies during transient states using a combination of GP and data depth functions.

### Introduction to the Problem

The complexity of process control installations, often characterized by nonlinearity and susceptibility to stochastic disturbances, makes traditional first-principles modeling approaches impractical. As a result, the research has shifted towards statistical models and machine learning methods. However, these data-driven models often produce "black-box" solutions that are difficult to interpret, particularly when dealing with rare or incomplete fault data.

The focus of our work has been on addressing these challenges by leveraging GP for modeling transient states and analyzing their depth using data depth functions. The novelty lies in the combination of these techniques to detect anomalies, which was previously difficult due to the limitations of existing models that primarily handled steady states.

### Model and Transient State Anomaly Detection

We used a GP to model transient states in a control system, trained on data from a water tank in both healthy and faulty conditions. By optimizing the Rational Quadratic kernel's hyperparameters with the LM-BFGS algorithm, the GP accurately modeled normal and faulty behavior. The GP generated synthetic data, expanding the dataset and enabling data depth analysis to distinguish normal from anomalous states.

Data depth analysis helped to separate healthy from faulty states using Euclidean and Mahalanobis depths presented by Tukey (1975). Euclidean depth identified outliers by measuring the squared distance from the mean, while Mahalanobis depth accounted for data covariance, providing better fault detection. Histograms of GP-generated trajectories showed that healthy data had higher depth values, effectively distinguishing anomalies.

### Experiment Setup

We tested the method on a system consisting of three hydraulic tanks connected in a cascade, as shown in Figure 2.4. The system was equipped with sensors to measure water levels, and the entire setup was controlled via a computer interface. The experiments involved generating both healthy and faulty data by manipulating the flow of water through the tanks using manual and electromagnetic valves.

Figure 2.4: Photo of a system of three water tanks connected in a cascade. The entire system is connected to a computer that controls the pump and solenoid valves.

### Results

The results demonstrate the effectiveness of the GP-based method in accurately modeling the healthy state and detecting deviations indicative of faults. The GP model effectively generated enough data for both cases in order to use the data depth algorithms. The model's accuracy is confirmed by the fit between sampled data and the predicted mean with confidence intervals. Depth histogram highlights a clear distinction between healthy and faulty states. Healthy data clusters around higher depth values, while faulty data shows much lower values, confirming the model's efficacy in detecting anomalies.

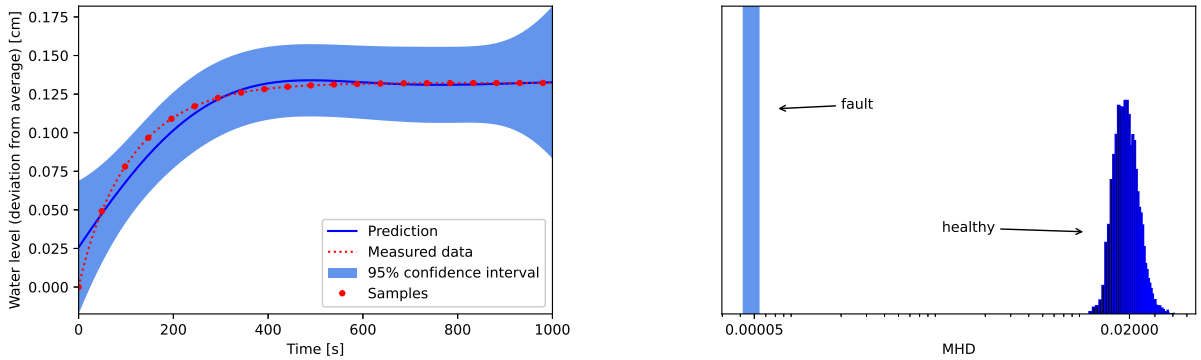



Figure 2.5: On left: GP model prediction for healthy state with measured data and 95% confidence interval. On right: Mahalanobis depth histogram showing clear separation between healthy and faulty states, with faulty data clustered at lower depth values.

Example result is shown in the Figure 2.5. On the left we can see GP model, optimized using healthy data, produced predicted trajectories closely aligned with actual measurements. The blue line represents the

predicted mean trajectory, with the shaded area indicating the confidence interval. The red dots, which are the sampled data points, fit well within this model. The left side shows the Mahalanobis depth analysis, which clearly separates healthy data, which clusters around higher depth values, from faulty data, which shows much lower values, marking them as outliers.

### Conclusion

Conclusion of this study presents method of using GPs in combination with data depth functions shows significant promise for transient anomaly detection in process control systems. While the results are encouraging, further research is possible to explore different depth functions and refine the GP models to improve their robustness and applicability in various industrial settings.

## 2.4 Diagnostics with mixture based classifier

Induction motors are critical components in various industrial applications, and their reliable operation is paramount. Diagnosing faults in these motors can prevent unexpected breakdowns and reduce downtime. Traditional diagnostic methods often struggle with the noisy data and complex relationships inherent in motor operation. To address these challenges, our study outlined in article [4] *"Mixture Based Classifier Using Gaussian Processes for Induction Motor Diagnosis"* (see page 101) leverages statistical techniques, particularly probabilistic classification using GMMs combined with Bayesian inference through GPs. The integration of these methods allows for the modeling of complex, non-linear relationships and provides robust classification capabilities even in noisy environments.

### Considered System

Our study focused on diagnosing induction motors during startup, where variations in motor behavior are most pronounced, which can be seen in Figure 2.6. Four identical induction motors were used, each subjected to different degrees of damage, including single and double broken rotor bars. The data collected from these motors, under varying voltage conditions, were analyzed using spectrograms to identify patterns indicative of motor health.


Figure 2.6: Setup of four motors used for data collection. One of them was healthy, and the rest was damaged in different ways: broken rotor bar, two rotor bars broken, broken end ring. In this analysis, we focused on the broken rod diagnostics.

Spectrogram analysis revealed distinct differences in the lower frequency bands between healthy and damaged motors, which were further analyzed using Fourier transforms. A notch filter was applied to isolate relevant features and mitigate interference, particularly the 50 Hz mains component.

Considered system posed a problem of a small dataset. To overcome this challenge we employed a generative modeling approach using synthetic data. The synthetic data was generated based on a B-spline model, allowing the augmentation of the dataset and improving the classifier's performance. This approach ensured that the model could handle the variability in the data, reducing the risk of overfitting.

## Methods

To solve the problem we employed GMMs, which are probabilistic models that assume the data is generated from a mixture of several Gaussian distributions. In this study, each component of the GMM is modeled as a GP, allowing the classification of motor states into healthy or faulty categories. The GMM classifier assigns probabilities to each possible outcome, making it a robust tool for handling data where the boundaries between classes are not distinct.

We also utilized the Stan framework to implement the GMM classifier. As stated by Stan Development Team (2024) Stan is a probabilistic programming language designed for Bayesian inference, which uses HMC algorithm for efficient sampling. It's flexibility allowed for the integration of GPs within the GMM framework, facilitating the complex calculations required for the model's development. The use of Stan provided us several advantages. First, it allowed for a fully Bayesian treatment of the GMM, enabling the incorporation of prior knowledge and the ability to handle parameter uncertainty naturally. Second, the advanced sampling methods in Stan ensured efficient exploration of the posterior distribution, even in the high-dimensional parameter space of the GMM.

## Results

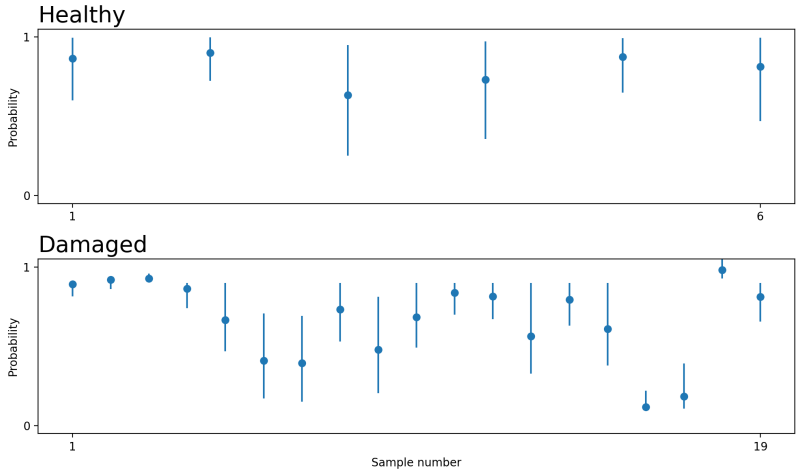



Figure 2.7: The plot displays probabilistic classifications for a set of samples categorized into two classes: healthy and damaged. Plots show the probability distribution of each sample belonging to its respective class across multiple tests. The blue dot represents the mean values, while the blue bar presents the 95% confidence interval

We tested the GMM classifier, combined with GP, on both synthetic and real-world data. The model was able to accurately classify the motor states, distinguishing between healthy and faulty conditions with high confidence. Our study presented the results as probabilistic classifications, emphasizing the importance of interpreting the outputs in terms of distributions rather than single-point estimates. The results can bee seen in Figure 2.7

### Summary and Conclusions

Conclusion of this study presents the potential of using a mixture model with GPs for the classification of faults in induction motors. The probabilistic nature of the model allows for robust classification even in the presence of uncertainty and noisy data. However, the study also identified areas for improvement, including computational efficiency and the need for a systematic approach to determining sample sizes for GPs. Our future work will focus on addressing these challenges and further enhancing the model's accuracy and efficiency. This includes exploring approximations for GPs, increasing the number of categories in the mixture model, and applying information criteria to improve modeling representation.

## 2.5 Li-Ion battery diagnostics and modelling

In this part, I review the conclusions drawn in the article [5] *"Modelling of Li-Ion battery state-of-health with Gaussian processes"* (see page 107), which main idea is, as the name suggests, to develop and research GP modeling methods of Li-Ion battery SoH. The focus of this work has been on refining methods for accurately predicting the SoH of Li-Ion batteries, a critical aspect of ensuring the reliability and longevity of battery-powered devices as well as expand awareness and test the possibilities of different variants of GP modeling approaches in diagnostics.

### Li-Ion Battery SoH Modelling

Lithium-ion batteries are widely used in portable consumer electronic as well as industrial devices due to their high energy density and long cycle life. However, based on work by, for instance Garay, Huaman, and Vargas-Machuca (2021a), the performance of these batteries degrades over time, leading to a reduction in capacity and an increase in internal resistance. The SoH of a battery is a key indicator of its remaining useful life, typically defined as the ratio of the current capacity to the initial capacity. Accurate prediction of SoH is crucial for the effective management and maintenance of battery systems.

Overall, GPs have been employed as a non-parametric, probabilistic approach for modeling the SoH of Li-Ion batteries. Unlike traditional parametric models, GPs provide a flexible framework that can model complex, non-linear relationships without assuming a specific functional form. In research, we have utilized GPs in two primary approaches: the ML-II method and the MCMC sampling method.

### Dataset

The dataset used in our study originates from the NASA AMES Prognostics Data Repository by Bole, Kulkarni, and Daigle (2014). It consists of randomized battery usage data for a set of 28 lithium-ion 18650

cells. These cells were categorized into seven different groups based on their charging and discharging conditions. The details of each group can be found in full text of article.

Each cycle in the dataset ends with a check of the battery's remaining capacity. The extracted features for each cycle include:

- Total time of battery life.
- Time of the selected cycle.
- Charge throughput during the cycle (integral of current over time).

This dataset provided us comprehensive resource for analyzing the SoH and degradation of lithium-ion batteries under various operating conditions.

### SoH Modelling

Our study consisted of two apporaches to the GP modelling problem. One based on ML-II alghoritm and second one based on MCMC sampling method. Each approch focused on predicting the changes in battery capacity in strict intervals based on the provided datasets. Then these values could be converted into prediction of SoH (capacity in certain time of life) values.

The ML-II approach involves optimizing the hyperparameters of the GP by maximizing the marginal likelihood of the observed data. This method is computationally efficient and relatively straightforward to implement. We applied the ML-II approach using various kernel functions, including the RBF, Matérn, Rational Quadratic (RQ), and Exp-Sine-Squared (ESS) kernels. The results demonstrated that the multi-kernel approach, particularly the combination of Matérn and RQ kernels, provided the best accuracy in predicting SoH, with a Root-Mean-Square Error (RMSE).

The MCMC sampling method, in contrast to ML-II, provides a more comprehensive probabilistic framework by generating samples from the posterior distribution of the model parameters. This approach, while computationally intensive, offers deeper insights into the uncertainty of the predictions. We implemented this approach using the Stan probabilistic programming language, with models built from scratch to account for expert knowledge in prior distributions. The RMSE values obtained using MCMC were slightly higher than those from the ML-II approach, but the method provided more consistent confidence intervals across predictions.

Both the ML-II and MCMC approaches have their advantages and limitations. The ML-II approach is faster and easier to implement, making it suitable for real-time applications. However, it may suffer from overfitting and can sometimes struggle with capturing the full uncertainty of predictions. On the other hand, the MCMC approach, while more computationally demanding, provides a richer understanding of the model's uncertainty, which is crucial for applications requiring high reliability. In Figure 2.8 we can observe the prediction of the same tested battery degradation as well as confidence interval for each method.

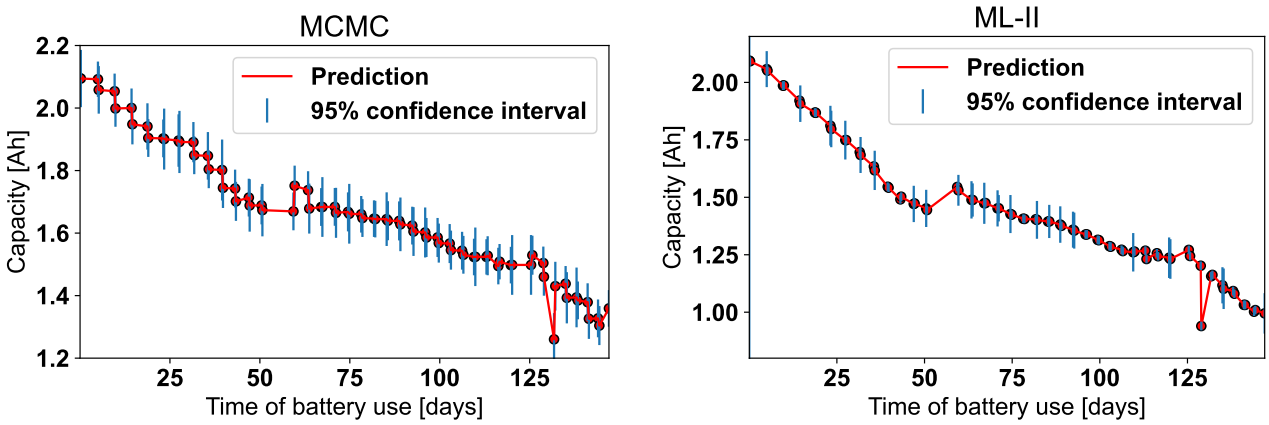



Figure 2.8: Comparison of SoH predictions using ML-II and MCMC approaches. Red dots show predicted values, and blue lines represent the 95% confidence interval.

### Summary and Conclusions

Conclusion of this study presents the effectiveness of GP in modeling the SoH of Li-Ion batteries. The ML-II approach provides a practical balance between accuracy and computational efficiency, while the MCMC approach offers deeper insights into the uncertainty of predictions. Future work can focus on refining the MCMC models to improve their accuracy and exploring the integration of these methods into real-time battery management systems.

## 2.6 Spatial modelling of air pollution

Our research [6] "Spatial modeling of air pollution using data fusion" (see page 124) addressed the challenge of accurately modeling spatial air pollution in specific locations, which is complicated by the necessity of integrating data from multiple sources, including professional monitoring stations and amateur sensors. Traditional approaches such as MCMC methods, while effective, are computationally intensive and less practical for large-scale spatial models. The core research problem addressed in this study is the development of an efficient method for spatial modeling that can yield precise predictions of air pollution levels by leveraging data fusion from diverse sources.

### Methodology and Approach

Our study employs the INLA as the central method for spatial modeling of air pollution. INLA is particularly advantageous due to its computational efficiency. It combines the Laplace approximation with numerical integration, enabling fast and accurate Bayesian inference, which is essential for handling large and diverse datasets typical in environmental studies.

We focused on applying INLA to model air pollution in both industrial environments and urban areas. In the industrial setting, the methodology involves simulating pollution dispersion based on bivariate normal distributions, reflecting the spatial distribution of pollutants within the selected facility. Additionally, a barrier model is employed to account for physical obstacles, such as walls, which significantly impact the

dispersion and concentration of pollutants in separated spaces. This approach allows for the detailed modeling of pollutant concentrations, their spatial gradients, and the influence of structural barriers within the facility.

In urban environments, the methodology was expanded to incorporate real-world data from the city of Kraków, Poland. The data used in this study are sourced from various sensors, including government-operated air quality monitoring stations from (*Portal Jakość Powietrza GIOŚ. Bank danych pomiarowych* 2023) and a network of amateur sensors from (*Airly.org* 2023). The integration of these heterogeneous data sources is achieved through advanced data fusion techniques, which are crucial for enhancing the accuracy of the spatial models. Data fusion allows for the combination of different types of measurements, ensuring that even areas with sparse official monitoring can be accurately represented in the pollution maps.

## Experimental Results

Our experimental results from the three models provide valuable insights into the effectiveness of the proposed approach:

- **Industrial Room Model:** Pollutant concentrations were highest near emission sources and decreased with distance. The spatial model, based on a refined Delaunay triangulation, accurately captured the spatial variability of pollutants across the facility, effectively highlighting areas with high pollution levels.
- **Industrial Room Barrier Model:** Including physical barriers (such as walls) in the model significantly improved prediction accuracy. The barriers restricted pollutant flow, resulting in lower concentrations in isolated rooms and higher levels near pollution sources, providing a more realistic simulation of air quality. Graphical representation of described result can be seen in Figure 2.9.

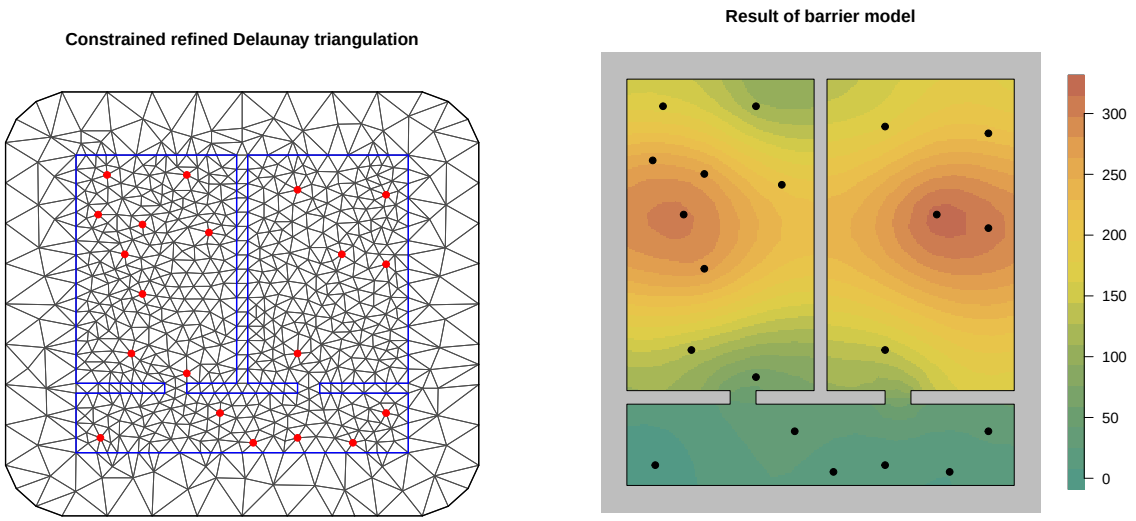



Figure 2.9: The left image shows a constrained refined Delaunay triangulation applied to an industrial room with barriers (blue lines), while the right image presents the corresponding pollutant distribution model. Higher pollutant concentrations are displayed in red near emission sources, with lower concentrations in green.

- **Kraków Air Pollution Model:** The urban model mapped pollution levels across Kraków, with higher concentrations observed in western and central regions. Data fusion from various sensors enhanced accuracy, especially in areas with sparse official data, yielding a more reliable pollution distribution across the city. Graphical representation of described result can be seen in Figure 2.10.

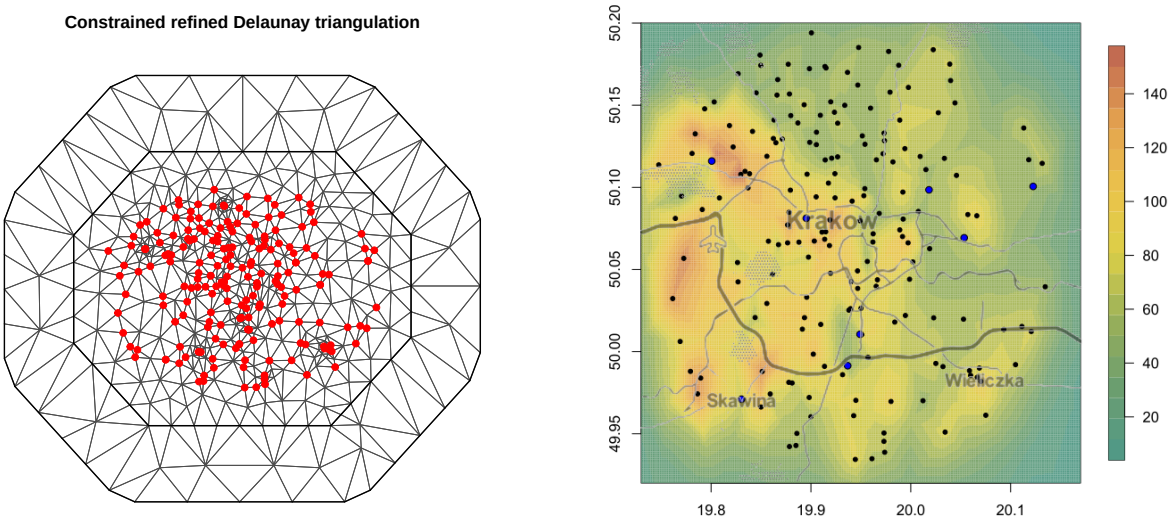



Figure 2.10: The left image illustrates the triangulation of an urban area (Kraków), where red points mark the locations of pollution sensors. The right image shows the spatial distribution of air pollution in the city, with higher concentrations in red and orange, particularly in the western and central areas.

## Summary and Conclusions

Conclusion of this study presents the effectiveness of using INLA in conjunction with data fusion techniques for spatial modeling of air pollution. The models we developed offer a more accurate representation of pollution distribution, especially in cases where data is fragmented or incomplete. The barrier model, in particular, proved to be crucial for modeling indoor environments where physical obstructions significantly influence the spread of pollutants. Our findings suggest that INLA method is highly suitable for spatial air pollution modeling, providing a robust framework for integrating heterogeneous data sources. Future work will extend these models to three-dimensional spaces and incorporate additional environmental factors such as wind patterns, further enhancing the predictive accuracy and practical application of the models.

## 2.7 Summary

These were summaries of my research, aimed at presenting the key conclusions and the most important achievements acquired from the conducted analyses. A full and detailed description of all the research stages, the methods employed, as well as the detailed results, can be found in the attachment. There, you will find all the data, charts, analyses, and discussions that thoroughly illustrate the entire research process and provide a better understanding of the context and significance of the obtained results. I encourage you to review the full document, which will provide a more comprehensive and exhaustive understanding of the research undertaken. **Full text of each paper can be found in appendix A.**

# Chapter 3

# Concluding Remarks and Future Work

This dissertation explored the application of GP models in the analysis of functional phenomena and fault detection across various technical domains. The primary aim was to address key challenges related to control systems, especially in contexts where traditional methods struggle to manage uncertainty and real-time anomaly detection. GP models, due to their probabilistic nature, offered significant advantages in not only predicting system behavior but also estimating the uncertainty of predictions, making them particularly valuable in fault diagnostics.

Throughout my research, I consider the following contributions to be the most significant:

1. Conducted an extensive review of the applications of GPs across control engineering, ranging from optimization to image processing, demonstrating the adaptability and utility of GPs in complex systems within control engineering.

2. Analyzed various GP methods in control engineering, focusing on computational techniques, applications, and future directions, offering insights for optimizing and effectively implementing GP models.

3. Devised algorithms to convert function values at Chebyshev points using FFT into series coefficients, optimizing computation time while maintaining high model accuracy.

4. Introduced a methodology for detecting anomalies in transient states using a combination of GPs and data depth functions.

5. Developed generative GP models for probabilistic estimation and data synthesis, enabling accurate density estimation and efficient sampling in scenarios with incomplete data.

6. Developed a Bayesian GMM integrated with GPs for accurate and early detection of induction motor faults.

7. Applied frequency analysis techniques to induction motor startup currents, establishing methods to improve maintenance and minimize equipment downtime.

8. Designed and optimized GP models for estimating the SoH of Li-ion batteries.

9. Implemented innovative approaches for GP modeling, including ML-II and MCMC sampling, enhancing prediction accuracy and model robustness.

10. Implemented multi-kernel GP methods to refine battery life estimations under diverse operational scenarios, contributing to advancements in battery management systems.

11. Established efficient spatial models for air pollution prediction using INLA and data fusion techniques, integrating diverse data sources.

12. Applied advanced barrier modeling for indoor environments, accounting for obstacles in pollutant dispersion, thereby improving the reliability and precision of air quality predictions.

13. Demonstrated the effectiveness of data fusion to fill gaps in measurement networks, enhancing the accuracy of urban and industrial air pollution models.

Each of my contributions aimed to confirm the formulated theses by addressing the identified problems, which has been successfully accomplished. The solutions developed throughout the research not only resolved these issues but also provided empirical evidence supporting the validity of the proposed theses. The research validated that GP models effectively address complex nonlinear relationships in diagnostics, offering predictive power and uncertainty quantification for real-time analysis. Additionally, generative GP models proved valuable in data-scarce environments by generating synthetic datasets that enhance diagnostic accuracy. These contributions laid a solid foundation for further development.

This research demonstrated GP models' potential, scalability for large datasets and real-time performance remain challenges. Future work should focus on improving efficiency through investigating methods of approximations, developing online GP models, and expanding generative applications in areas like healthcare and environmental monitoring. Further refinement of uncertainty quantification and real-world testing in practical settings, such as smart grids and autonomous vehicles, are also worthwhile development paths to consider for GP models for broader technical diagnostics and predictive maintenance applications.

In summary, this dissertation made substantial contributions to the field of technical diagnostics by advancing the use of GP models in various applications, ranging from battery health monitoring to anomaly detection in industrial processes. The research demonstrated that GP models are not only powerful for predictive purposes but also offer significant advantages in handling uncertainty and generating synthetic data, thus enhancing the robustness and applicability of diagnostic algorithms in complex technical systems.

## Selected bibliography


Adler, R. J. and J. E. Taylor (2007). *Random Fields and Geometry*. Springer Monographs in Mathematics. New York, NY: Springer.

Ahmed, N. and P. Fisher (1968). "Study of algorithmic properties of chebyshev coefficients". In: *International Journal of Computer Mathematics* 2.1-4, pp. 307–317.

*Airly.org* (2023). https://airly.org/. Accessed: 2023-04-20.

Betancourt, M. J. and M. Girolami (2013). *Hamiltonian Monte Carlo for Hierarchical Models*. arXiv: 1312.0906 [stat.ME].

Betancourt, M. (Jan. 2017). "A conceptual introduction to Hamiltonian Monte Carlo". In: arXiv: 1701.02434 [stat.ME].

Bikku, T. (July 2020). "Multi-layered deep learning perceptron approach for health risk prediction". In: *Journal of Big Data* 7.1.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press.

Bogachev, V. (2015). *Gaussian Measures*. Mathematical Surveys and Monographs. American Mathematical Society.

Bole, B., C. Kulkarni, and M. Daigle (2014). *Randomized battery usage data set*. NASA AMES Prognostics Data Repository, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA.

Bottou, L., O. Chapelle, D. DeCoste, and J. Weston (2007). "Approximation Methods for Gaussian Process Regression". In: *Large-Scale Kernel Machines*, pp. 203–223.

Box, G. E. P. and G. M. Jenkins (1970). *Time Series Analysis: Forecasting and Control*. 1st. San Francisco: Holden-Day.

Courant, R. and D. Hilbert (1966). *Methods of Mathematical Physics, Volume 1*. 2nd. New York: Interscience Publishers.

Dudek, A. and J. Baranowski (2022). "Gaussian Processes for Signal Processing and Representation in Control Engineering". In: *Applied Sciences* 12.10.

Garay, F., W. Huaman, and J. Vargas-Machuca (2021a). "State of health diagnostic and remain useful life prognostic for lithium-ion battery by combining multi-kernel in Gaussian process regression". In: *2021 IEEE XXVIII International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pp. 1–4.

Garay, F., W. Huaman, and J. Vargas-Machuca (2021b). "State of health diagnostic and remain useful life prognostic for lithium-ion battery by combining multi-kernel in Gaussian process regression". In: *2021 IEEE XXVIII International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pp. 1–4.

Gönül, M., O. A. Kutlar, A. T. Calik, and F. Orcun Parlak (2021). "Prediction of oil dilution formation rate due to post injections in diesel engines by using Gaussian process". In: *Fuel* 305, p. 121608.

Gonzalvez, J., E. Lezmi, T. Roncalli, and J. Xu (2019). *Financial Applications of Gaussian Processes and Bayesian Optimization*. arXiv: 1903.04841 [q-fin.PM].

Kolmogorov, A. N. (1956). *Foundations of the Theory of Probability*. 2nd. Originally published in 1933. New York: Chelsea Publishing Company.

Krainski, E., V. Gómez Rubio, H. Bakka, A. Lenzi, D. Castro-Camilo, D. Simpson, F. Lindgren, and H. Rue (Sept. 2018). *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*.

Lindgren, F., H. Rue, and J. Lindström (2011). "An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.4, pp. 423–498.

Mackay, D. J. C. (1998). "Introduction to Gaussian processes". In.

Margossian, C. and A. Gelman (2023). "For how many iterations should we run Markov chain Monte Carlo?" In: *arXiv preprint arXiv:2311.02726*.

Matérn, B. (1960). *Spatial Variation*. Vol. 49. Meddelanden fran Statens Skogsforskningsinstitut 5. Stockholm, Sweden: Statens Skogsforskningsinstitut.

*Portal Jakość Powietrza GIOŚ. Bank danych pomiarowych* (2023). https://powietrze.gios.gov.pl/pjp/archives. Accessed: 2023-04-20.

Powell, M. J. D. (1985). "A theory of radial basis function approximation". In: *Algorithms for Approximation*, pp. 143–167.

Quiñonero-candela, J., C. E. Rasmussen, and R. Herbrich (2005). "A unifying view of sparse approximate Gaussian process regression". In: *Journal of Machine Learning Research* 6, p. 2005.

Rasmussen, C. and C. K. I. Williams (2006). *Gaussian Processes in machine learning*. MIT Press.

Riutort-Mayol, G., P.-C. Bürkner, M. R. Andersen, A. Solin, and A. Vehtari (2020). *Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming*.

Rue, H. and L. Held (Feb. 2005). "Gaussian Markov Random Fields: Theory and Applications". In: *Gaussian Markov Random Fields* 104.

Rue, H., S. Martino, and N. Chopin (Apr. 2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations". In: *Journal of the Royal Statistical Society Series B* 71, pp. 319–392.

Schölkopf, B. and A. J. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press.

Shubin, M. (1987). *Pseudodifferential Operators and Spectral Theory*. Springer Series in Soviet Mathematics. Berlin: Springer-Verlag.

Simpson, D. (2021a). *Un garçon pas comme les autres (Bayes): Yes but what is a Gaussian process? or, Once, twice, three times a definition; or A descent into madness*.

— (Nov. 3, 2021b). *Yes but What Is a Gaussian Process? Or, Once, Twice, Three Times a Definition; or A Descent into Madness*. URL: https://dansblog.netlify.app/yes-but-what-is-a-gaussian-process-or-once-twice-three-times-a-definition-or-a-descent-into-madness.

Solin, A. and S. Särkkä (2014). "Hilbert space methods for reduced-rank Gaussian process regression". In: *arXiv preprint arXiv:1401.5508*.

Stan Development Team (2024). *Stan User's Guide*. https://mc-stan.org/users/documentation/. Accessed: 05.01.2024.

Tagade, P., K. S. Hariharan, S. Ramachandran, A. Khandelwal, A. Naha, S. M. Kolake, and S. H. Han (2020). "Deep Gaussian process regression for lithium-ion battery health prognosis and degradation mode diagnosis". In: *Journal of Power Sources* 445, p. 227281.

Trefethen, L. N. (2012). *Approximation Theory and Approximation Practice.* SIAM, pp. I–VII, 1–305.

Tukey, J. W. (1975). "Mathematics and the picturing of data". In: *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*. Vol. 2, pp. 523–531.

Wang, B. and Z. Mao (2019). "Outlier detection based on Gaussian process with application to industrial processes". In: *Applied Soft Computing* 76, pp. 505–516.

Zeng, A., H. Ho, and Y. Yu (2020). "Prediction of building electricity usage using Gaussian Process Regression". In: *Journal of Building Engineering* 28, p. 101054.

# Appendix A

# Publication Series - Full Texts

applied sciences

MDPI

*Review*

# Gaussian Processes for Signal Processing and Representation in Control Engineering

**Adrian Dudek and Jerzy Baranowski ***

Department of Automatic Control & Robotics, AGH University of Science & Technology, 30-059 Cracow, Poland; addudek@agh.edu.pl
* Correspondence: jb@agh.edu.pl

**Abstract:** The Gaussian process is an increasingly well-known type of stochastic process, which is a generalization of the Gaussian probability distribution. It allows us to model complex functions thanks to its flexibility, which would not be possible with the use of other tools. Gaussian processes also have a couple of other features that are used in various branches of automation with positive results, ranging from industrial processes to image processing. There are also many ways of setting up the Gaussian processes, which required knowledge on the topic and depend on the presented problem. Considerations on these topics lead to the conclusion that the current state of practical usefulness of Gaussian processes increases significantly, therefore the deepening of knowledge about the ways of its use is highly suggested. In this review, we present selected technical applications of Gaussian Processes allowing an understanding of their broad applicability.

**Keywords:** Gaussian process; control engineering; signal processing; practical applications; optimization; modelling



**Citation:** Dudek, A.; Baranowski, J. Gaussian Processes for Signal Processing and Representation in Control Engineering. *Appl. Sci.* **2022**, *12*, 4946. https://doi.org/10.3390/app12104946

Academic Editors: Jeng-Shyang Pan, Huanyu Liu, Jun-Bao Li, Meng Li and Shi-Huang Chen

Received: 31 March 2022
Accepted: 11 May 2022
Published: 13 May 2022



## 1. Introduction

Complicated phenomena surround us, motivating us to find their models. Such models allow solutions to the problems of modern science. Those models, however, need to represent the uncertainty of the world and of the measurement process itself. This is the reason for the popularity of statistical models. Intuitively, we associate them with linear (regression) models and perhaps with generalized linear models (e.g., logistic regression). However, needs for modelling are much more advanced, especially when we need to represent unknown nonlinear functional relationships. Those issues until relatively recently were unavailable because of their computational complexity. Fortunately, current computing technologies and algorithms allow us to use advanced statistical models, like Gaussian processes.

Gaussian processes (GP) are a non-parametric method of specifying probabilistic models on function spaces. Such modeling is potentially intractable computationally, however, practice is much easier. Briefly speaking, GPs are a generalization of a Gaussian distribution such that every finite set of random variables has a multivariate normal distribution. It is in the sense that every finite linear combination of them has a normal distribution [1]. The resultant distribution of the GP is the joint distribution of all these random variables. This ensures that the solution benefits from the normal distribution features. This way, inference on functional spaces reduces to inference on a set of points belonging to those functions. The distinguishing feature of GP is getting the result, apart from the solution itself, also the uncertainty range around it. This is with much success used in multiple applications.

Our review of GP applications differs from existing ones, because of the spectrum of covered fields and problems, i.e., control engineering and signal processing. Available reviews cover among the others:

- machine learning applications for speech recognition, like [2];
- finance risk management [3];
- sructural chemistry [4].

There are also works, covering technical (computational) aspects of GPs, like [5].

In our review we focus on applications of GPs in control engineering, discuss relevant issues connected and propose of future directions in this field. Introduction of GPs in control engineering is a relatively recent development. The reason for it were computational issues with large matrix inversions [6]. Modern computing power, efficient algorithms and accessible software allow fitting GPs to a wide scope of problems provided data are available. Applications of GPs.

Applications of GPs are many, but we can group them into problems such as regression, classification, prediction etc. Regardless of the problem, a well-chosen and set up GP can handle with numerous cases. Its appropriate application to problems where it outclasses the competition is the key. The most common of such cases are problems where obtaining data is difficult or they are deficient, and when problems are difficult to model using other methods [7,8].

Of course, it is not possible to cover all the currently usage on each issue equally, so we would like to describe as widely as possible the field of the use of GP. Our goal is to reliably determine the current state of the field and draw conclusions from it. We connect specific examples of problems with industrial branches, problems where modeling is difficult, decision control support, where the time of the operation is important, and many more. However, besides the solutions themselves, it is also worth considering the essence of a way of accomplishing them with a range of possibility of methods of handling the problems which occur with its usage. We organised the rest of the paper in the following way. First, we present a basic theory of GPs. Then we show computational methods and problems correlated to them. The next few sections introduce various examples of GP usage in practice in broadly understood control engineering. Paper ends with discussion, proposed future direction and conclusions.

## 2. Gaussian Process

Carl Edward Rasmussen [1] came up with the following definition, which captures the essence of Gaussian Process:

> A Gaussian process is a collection of random variables, any Gaussian process finite number of which have a joint Gaussian distribution.

If we define mean function $m(x)$ and the covariance function mean function $k(x, x')$ of a real process $f(x)$ as:

$$m(x) = E[f(x)] \tag{1}$$

$$k(x, x') = E[f(x) - m(x))(f(x') - m(x'))]. \tag{2}$$

Then we can define GP as:

$$f(x) \sim GP(m(x), k(x, x')). \tag{3}$$

The mean and covariance functions completely characterize the GP prior, which controls the full behavior of function $f$ prior. If we assume that $\boldsymbol{f} = \{f(\boldsymbol{x}_n)\}_{n=1}^{N}$ then the prior distribution of $\boldsymbol{f}$ is a multivariate Gaussian distribution $\boldsymbol{f} \sim Normal(\boldsymbol{\mu}, \boldsymbol{K})$, where mean is $\boldsymbol{\mu} = \{\mu(\boldsymbol{x}_n)\}_{n=1}^{N}$, and $\boldsymbol{K}$ is covariance matrix, where $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ The joint distribution of $f$ and a new $\tilde{f}$ is also a multivariate Gaussian with assumption of mean equal 0 is presented by the Formula (4):

$$p(\boldsymbol{f}, \tilde{f}) = \text{Normal}\left( \begin{bmatrix} \boldsymbol{f} \\ f^* \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \boldsymbol{K}_{f,f} & \boldsymbol{k}_{f,\tilde{f}} \\ \boldsymbol{k}_{\tilde{f},f} & k_{\tilde{f},\tilde{f}'} \end{bmatrix} \right) \tag{4}$$

where $\boldsymbol{k}_{f,\tilde{f}}$ is the covariance between $f$ and $\tilde{f}$ , and $k_{\tilde{f}}, \tilde{f}$ is the prior variance of $\tilde{f}$ [9]. On such a prior distribution, we then make an inference from the GP model, by adding the data from the process we would like to model.

In general, GP is a stochastic process used for modeling data, which were observed over time, space or both [10]. Main thing that can characterise GP is that is a kind of generalization of normal probability distributions, where each of them describes a random variable (scalar or vector if we deal with multivariate distribution). This kind of generalization can be defined as follows: let us assume a set of index $T$, then GP on set $T$ is a collection of random variables indexed by a continuous variable, that is:

$$\{f(t) : t \in \mathcal{T}\}, \tag{5}$$

which has a property that for

$$\forall n \in N, \forall t_1, \ldots, t_n \in \mathcal{T}, \tag{6}$$

the marginal distribution of any finite subset of random variables

$$\{f(t_1), f(t_2), \ldots, f(t_N)\}, \tag{7}$$

is a multivariate Gaussian distribution [9,11].

GP can be fully determined by only declaring mean and covariance functions [1,10,12]. Mean in most cases is set to value "0", because such a setting can be useful, simplifies matters and is not a difficult requirement to fulfill. Of course, there are examples where we would like to change the mean e.g., for better model interpretability or the specification of our prior [1,12]. Covariance function (also called kernel function) represents a similarity between data points [13]. Note that usually covariance is chosen from the set of already defined functions. One should at least pick one, which represents the prior beliefs of the problem. However, in fact, covariance function can be any function, which has the property of generating a positive definite covariance matrix [14]. Anyway, creating and defining new covariance functions, which will simultaneously be correct and have a practical usage, can be really difficult.

The most basic and common kernel function is the Radial Basis Function (RBF), which is also called Squared Exponential (SE) and is defined with the formula (8):

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right). \tag{8}$$

Its main property is that its value is usually only dependent on the distance from the specified point. The parameter which RBF kernel uses is $l$ as a characteristic length scale and $d(x_i, x_j)$ is the Euclidean distance between points. RBF is infinitely differentiable. That means the GP with this kernel has a mean square derivatives for all orders. Despite that RBF is very smooth, strong smoothness assumptions are unrealistic and the using kernel from Matérn family is recommended [1].

The Matérn kernel family is a kind of generalization of RBF function. The general formula for Matérn covariance functions is given by (9):

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right), \tag{9}$$

where $K_\nu$ is a modified Bessel function, $l$ is length-scale parameter and $v$ is a parameter, which allows to change smoothness of function, which give an ability to flexibly control the function in relation to the one we want to model. Also $d(x_i, x_j)$ is the Euclidean distance. Of particular note are frequently used parameter $v$ with values of $3/2$, which is used for

learning functions, which are at least once differentiable and 5/2 for functions at least twice differentiable [1].

Additionally, worth mentioning is the Exp-Sine-Squared kernel function (also called periodic kernel, proposed by MacKay [15]). Main differential here is a possibility to catch periodicity of a given functions and of course model them. Formula (10) represents Exp-Sine-Squared function.

$$k(x_i, x_j) = \exp\left(-\frac{2\sin^2\left(\pi d\left(x_i, x_j\right)/p\right)}{l^2}\right). \tag{10}$$

In addition to $l$ length-scale parameter we have $p$ as a periodicity parameter, which allows flexibility with respect to periodicity. This kernel comes in handy when we would like to model some kind of seasonality of our problem.

As mentioned, each of the selected covariance functions has a set of free parameters called hyperparameters whose values are need to be determined. Properly handling this task is one of the main difficulties in a GP usage. As can be seen in the next part of this paper, currently the ML-II approach is widely used, where we calculate parameters by a sort of marginal likelihood optimization. Its advantage is ease of use and uncomplicated computation process with a major drawback of less accuracy. For instance, it can suffer from a multiple local maxima problem. There are also solutions for calculating hyperparameters such as statistics methods such as hierarchical Bayesian models. More on this topic can be found in next chapter [1].

Equation (3) is widely used across all the literature, sometimes in different forms, but always all comes to this formula. GP has a variety of fields which can be used, where machine learning and statistics methods are applicable. GP works best under the difficult circumstances, where building a model could be a thought task, because of hard to obtain data. The overall usage and how it is applied in nowadays problems is described in detail in the following sections.

## 3. Computational Methods of Gaussian Process

### *3.1. Modelling and Computation of Hyperparameters*

The first problem, which comes to computing the GP is the hyperparameters of chosen kernel. Dealing with setting up those parameters mostly comes from complex Bayesian inference in complex hierarchical model with three levels of inference: posterior over parameters, posterior over hyperparameters and at the top posterior of model [1]. All levels can be described with formulas, where $\mathbf{w}$ is a set of parameters, $\boldsymbol{\theta}$ are hyperparameters and $\mathcal{H}_i$ as set of possible model structures. Equation (11) (posterior over parameters) is the first level:

$$p(\mathbf{w} \mid \mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y} \mid X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w} \mid \boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_i)}, \tag{11}$$

where $p(\mathbf{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_i)$ is independent of the parameters and calledmarginal likelihood. Marginal likelihood is given by equation number (12):

$$p(\mathbf{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y} \mid X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w} \mid \boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w}. \tag{12}$$

The second level equation (posterior over hyperparameters) is given by:

$$p(\boldsymbol{\theta} \mid \mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta} \mid \mathcal{H}_i)}{p(\mathbf{y} \mid X, \mathcal{H}_i)}, \tag{13}$$

where normalizing constant is:

$$p(\mathbf{y} \mid X, \mathcal{H}_i) = \int p(\mathbf{y} \mid X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta} \mid \mathcal{H}_i)d\boldsymbol{\theta}. \tag{14}$$

The top level (posterior of model) is given by:

$$p(\mathcal{H}_i \mid \mathbf{y}, X) = \frac{p(\mathbf{y} \mid X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y} \mid X)}. \tag{15}$$

This approach can be difficult even to prepare, because it requires not only knowledge about the problem, but also the necessary Bayesian inference skills. Therefore, a fully Bayesian approach is a much less common solution in practice, although it may bring concrete results. Such modeling may result in a higher accuracy of the model and its easier adaptation in the case of small amounts of data. The shown integrals may not be analytically tractable, so idea is to use an analytical approximations or Markov chain Monte Carlo (MCMC) methods of sampling from a probability distribution. MCMC itself requires as to use more advanced methods like building whole model in Stan. This method, however, clearly makes GP more suitable to the certain problem with no harm to its overall flexibility in specific field. Drawback of such approach is computational burden of heavy computation requirements, so it may appear a necessity of approximation. Another way to accomplish the task of getting appropriate hyperparameters is called ML-II (type II maximum likelihood), which is based on maximizing marginal likelihood from Equation (12). Then the integral from Equation (14) can be approximated with use of a local expansion around the maximum (the Laplace approximation). In fact we should be careful, with such optimization approach, because of its tendency to overfitting (especially with more number of hyperparameters). In general ML-II has a lesser accuracy then method involving MCMC, but is easier in implementation and use in practical examples, especially because of lesser computation performance demand [1].

### *3.2. Approximation of Gaussian Process*

GP computation is problematic, because of the earlier mentioned necessity of matrix inversion, which grows in size with the size of the training dataset. Nowadays, computers can easily handle reasonable sized problems with GP as solution. On the other hand, if we are dealing with large GP model and its computation becomes a burden, there are several methods of approximating GP. An example of such solution are sparse methods. Common to all different variations is that only a subset of the latent variables is treated exactly, and the remaining variables are given some approximate, which is computationally easier [16]. The authors of [16,17] had described the whole idea in detail, but in general their approach is interpretable as exact inference with an approximate prior.

To be more specific, if we infer from the GP model by putting the joint GP prior on training $f$ and testing latent values $f_*$ and combining it with the likelihood $p(y|f)$ we achieve a joint posterior given by Equation (16).

$$p(\boldsymbol{f}, \boldsymbol{f}_* \mid \mathbf{y}) = \frac{p(\boldsymbol{f}, \boldsymbol{f}_*)p(\mathbf{y} \mid \boldsymbol{f})}{p(\mathbf{y})}. \tag{16}$$

Then there is a need to marginalize out the unwanted training set latent variables:

$$p(\boldsymbol{f}_* \mid \mathbf{y}) = \int p(\boldsymbol{f}, \boldsymbol{f}_* \mid \mathbf{y}) \mathrm{d}\boldsymbol{f} = \frac{1}{p(\mathbf{y})} \int p(\mathbf{y} \mid \boldsymbol{f}) p(\boldsymbol{f}, \boldsymbol{f}_*) \mathrm{d}\boldsymbol{f}. \tag{17}$$

The posterior predictive is received:

$$p(\boldsymbol{f}, \boldsymbol{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{\boldsymbol{f},\boldsymbol{f}} & K_{*,\boldsymbol{f}} \\ K_{\boldsymbol{f},*} & K_{*,*} \end{bmatrix}\right), \quad \text{and} \quad p(\mathbf{y} \mid \boldsymbol{f}) = \mathcal{N}\left(\boldsymbol{f}, \sigma^2_{\text{noise}} I\right). \tag{18}$$

Since both factors in the integral are Gaussian, the integral can be evaluated in closed form to give the Gaussian predictive distribution:

$$p(\boldsymbol{f}_* \mid \mathbf{y}) = \mathcal{N}\left(K_{*,f}\left(K_{f,f} + \sigma^2_{\text{noise}}\, I\right)^{-1}\mathbf{y}, K_{*,*} - K_{*,f}\left(K_{f,f} + \sigma^2_{\text{noise}}\, I\right)^{-1}K_{f,*}\right). \quad (19)$$

The problem is that the equation requires the earlier mentioned inversion of matrix, the size of which depends on the number of training cases [16,17].

The idea is to modify joint prior $p(\boldsymbol{f}_*, \boldsymbol{f})$ to reduce its computation requirements. In order to achieve this, inducing variables $u$, which are a set of latent variables that are values of GP, corresponding to a set of input locations (which are called the inducing inputs). The consistency of GP allows us to recover the mentioned joint prior to $p(\boldsymbol{f}_*, \boldsymbol{f})$ by integrating (marginalizing) out u from joint GP prior $p(\boldsymbol{f}_*, \boldsymbol{f}, u)$.

$$p(\boldsymbol{f}_*, \boldsymbol{f}) = \int p(\boldsymbol{f}_*, \boldsymbol{f}, \mathbf{u})\mathrm{d}\mathbf{u} = \int p(\boldsymbol{f}_*, \boldsymbol{f} \mid \mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u}, \quad \text{where} \quad p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{u},\mathbf{u}}) \quad (20)$$

Then joint prior is approximated by assuming that $f_*$ and $f$ are conditionally independent given $u$.

$$p(\boldsymbol{f}_*, \boldsymbol{f}) \simeq q(\boldsymbol{f}_*, \boldsymbol{f}) = \int q(\boldsymbol{f}_* \mid \mathbf{u})q(\boldsymbol{f} \mid \mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u}. \quad (21)$$

The way of choosing the inducing variables as well as making addition assumptions to Equation (21) depends on an exact chosen sparse algorithm [16,17].

Another approach to the approximation of GP is presented in [9] as Hilbert space approximate GP model for a low-rank representation of stationary GPs. Main principal consist of using basis function approximations based on approximation via Laplace eigenfunctions for stationary covariance functions proposed in [18]. GP is approximated with the use of a linear model, by the use of basis function expansion. This way, the whole computation and inference can be done significantly quicker. Linear structure has advantages like simplifying approximation computation and making it easier to use GP as a latent function, when we approach non-Gaussian models, which gives more overall flexibility. The Laplace eigenfunctions computation is independent from the choice of kernel (and its hyperparameters). It also can be performed analytically. The idea is to create the covariance operator of a stationary covariance function as a pseudo-differential operator constructed as a series of Laplace operators. This operator $\mathcal{K}$ (over function $f(x)$ ) is formulated as in Formula (22):

$$\mathcal{K}f(\boldsymbol{x}) = \int k(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}', \quad (22)$$

where $k(\boldsymbol{x}, \boldsymbol{x}')$ is of course covariance function. This operator is approximated with Hilbert space methods on a compact subset, subject to boundary conditions. This is required to the the approximation of a stationary covariance function as a series expansion of eigenvalues and eigenfunctions of the Laplacian operator. Mathematical details provided by Solin and Särkkä in [18] give a concrete approximation of covariance, featured in Formula (23):

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_j S\left(\sqrt{\lambda_j}\right)\phi_j(\boldsymbol{x})\phi_j(\boldsymbol{x}'), \quad (23)$$

where $S(\cdot)$ is spectral density of the covariance function, $\lambda_y$ is is the $j$th eigenvalue and $\phi(\cdot)$ is the eigenfunction of the Laplace operator. Calculation of such formula is relatively easy, because of it consist of evaluating the spectral density at the square roots of the eigenvalues and multiplying them with the eigenfunctions of the Laplace operator [18]. Such approximated GP is then called HSGP. The authors of [9] have created a whole framework , which allows us to work with HSGP with, e.g., Stan and the MCMC sampling method. Moreover, they added methods for GP with periodic covariance functions.

## 4. Anomaly Detection in Industrial Processes

Gaussian Process is growing in popularity of usage in domains connected with statistics and machine learning. It is an especially efficient solution, while we are dealing with a problem of obtaining data e.g., for faulty cases or just an overall lack of comprehensive data. The main areas where GP is used are: regression, prediction, classification and identification. Most of solutions, which are currently used in practice, in whole or in part, is classified to at least one of the mentioned areas.

The usage of statistics and machine learning in the industrial process is not anything new. There is a large debate about which approach could give the best results on which issues. However, on the subject of detecting faulty states, GP seems to be widely used and appreciated. In industrial processes, it is key to guaranteeing continuous workflow without any disturbances and simultaneously taking care of production quality [19]. Any failures can lead to significant costs, so assuring the reliable anomaly detection method has a great impact on the world's economy.

In "Outlier detection based on Gaussian process with application to industrial processes" [19] we are encountering the problem of the detection of anomalies called outliers. They agree that the the definition of an outlier can vary depending on the problem, but basically there are "patterns in data that do not conform to a well defined notion of normal behavior". In the case of industrial processes, outliers are not so easy to measure and contaminate, because they can come from different sources such as parameters, sensors and actuators fault or any structural changes. Principal component analysis (PCA) and partial least squares (PLS) are the two most common techniques used in anomaly detection but suffers from practical conditions of gathering abnormal datapoints—outliers. GP stands as an alternative for typical model-based methods in order to handle industrial scales [19].

The authors' main workflow consists of:

- extending the GP models in order to detect outliers in industrial processes;
- selecting GP specific parameters suitable for problem;
- testing effectiveness of algorithm with synthetic and real-world data.

For calculating outlier score both regression and classification frameworks of GP were used. Both were carefully set up with special care for defining specific selections of its parameters. Different approaches for selecting every parameter used in the paper method were discussed. However, in conclusion, the mean for all approaches in the paper was set to zero and the kernel function has been set to squared exponential kernel and to a simple composite. In addition, the authors used a specified likelihood function: Student's t and Laplace distribution and determined special inference methods [19].

At the end they came up with three solutions based on GP regression and one based on GP classification. Each of them has been exhaustively tested on the Tennessee Eastman benchmark process, electric arc furnace process control and wind tunnel process control. Presented methods were compared to well-known competitors such as: Gaussian mixture model (GMM), k-mean, k-nearest neighbor, support vector data description and principal component analysis (PCA). The drawn conclusion is clear: compared with traditional detection methods, proposed scheme has less assumptions and is more suitable for modern industrial processes [19].

Synonymous problems were encountered in article [20], where anomalous situations are detected in Industry 4.0 environment. The main principles of these two projects are similar—diagnosis of faulty states in industrial usability. However, here we encounter Big Data environment with concepts called Digital Twin (DT). DT is defined as "a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level". According to [20], the main characteristics of DT are as follows:

1. DTs are virtual dynamic representations of physical systems;
2. DTs exchange data with the physical system automatically and bidirectionally;
3. DTs cover the entire product life cycle.

The DT module was used alongside the anomaly detection algorithm. The presented use case scenario consists of the creation of the mentioned algorithm, which is based on GP in order to detect defective products in the extrusion process of aluminum profiles. Setting up the GP usually consists of choosing predefined mean and covariance functions. However, the authors used some workaround described in [21–23] in order to avoid a situation where chosen parameters do not fit the problem. The training multiple different functions and proceeding with the best model and use of linear combinations of basis kernel functions were proposed there. Moreover the threshold to divide data was defined. Whole dataset was divided into three categories: training with only good data. validation and training set with both good and anomalous data. First was used to to approximate the probability density function by estimating the mean and the covariance function. Second helps with declaring the right likelihood threshold. Last one provided evaluation for the model [20].

The model was tested in a relevant environment with a result performance of 0% FPR and 97.8% TPR as a result, so it can be conveniently told that it is possible to detect anomaly states (too-high pressures at the machine) and faulty products. Moreover, the authors calculated the impact on production efficiency, which leads to significant cost reduction, energy savings, and quality improvements (savings up to 80,000 EUR/a) [20].

Other anomaly detection issue in industrial processes can be encountered in [7]. The authors propose a method to to fill the gap in the transient detection using approach consisting of:

- modelling of transient states (both healthy and suspect of faulty) using Gaussian Processes;
- using distribution obtained with a Gaussian Process analyze the data depth distribution for healthy and faulty signals;
- difference in the depth is an anomaly indicator.

The experiment was conducted on the system of three water tanks filled with water in a specific order, all controlled by automated system with a constantly measured water level. Anomalous situations were simulated with manual valves, which created the situation of unexpected and failure states, where water flow was disturbed. Datasets were gathered from both scenarios. The authors then used a healthy set to set up GP. The chosen kernel function was Rational Quadratic kernel, which can be considered as a scale mixture (an infinite sum) of Radial Basis Function kernels with different characteristic length scales. GP fitting consisted of a running optimization algorithm called Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LM-BFGS) in order to obtain specific hyperparameters. Then this GP parameters were used to create models for both health and failure cases. These models allowed us to generate a large number of samples (transient states) for those cases. Samples were used for data depth analysis with Mahalanobis and Euclidean depths. This analysis resulted in the difference in healthy and faulty samples, which led to a method of anomaly detection with the use of GP models [7].

## 5. Gaussian Process Usage in Terms of Electrical Energy

### *5.1. Prediction of Battery Condition Parameters*

Other areas of interest are complex objects in terms of modeling. The exemplary problem is predicting the life parameters of batteries. This issue can be found in several papers, where authors focus on different aspects of battery condition values, but always using GP as a method of calculating these.

All of the selected articles face the same problem of lithium-ion cells, which degrade in time on their own and while being used. This causes a significant decrease of total capacity and an increase of inner resistance. So it is really important to have a way to predict and simulate the state of the remaining usability of a battery, even with economics reasons. The process and description of cell degradation is very complex and depends on various variables. Classical methods are based on fitting a somewhat arbitrary parametric function to laboratory data and, on the other hand, electrochemical modelling of the physics of

degradation. Alternative solutions are machine learning ones or non-parametric such as support-vector machine or GP [24,25].

The authors of each paper benchmarked their proposal solutions on an open source data base, which belongs to NASA, of various different batteries. However, they researched dissimilar kind of problem with divergent approaches. In [24–26] the main purposes are to calculate State of Health (SoH) and prognose Remaining Useful Life (RUL) states of battery. SoH refers to reducing capacity and increasing resistance over time and RUL can be described as the time that elapses from now until the end of the total useful life. In [27], the authors aim to calculate the "knee point". Achieving knee point indicates that significant deterioration has occurred and the battery suddenly accelerates in ageing. At this point it is often suggested to replace cells [27]. Everyone admits that the GP fits very well in solving such a problem and is surprisingly uncommon as a used method. Moreover it has a crucial advantage over other approaches such as neural networks, because of its ability to capture model uncertainty.

Methods of using GP were slightly different in each scenario. Approaches used in [24,25] were based on simple assumption of mean value. The distinction comes with choosing the kernel function. In [25] four models were created, two based on the Matérn kernel, and the other two on a linear one. Additionally two models were using gradient boosting regression (GBR) as a point of reference. Moreover, each pair uses different methods of capturing long term trends. On the other hand, in [24] the authors used different settings of covariance functions, where they use multikernel GP approach. They combine the Radial Basis Function (RBF), the Matérn, Rotational Quadratic (RQ) and Exp Sine Squared (ESS) in various sets. The first set of hyperparameters is constant and then optimized over time with ML methods. GP in test produced satisfying results and compared to other data-driven approaches it offers great flexibility with a relatively high accuracy on mean capacity predictions (5% of the actual values). The conclusion of the multikernel approach was that the combination of Matérn kernels (Ma1.5 + Ma2.5) gave the best results. The authors draw attention to the future use of GP in other aspects connected with battery cells as well as researching other approaches based on the GP method [24,25].

Other methods based on GP can be seen in the approach presented in [26]. Of course, the authors solving the same problem of predicting the end of life time of the cells and SoH in general, but in a very different manner. They propose deep Gaussian Process variations, which use GP to model mapping between layers, and matrix-variate Gaussian distribution to model the correlation between nodes of a give layer [26]. That allows to estimate the capacity of the cell using partial charge-discharge time series data (voltage, current, temperature). It also eliminates the need for input feature extraction. The authors focused on a special GP variant known as the special Gaussian Process (SGP). SGP solves two problems of GP Regression: GPR accuracy dependence on the locations of training data points (its highly accurate near those datapoints) and computational complexity (of inversed kernel matrix). In SGP it is assumed that the training dataset is augmented with pseudo-inputs $Z$ and the associated outputs $u = f(K)$. SGP has been also extended by Damianou and Neil [28] to deep Gaussian Process Regression. The main idea is to use the $l$ layer as on input to the $l + 1$ layer. Both SGP and DGPR concepts can be seen in Figure 1. DGPR in fact is created by stacking some SGPs on top of each other. Layers between first input and last output are called "hidden".

The last reviewed approach in [27] focuses on searching and estimating a special moment in the battery life cycle called "knee point". The algorithm is also different as it consists not only of the Gaussian Process but is also augmented with feature extraction and selection modules. The authors point out that not all batteries behave just like those from the provided datasets so the knee point can happen in different points that could be assumed. The proper indication of knee points is important in that it informs that the cell will soon need a replacement [27]. There also exist some other estimation methods (for instance approaching 80% of capacity), but they are mostly based on other searching principals such as point estimations instead of whole transients. The whole idea presented

in Figure 2 takes a special algorithm to extract features from the datasets (voltage, current etc.), select a relevant subset of key features and finally feed the Gaussian Process regression model with it. The output should be a capacity fade trajectory estimation [27]. GP was poorly conditioned by initial settings. The hyperparameters were fitted to the training data in the standard way using maximum marginal likelihood estimation. The kernel was set to the Matérn 2.5 function, because of previous research in [24,25], where it has been shown that it works well. Moreover, contrary to previous examples, the authors used different datasets in their research. The general conclusion is that the whole idea makes practical sense. The accuracy is about half a day of time for achieving knee point (or 2.6% across 600 predictions). They also used different subsets of data, but it turned out that the more the better. Overall, the information on how all different features correlate with each other is extremely viable with the use of the accurate forecast of capacity, end of life and the knee point. Moreover, they suggest that the problem should still be developed for other approaches and deeper research of data correlation.

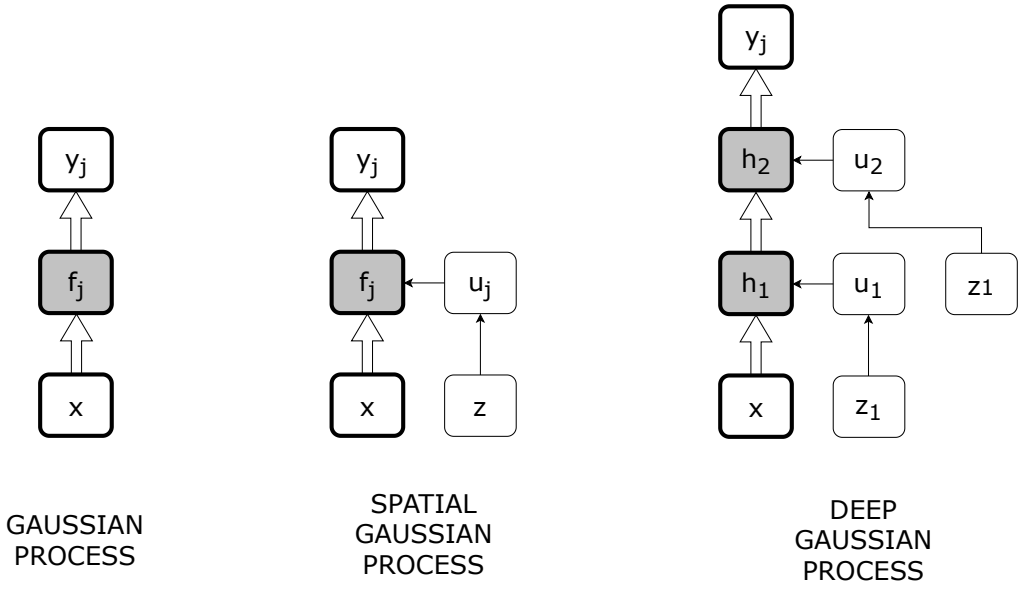



**Figure 1.** Particular problem of predicting battery parameters such as State of Health and Remaining Useful Life was not solved but is a simple use of GP. In order to receive desirable solution, the authors enhanced GP into Deep GP (DGP), which uses principals of spatial GP (SGP). The figure shows the major differences in such approaches. To create an SGP, a standard GP dataset is augmented with pseudo-inputs $z$ and the outputs $u$ associated with them. DGP uses the SGP idea and stacks layers of outputs and pseudo-inputs on each other, which creates hidden layers $h$. This solves problems of accuracy dependence on the locations of data points and computational complexity [28].


**Figure 2.** In the problem of predicting the knee point (point of drastic acceleration of battery condition degradation), the authors developed a strict algorithm which used a large amount of data as an input. To receive useful information from those datasets, they created extraction and selection algorithms for features. First, it has to only obtain the information necessary to build the model (like voltage, current etc.) and then specific features of it had to be selected in order to pass them to GP. This allows them to build a model, which has a possibility to achieve capacity fade trajectory estimation and predict knee point appearance from it [27].

### *5.2. Electricity Usage and Generating Prediction*

Nowadays, more and more devices and structures use electricity, so the prediction of its usage can be an important matter especially in terms of smart cars or smart homes where it can be both a matter of proper operation and of huge savings. The same principles guide the authors of [29], where the main aim is to predict energy consumption by different kinds of buildings which can lead to optimizing it through special control and low-energy strategies.

General workflow starts from collecting and preparing data, which comes from actual buildings. Being more accurate, from six different commercial buildings and its energy use database. The principal parameters were chosen by taking into account their impacts on the building electricity use. Those impacts were measured by calculating the correlation between energy use and meteorological parameters. The authors also considered users' behavior. Then the whole set was filtered, cleansed and parsed in order to be used to predict the electricity usage [29].

GP regression is used as the prediction algorithm. The authors claim that, compared to the popular support vector machine (SVM) method, GPR shows great advantages in learning the kernel and regularization parameters, integrated feature selection, and generating fully probabilistic predictions. The model treats the occupancy schedule and the meteorological parameters as inputs. The output is building energy consumption. GP models were trained on a training dataset, and the model gives a numeric value of energy use of the test dataset based on the known inputs. The whole framework is shown in Figure 3 [29].

Data used in the creation and validation of model came from shopping malls, offices and hotels. The statistical analysis of results were conducted by authors. They calculated Root Mean Squared Error (RMSE) as well as Normalized Mean Bias Error (NMBE) and R-square to catch the accuracy of the model's prediction. Conclusions were that GPR was regarded as a very accurate way of predicting the energy usage in this case study. It also generates predictions in very limited CPU time (at around 0.02 s per prediction). GPR can catch different energy patterns and can be used with very different inputs such as weather conditions, occupancy activities and others. Moreover, the authors claim that GPR can also be used with problems connected to predictions of the thermal dynamics of buildings and variable operations. In addition it could be compiled with a control system based on deviation of prediction.

On the other hand, GP can also be used for power generation prediction. In [30] we can see the comparison of some methods used for the prediction of the wind turbine power curve (WTPC). WTPC is a characteristic of the turbine which shows the relationship between output power and wind speed. Stochastic and intermittence exist in the generated wind power, which affect the safety of the power grid. The prediction of wind power is also very critical in wind power dispatching and even in enhancing wind turbine controllability [30]. Methods chosen by authors to predict WTPC are as follows:

- Gaussian Bin method;
- kernel density estimation (KDE);
- conditional kernel density estimation (CKDE);
- conditional probability via Copula;
- relevance vector machine (RVM);
- and of course Gaussian Process Regression.

In this review, we will focus only on GPR usage and its results in this problem. The whole process of setting up GP lacks a detailed description. Hyperparameters of coviariance functions are obtained by maximizing the marginal likelihood. The authors also notice that GPR is adopted for WTPC modeling, so both point estimation and confidence intervals can be calculated. Interval modeling was based on a brief description in [31]. The main issue with GPR is computational complexity, which is unfeasible for large datasets. Results differ amongst all methods and depend on what was to be accomplished. GPR was best for modeling efficiency and is suitable for online estimation and can be applied in super-short-term or short-term wind power prediction. Statistical models (Gaussian Bin,

KDE, Copula) can be used in offline estimations as operational performance evaluation of wind turbines or wind farms [30].

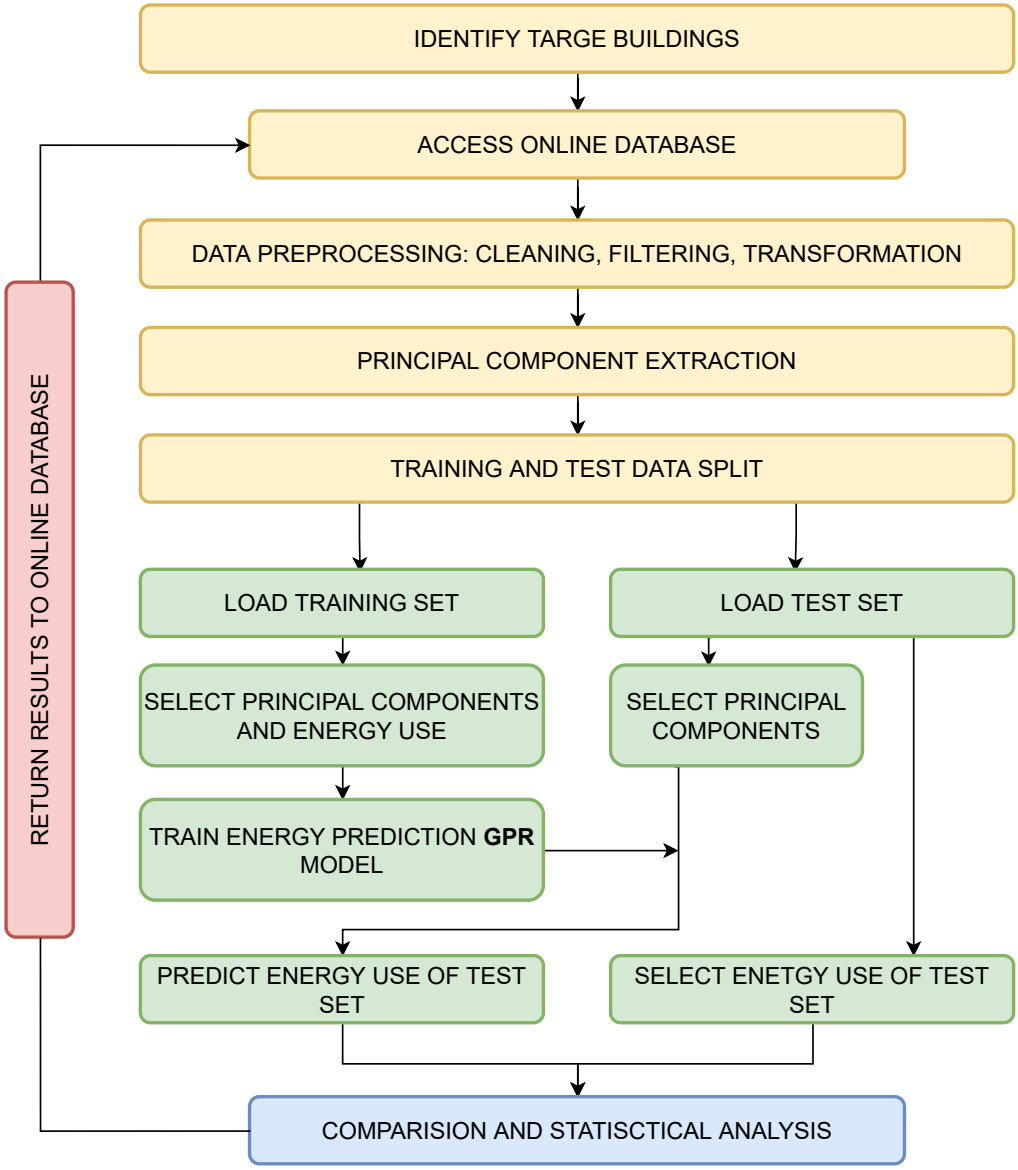



**Figure 3.** The problem of predicting energy usage in different kinds of building starts with identifying a target building. In the mentioned case, these are mostly offices and malls (buildings where people come regularly). The first component (yellow blocks) focuses on data processing. The algorithm connects to the base of the gathered data (historical data for training, real-time data for tests) and makes necessary operations like cleaning, filtering and component extraction. At the end, data are divided into testing and training sets. The green part of the scheme relates to the use of the GP model. Data are then used to train and test GP and at the end, to use it as a prediction of energy use. Results are then analysed and evaluated with several things, such as RMSE. The final prediction and other gathered results and analysis are finally saved to the database [29].

### 6. Decision Support for Bike-Sharing System

Bike-sharing systems are widely used across the globe especially in major cities. Their main use is in improving healthy and convenient transport for citizens. People can just rent a bike at any station and return it at any other. The significant problem with this system is the high usage of single stations which results in a lack of bikes or not enough free docks to return one. System operators try to solve it by reacting in real time to high demand and manually repositioning bikes between stations. This solution cannot properly handle this problem as it is always too late to make an action after the jammed station was observed [32]. To efficiently operate such a system, the authors of [32] proposed a complex model to predict citywide usage of bikes in the near future which allows the prevention of problems before they occur. The whole proposition of Hierarchical Consistency Prediction (HCP) consists of:

- Adaptive Transition Contraint (AdaTC)—which is a clustering algorithm used to cluster station into groups. This makes the rent and transition more regular within each cluster;
- Similarity-based Gaussian Process Regressor (SGPR)—which is a variation of GP used to predict how many bikes will be rented in different scale locations (e.g., single station, cluster etc.);
- General Least Square (GLS)—which is used to collectively improve obtained predictions;
- Transition based Inference (TINF)—which is designed to infer citywide bike return demand on predicted rent demands.

Figure 4 shows a whole framework which includes all mentioned parts. First, the AdaTC clusters' whole data based on station locations and bike usage patters. Then 3-level hierarchy model (HCP) is formulated. SGPR is used to make a check-out prediction based on available data and GLS improves it to final check-out form. At the end, the TINF method impact system is used with check-in prediction [32].

For us the most interesting part is the usage of Gaussian Process variation. The authors started with the use of the basic GP regression model, with the most common settings:

$$\mu(f) = 0 \tag{24}$$

$$\kappa(f_1, f_2) = a^2 \times \exp\left(-\sigma_4^{-2} \times (f_1 - f_2)^2\right), \tag{25}$$

where $\mu$ is mean, $\kappa$ covariance function and $a, \sigma$ are parameters. Then to a specific location the $M$-similar matrix training set in a target period $f_{t+1}$ is created:

$$H_{t+1} = \left\{ \left(f_1^{t+1}, x_1^{t+1}\right), \left(f_2^{t+1}, x_2^{t+1}\right), \ldots, \left(f_M^{t+1}, x_M^{t+1}\right) \right\}. \tag{26}$$

SGPR is an extension of GPR, which allows us to predict the value of function in $f_{t+1}$ in a certain location based on $H_{t+1}$. The authors also base some formulas on [1]. SGRP hyperparameters are learned by minimizing the total regression error on its validations set. Taking into account the specify of every locations, the authors learnt SGPR separately for each of them. Otherwise, the prediction accuracy could be negatively impacted [32]. In conclusion, GP was used in a slightly different form in order to adapt it to the problem, but overall usage was the same as usual. To sum up, the authors conducted a series of tests on real data which confirmed the effectiveness of their model, and they would like to generalize the model in the future.

In [33] we encounter a similar problem of bike-sharing difficulties connected to high-demand on stations. In addition to a general solution, they take into account the limiting effect of supply on demand. In order to accomplish that, the authors propose to implement censored likelihood which is able to train supply-aware models. In addition, the Gaussian Process model is used for time-series analysis of bike rentals, in order to calculate bike-sharing demand prediction. Both solutions are combined together to obtain satisfactory results.

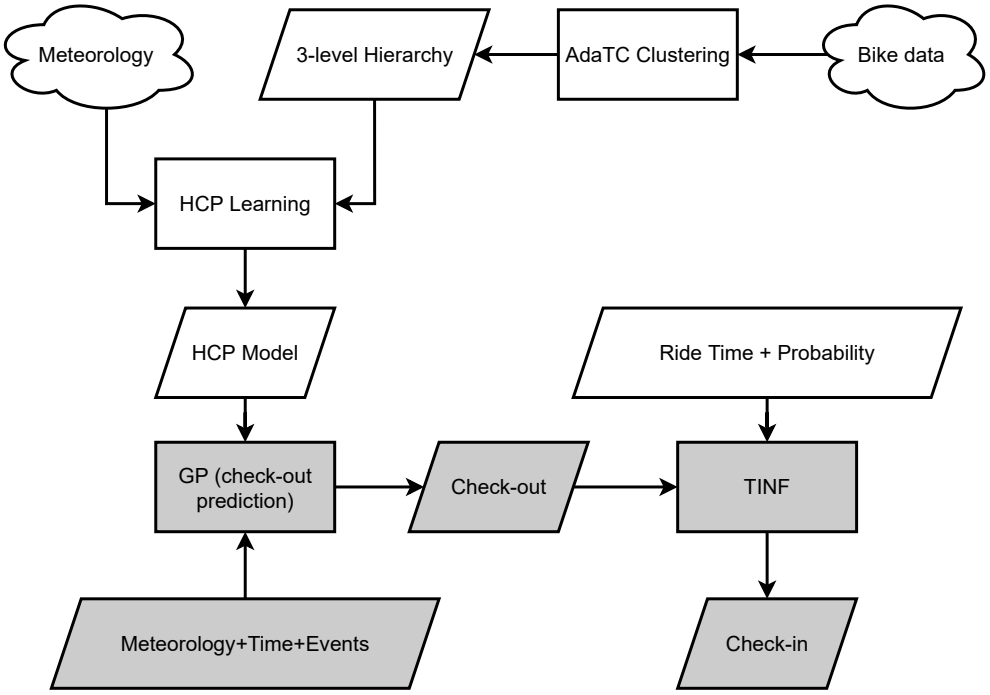



**Figure 4.** Prediction framework used for solving the bike-sharing system problem consists of various algorithms, which work together. The point of such a complex algorithm is to inform us ahead of time about the possibility of crowding the station and allow us to reposition bikes in advance. Bike data gathered from the whole city are firstly divided into clusters (by city segments, stations etc.) by AdaTC Clustering method. Based on this, a 3-level hierarchy is created and then meteorology data are added in order to create the Hierarchical Consistency Prediction (HCP) model. This with the combination of data about meteorology, time and particular events led to the creation of GP, the main purpose of which is to predict the check-outs of bike stations (count and time of renting a bike). With Transition based Inference (TINF) and ride time data, check-in times are inferred (count and time of returning a bike). Offline processes have a white background while online processes have greyed one [32].

The censoring problem was implemented, referring to the Tobias model [34], which can be simplified to: for each observation $i$, the model assumes that there is a latent (e.g., unobservable) variable $y_i^*$ and an observable censored realization $y_i$. The real demand is represented by $y_i^*$ while the available historical observations are represented by $y_i$. The simple form of the Tobit model is a dependency of variables on input data $x$ in a linear relationship with parameters $\beta$ with the addition of error $\varepsilon_i$, which we assume to be normally distributed (Formula (27)) [33].

$$y_i^* = \boldsymbol{\beta} \boldsymbol{x}_i + \varepsilon_i, \quad \varepsilon_i \sim N\left(0, \sigma^2\right). \tag{27}$$

The Gaussian Process, as mentioned before, is used for the prediction of demand. The authors note that, one of the most important values which comes from GP is not only the prediction on its own, but also its confidence interval. They also focus on three different kernels: Squared Exponential Kernel (SE), Periodic Kernel, Matérn Kernel, which are used in the project. Moreover, the optimization methods based on the maximizing likelihood, which was mentioned earlier in this review, were used to optimize hyperparameters [33].

What distinguishes this approach is combinations of GP and censoring problem, which gives us Censored Gaussian Processes (CGP). CGP have to deal with a censored regression problem and simultaneously be able to exploit GP flexibility to fit complex patters. It can be done by defining a censoring-aware likelihood within a GP framework, using it instead of standard GP likelihood. At the end, experiments on datasets from Donkey Republic (bike-sharing provider in Copenhagen) were conducted. Results showed that taking into account the limiting effect of supply on demand is very important for creating an unbiased predictive model of user demand behavior. Moreover, GP is considered a great way to solve a problem, because of its capability of representing an elegant framework which is able to work in a fully probabilistic way and to deal with uncertainty in predictions [33].

## 7. Gaussian Process in Tracking Problems

Gaussian Process Regression finds its usability in some kind of tracking enhancment. For instance in [35], GPR is used to reconstruct the trajectory from the recorded location samples. The whole idea base for the application was called SensTrack. The creators of the application claim that more and more smartphones are equipped with powerful sensors such as GPS, WiFi, acceleration and orientation sensors, so all of them can be used to enhance the tracking of devices, for instance in areas with no GPS signal such as indoors, and most importantly to reduce the energy consumption overall (because constant GPS usage drains the battery) [35].

SensTrack smartly chooses the location sensing method between WiFi and GPS, and also reduces the sampling rate by using data from other sensors: acceleration and orientation. Data collected this way are analyzed and filtered to reconstruct a user's original trajectory with GPR. Critical locations, decided by the sensor hints, which captures most of the key features of trajectory, were taken for a training set of GPR. As a test set—predicted locations between the successive but faraway location samples were used. The authors used standard settings of GPR: mean equals 0, for notation simplicity, and kernel functions of square exponential one. The real-world paths were examined and showed that SensTrack only needs 7% GPS samples of the naive approach and saves nearly 90% GPS activated time, while still reconstructing the original trajectory with high accuracy and coverage [35].

The authors of [35] also created an equivalent of SensTrack, the main purpose of which was for use in the Internet of Things (IoT) [36]. The IoT concept consists of Internet-connected devices with computing capability and have an ability to share data over the Internet wirelessly [37]. The main principals are the same, but the overall project here is in an earlier state. Most of the assumptions there are connected to GPR based on other work [38], which is in fact written with the use of machine-learning principals created by Rasmussen [1], so it concentrates all means of the machine-learning approach, as many of the other examples cited in this review.

The other kind of track problem, without actual use of GPS, is in [14]. Here, we encounter a problem of indoors tracking (which is the reason for the no-GPS approach). Contrary to other solutions, the authors did not base it on WiFi, Ultra Wide Band, Bluetooth nodes or similar. The whole 2D tracking algorithm is based on localization technologies, which use artificial light signals to substantiate positioning. Several methods to obtain such positioning exist: Received Signal Strength (RSS), Angle of Arrival (AoA), Time of Arrival (ToA) and Time Difference of Arrival (TDoA). The authors focus on an RSS based strategy with the intensity modulation of light sources for illumination as a localization system. This technique is called the Visible Light Positioning strategy (VLP), which consists of some Light Emitting Diodes (LED) for the purpose of fixed transmitter beacons and a photodetector as a receiver sensor [14].

The main parts of the project are the VLP Propagation model and the GP model. In this review we will mostly focus on how GP was handled here. A detailed description of the VLP model can be found in [14]. In general, the main purpose of it is to get an estimate of the coordinates of the location. The authors note that to solve the data oriented RSS-based VLP regression problem, there are many methods available currently, but using GP can handle problems very well, because of being very data-efficient and having an automatic optimization of hyperparameters and regularization. The authors use the predefined RBF kernel, with each entry of the RSS-base input and set to mean value 0. The hyperparameters of the RBF kernel were calculated by maximizing the log likelihood of data. Of course, this approach provides us with prediction, as well as an uncertainty for function values in new points [14].

As a validation of the created approach, the authors gathered data in slightly different scenarios (for instance: changing the LED installation high). Then the multilateration and GP approaches were tested and the performance of each was calculated in terms of $p90$ and $p95$ errors. Generally, GP turned out to outperform other methods by any means, in each scenario. Moreover, the author notes that GP has an advantage when it comes to a higher installation for data gathering, which can implicate the future use of such a solution in, for instance, warehouses [14].

## 8. Various Optimization Problems

Apparently, GP can be used in solving various problem stating with simple regression to complex prediction or identification. Last but not least, we would like to show some different, unrelated to each other applications of GP, which mostly aim to optimize different processes.

### *8.1. Cost Optimization*

First of all, in [39] we can see the optimization of the cost of creating the cloud data center. The problem consists of creating a project and a model of the mentioned cloud data center, taking into account things such as infrastructure, fault possibility etc. Data center main parts are: power, cooling, and IT. Each of these systems are described in detail. Then the project was divided into subsystems, which consist of simple blocks. Thanks to that it is possible to calculate the availability and to calculate the cost of the projected architecture. Then, the optimization is needed in order to find the best cost solution while taking into account the constraints. The authors used algorithms like Particle Swarm Optimization (PSO), Differential Evolution (DE) and Genetic Algorithms (GA) to achieve this goal [39].

Creating such a model, with many places, transitions, and tokens turns out to be extremely complex to compute. To reduce the complexity, and get rid of the heavy computing problem, the authors decided to use a surrogate model with the use of GP. Contrary to other proposed solutions such as neural networks, GP does not need a huge training dataset to be useful. They used a kind of GP called design and analysis of computer experiments (DACE), often referred to as kriging. In the original mean ,kriging is a statistical interpolation method based on GP to interpolate complex functions. Moreover, kriging is used to represent the input/output mapping of expensive computational models. It is now also connected with problems such as regression and identification [40]. A sampling strategy of Latin Hypercube Sampling (LHS) was adopted in this case. The authors evaluated the surrogate to ensure a low error in comparison to the original model. Thy created different scenarios with:

- exponential covariance, optimization via genetic algorithms with a maximum of 1000 generations;
- with similar configuration, only changing the covariance to Gaussian;
- with Gaussian covariance and Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization method with 1000 start points.

The results of creating the surrogate model were satisfying, with low RMSE. The whole problem was also tested, but these results focus mostly on choosing which one from three presented optimizing algorithm performs best [39].

### *8.2. Diesel Engines*

A completely different problem can be encountered in [41]. The main focus is on problems connected to diesel engines, more specifically post-injection models. Due to several factors, this method is already widespread and its importance will only grow. The main reason is ecology: the use of DPFs and the optimal selection of operating parameters is key to achieving a compromise between the pollutants produced and harmful NOx—we are dealing here with frequent DPF regeneration cycles, catalyst wear and cold start. All these factors contribute to the transfer of fuel to the oil. This causes the oil to dilute and, over time, makes it a necessity to be replaced (otherwise it will damage the engine). The described experiments are designed to find the optimal setting of parameters and types of post-injection of the engine in order to minimize oil dilution—this results in lower operating costs, etc. Additionally, the authors also take into account the automatic evaporation of the fuel [41].

The GP model is built on the basis of test data obtained on the basis of various scenarios of diesel engines. Then, on this basis, predictions are made as to the behavior of the oil in a given configuration. The authors look for correlations between factors and outcomes. The whole GP model was based on several sources [42–44], which exhaustively describe its application in this field. Studies also resulted in a comparison of GP to polynomial regression. It turns out that in engine related problems, GP outclasses polynomial regression. Overall, the results met the expectations of the authors, showing the relationships between various factors on the basis of several different sample tests. The GP was shown to be a good approach to the model formation rate of oil dilution [41].

### *8.3. Model Predictive Control*

Many control problems are based on designing a stabilized feedback with minimized performance criterion which satisfies constrains of the controls and states. The best solution would be to look for a closed solution for the feedback. Unfortunately, often closed solutions cannot be found analytically. A known approach to solving this problem is the repeated solution of the open-loop optimal control problem in a specific state. So the first part of this open-loop result is implemented and then the whole process is repeated. A control approach that utilized this strategy is called model predictive control (MPC) [45].

The GP based model of predictive control is the main purpose of the [46] article. The authors propose that kind of solution, mostly because of the measure of confidence offered by GP. That feature is very helpful in non-linear MPC (NMPC) design. They also note that GP is a probabilistic non-parametric black-box model that provides information about prediction uncertainties, which are difficult to evaluate appropriately in non-linear parametric models. The whole GP was set based on complex and different approaches, which are feasible in order to create a model for NMPC. A detailed description can be found in the article [46].

The whole idea was implemented on the gas–liquid separation plant example. The results of this approach were successful. The authors note that thanks to the confidence interval, the GP model can highlight input spaces, where prediction quality is poor, for instance due to data complexity or lack of it. This is a very attractive feature in control design. The only drawback of this solution is a computational complexity for industrial purposes. For the presented case (model contained about 1000 points to identify), the calculation of covariance matrix inversion was time consuming. The model barely managed to keep time constraints, but as the authors say, it should not detract from the overall value of GP usage [46].

*8.4. Image Processing*

The last presented application of GP is from the field of image processing. The whole idea in this example is to use GPR for image time series fap-filling and crop monitoring [47]. Basically, the optimization problem here is mainly about optimizing the GPR itself for a better performance in this field, but it is still a very good example of its general application. GP proves to be a very competitive solution in the field compared to the others available, especially as it has an uncertainty band. However, its computative complexity is problematic especially in the per-pixel approach mainly for GPR training rather than the fitting step. The authors proposed their own solution to get rid of this burden.

The authors demonstrate different ways to accomplish GPR fitting. They start with the base of other attempts to use GP for image processing which have been successful such as in [48–51] and many more. The authors focus mainly on describing the process of setting up the model for their case. A kernel function called Squared Exponential deserves special attention, as it is the most commonly employed covariance function in this area as it reflects the prior beliefs of the problem. SE is described in detail in the work, especially by means of its parameters. The process of optimizing hyperparameters is maximising the log marginal likelihood.

To mitigate the mentioned computational burden, the authors proposed to substitute the per-pixel optimization step with the creation of a cropland-based precalculations for the GPR hyperparameters. The whole process is shown in Figure 5 and consists of:

1. Crop type selection : for each crop type found, select random parcels;
2. Hyperparameter optimization: performed by assessing individually each pixel;
3. Hyperparameter average: mean of the previously trained hyperparameters for each crop type—also global average is computed;
4. Time series prediction: LAI-reconstructed time series computation with different GPR parameterizations;
5. Statistical analysis for performance comparison: evaluation of the performance of the different GPR models in terms of reconstruction with RSME indicator;
6. Phenological metrics extraction: analysis on how different GPR parametrizations affects estimations.

Overall, the results of this optimization are quite promising. The proposed method of the calculation of hyperparameters can be even 90 times faster, while being just slightly less accurate than non-reconstructed parameters. Moreover, the authors point out that GP is underutilised currently in time series processing and just some recent studies started to explore it in crop monitoring and as can be seen there is still room for GP to be researched in this field [47].

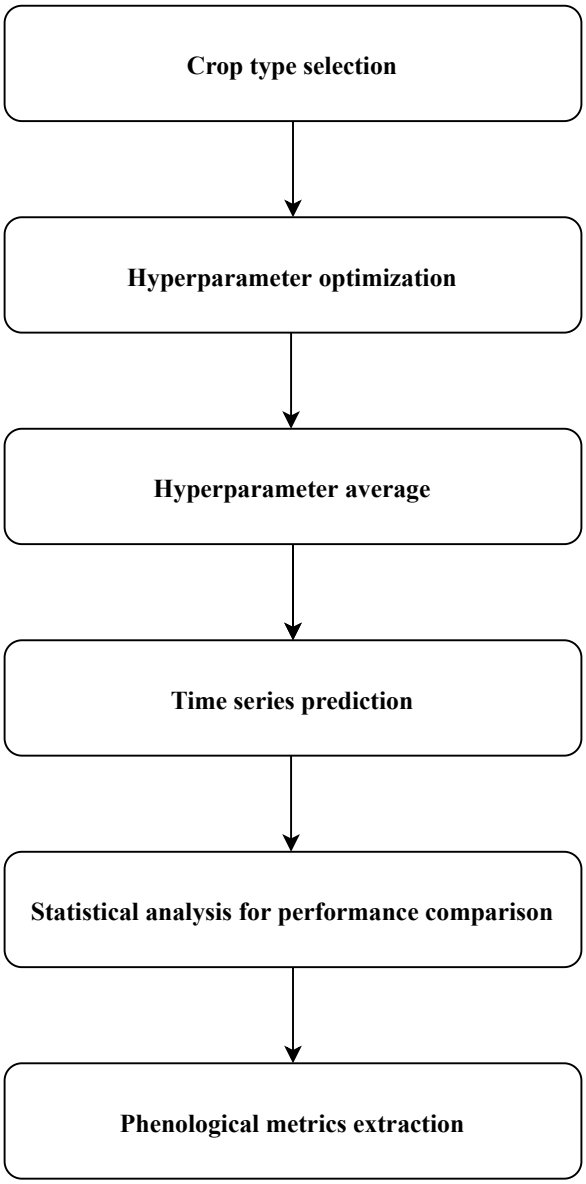



**Figure 5.** Figure presents an algorithm for image processing, which mitigates the computation burden connected with per-pixel computation for GPR hyperparameters. First, the 100 random parcels larger than 50 pixels are selected for each crop type found in the dataset. Then hyperparameters are optimally determined by assessing individually each pixel across time series. The third step is simply the calculation of the average of trained hyperparameters. Then time series were computed for different GPR model parameterizations using hyperparameters from steps 2 and 3. Finally, performance was evaluated in terms of original vs. reconstructed LAI time series and the results were analyzed in terms of how different parameters affected the estimation of phenological indicators [47].

## 9. Discussion and Future Direction

As research shows, a significant part of the use of GP in various fields focus on being used based on testing several covariance functions, selecting accurate ones and using automated optimizing of parameters consists of maximising the log likelihood (ML-II method). Of course, such an approach is not fundamentally wrong. Its primary advantage is the ease of use, high flexibility and speed of operation. As mentioned in examples, we can see that it can be easily set up with various environments (e.g., Matlab or python packages). Users can just fill the program with data and easily fit it with an automated

method of hyperparameters optimization. The only visible challenge here is the correct choice of kernel, almost always from an already defined base of covariance functions. With the high overall flexibility of GP usage, it can be a useful tool to solve a problem with the help of GP.

Besides that, there are more different approaches to GP. Some of them we mentioned in this review in the examples of use. In an article on the DT concept [20], the authors used another method of kernel choosing, which consisted of making a calculation on linear combinations of covariance functions and optimizing parameters in a different manner. One example with battery health parameters prediction [26] shows an extension of GP called SDGP, which is a "neural network patter" fix for some common GP problems known in this specific scenario; the same kind of solution as "extending" GP capabilities to the certain problem we saw in [33]—bike-sharing load prediction with censoring. The authors of [47] also managed to make GP more suitable for certain problems. Approaches such as this give overall better results not only in accuracy but also, e.g., in computing time.

Regardless of the chosen method, GP has a serious problem with the computation burden of covariance matrix inversion. The size of this matrix depends on the number of data considered in the problem. If there are too many of them, GP becomes unbearable while other methods become more appealing, especially from the machine-learning field while we encounter a lot of data. Of course, there are approximation methods, which helps to overcome this issue, but at an obvious cost of accuracy and sometimes unnecessarily complicate the problem, when one could just use methods other than GP. Moreover, solving GP analytically is often complicated, so there is always the necessity to use some sampling methods. Some optimizing methods have a problem with overfitting, low accuracy and bad generalization of use. Overcoming this challenge requires not only specific methods but also specific knowledge. In addition, it is worth noting that, if we cannot find a kernel matching our prior belief of the problem, it is really hard to create a new one. The task is so inconvenient that in most cases less appealing kernels are used, or the GP is dropped as a solution. These problems arise mostly when we are trying to solve some complex issues or they do not generally fit methods like GP, so everyone who tries GP in their research should be aware of that and if necessary reconsider using GP.

The whole paper shows a fragment of the GP world in control engineering, their practical application in specific areas and the ways of its implementation, as well as the results achieved. We can easily deduce two solid facts. First, GP is used in many fields related to control engineering, and with passing time, it is only gaining in popularity. Second, it is mainly used in the simplest and fully automated use that does not require complex modeling, knowledge and a lot of time. The second fact is where we can make further progress. Basically, there is nothing wrong with the most common use of GP. At a low cost, it shows whether the use of GP to solve the problem makes sense, or whether there are perhaps more appropriate algorithms. When such a problem seems to be ideal for the use of GP, such as detecting industrial anomalies or predicting battery life parameters, it is advisable to further develop and improve GP using available methods, e.g., by more accurate model creation, better hyperparamters optimization or better kernel matching (for instance by combining them). The development of GP does not stop on even presented approaches such as the Bayesian inference hierarchical model with MCMC sampling or approximation mentioned approximation methods. There are more methods, which develop values of Gaussian GP. For instance, we could use Gaussian Random Fields (GRF) in solving widely oriented control engineering problems, because GP is actually a one-dimensional GRF. In [52], the authors show the way of exploring this topic in spatial modelling. Another topic called Gaussian Process latent models, which generally attempt to capture hidden structure in high-dimensional data, can be an interesting direction as well. A more detailed review on GP latent models can be seen here [53]. Other interesting work related to the GP area can also be seen in [54,55].

Developing new strategies and applying ones which exist but are not currently commonly used in the field of control engineering is the direction which is highly suggested to be followed in terms of the future of GP.

## 10. Conclusions

Gaussian Processes are one of statistical modeling methods, which is rising in the popularity alongside Bayesian statistics in several recent years, despite that concepts of the GP were well-know to the scientist before. One reason of a such state is probably the computation burden of GP modeling connected mostly with the necessity of large covariance matrix inversion. The current state of technology allows us to use GP to a wider scale, but still more complex approaches require too much computation power than can be provided. For such cases, approximation methods of GP were developed. It can be noticed that most of the reviewed cases are still handing GP solutions with ML-II methods. However, it has to be said that more and more scientists are encouraged to develop more advanced approaches to solving problems with GP, even creating their own methods.

This review shows that GP has enormous potential when it comes to real-word control engineering applications. We can easily find a GP based solution in fields of anomaly detection (especially in industry), battery parameters and other electricity related prediction, image processing, expenditure planning as well as support for bicycle or location systems and others. We can classify the primary uses of GP as regression, identification, prediction, and classification. A certain correlation was noticeable: the more research examples we found in a given field, the more modified GP methods were encountered. This is probably due to the discovery of the potential of GP for a certain field and further willingness to improve these algorithms by other means.

The methods that can build more and more accurate GP models such as the hierarchical model with MCMC sampling involve following the direction to increase the usability of GP. With too complex problems in terms of computation, some approximation should be involved. The first step for the future of GP is to involve more complex methods (different from the mainstream, not only those described here) for resolving problems, which will give us an advantage in solutions over the currently dominant methods. The major result of this review paper is an orientation on the current state of use in the areas and methods of using GP in related fields with broadly understood control engineering and providing a future direction and tips on where to develop the field further.

**Author Contributions:** Conceptualization, A.D. and J.B.; methodology, J.B.; resources, J.B.; writing—original draft preparation, A.D.; writing—review and editing, A.D. and J.B.; visualization, A.D.; supervision, J.B.; project administration, J.B.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by AGH's Research University Excellence Initiative under project "Interpretable methods of process diagnosis using statistics and machine learning" and by Polish National Science Centre project "Process Fault Prediction and Detection" contract no. UMO-2021/41/B/ST7/03851.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| GP | Gaussian Process |
| GPR | Gaussian Process regression |
| PCA | principal component analysis |
| PLS | partial least squares |
| DT | Digital Twin |
| SoH | State of Health |
| RUL | Remaining Useful Life |
| GBR | gradient boosting regression |
| RBF | Radial Basis Function |
| RQ | Rotational Quadratic |
| ESS | Exp Sine Squared |
| SGP | spacial Gaussian Process |
| DGPR | deep Gaussian Process Regressio |
| HCP | Hierarchical Consistency Prediction |
| AdaTC | Adaptive Transition Contraint |
| SGPR | Similarity-based Gaussian Process Regresso |
| GLS | General Least Squar |
| TINF | Transition based Inference |
| CGP | Censored Gaussian Process |
| RSS | Received Signal Strengt |
| AoA | Angle of Arrival |
| ToA | Time of Arrival |
| TDoA | Time Difference of Arrival |
| VLP | Visible Light Positioning strategy |
| RMSE | Root Mean Squared Error |
| NMBE | Normalized Mean Bias Error |
| WTPC | wind turbine power curve |
| KDE | kernel density estimation |
| CKDE | conditional kernel density estimation |
| RVM | relevance vector machine |
| SVM | support vector machine |
| MPC | model predictive control |
| GA | Genetic Algorithms |
| PSO | Particle Swarm Optimization |
| DE | Differential Evolution |
| DACE | design and analysis of computer experiments |
| MCMC | Markov chain Monte Carlo |
| IoT | Internet of Things |

## References

1. Rasmussen, C.; Williams, C.K.I. *Gaussian Processes in Machine Learning*; MIT Press: Cambridge, MA, USA, 2006.
2. Koriyama, T. An introduction of Gaussian processes and deep Gaussian processes and their applications to speech processing. *Acoust. Sci. Technol.* **2020**, *41*, 457–464. [CrossRef]
3. Gonzalvez, J.; Lezmi, E.; Roncalli, T.; Xu, J. Financial Applications of Gaussian Processes and Bayesian Optimization. *arXiv* **2019**, arXiv:1903.04841.
4. Deringer, V.L.; Bartók, A.P.; Bernstein, N.; Wilkins, D.M.; Ceriotti, M.; Csányi, G. Gaussian Process Regression for Materials and Molecules. *Chem. Rev.* **2021**, *121*, 10073–10141. [CrossRef] [PubMed]
5. Kanagawa, M.; Hennig, P.; Sejdinovic, D.; Sriperumbudur, B.K. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. *arXiv* **2018**, arXiv:1805.08845v1.
6. Nguyen, D.T.; Filippone, M. *Exact Gaussian Process Regression with Distributed Computations*; ACM: New York, NY, USA, 2019; pp. 1286–1295. [CrossRef]
7. Baranowski, J.; Dudek, A.; Mularczyk, R. Transient Anomaly Detection Using Gaussian Process Depth Analysis. In Proceedings of the 2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland, 23–26 August 2021; pp. 221–226. [CrossRef]

8. Liu, Y.; Noomen, R.; Visser, P. A Gravity Assist Mapping for the Circular Restricted Three-Body Problem using Gaussian Processes. *Adv. Space Res.* **2021**, *68*, 2488–2500. [CrossRef]
9. Riutort-Mayol, G.; Bürkner, P.C.; Andersen, M.R.; Solin, A.; Vehtari, A. Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming. *arXiv* **2020**, arXiv:2004.11408.
10. Davis, R.A. Encyclopedia of Environmetrics, Gaussian Process. In *Encyclopedia of Environmetrics*; American Cancer Society: Atlanta, GA, USA, 2006. [CrossRef]
11. Simpson, D. Un Garçon Pas Comme les Autres (Bayes): Yes But What Is a Gaussian Process? Or, Once, Twice, Three Times a Definition; or A Descent into Madness. Available online: https://dansblog.netlify.app/posts/2021-11-03-yes-but-what-is-a-gaussian-process-or-once-twice-three-times-a-definition-or-a-descent-into-madness/ (accessed on 30 March 2022).
12. Garnett, R. *Bayesian Optimization*; Cambridge University Press: Cambridge, UK, 2022; *in preparation*.
13. Blum, M.; Riedmiller, M. Optimization of Gaussian Process Hyperparameters Using Rprop. In Proceedings of the ESANN 2013 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Belgium, Bruges, 24–26 April 2013; pp. 339–344.
14. Raes, W.; Dhaene, T.; Stevens, N. On The Usage of Gaussian Processes for Visible Light Positioning With Real Radiation Patterns. In Proceedings of the 2021 17th International Symposium on Wireless Communication Systems (ISWCS), Berlin, Germany, 6–9 September 2021; pp. 1–6. [CrossRef]
15. Mackay, D.J.C. Introduction to Gaussian Processes. In *Neural Networks and Machine Learning*; Bishop, C.M., Ed.; Springer: Berlin, Germany, 1988.
16. Bottou, L.; Chapelle, O.; DeCoste, D.; Weston, J. Approximation Methods for Gaussian Process Regression. In *Large-Scale Kernel Machines*; MIT Press: Cambridge, MA, USA, 2007; pp. 203–223.
17. Quiñonero-candela, J.; Rasmussen, C.E.; Herbrich, R. A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.* **2005**, *6*, 2005.
18. Solin, A.; Särkkä, S. Hilbert Space Methods for Reduced-Rank Gaussian Process Regression. *arXiv* **2014**, arXiv:1401.5508.
19. Wang, B.; Mao, Z. Outlier detection based on Gaussian process with application to industrial processes. *Appl. Soft Comput.* **2019**, *76*, 505–516. [CrossRef]
20. Trauer, J.; Pfingstl, S.; Finsterer, M.; Zimmermann, M. Improving Production Efficiency with a Digital Twin Based on Anomaly Detection. *Sustainability* **2021**, *13*, 155. [CrossRef]
21. Aye, S.; Heyns, P. An integrated Gaussian process regression for prediction of remaining useful life of slow speed bearings based on acoustic emission. *Mech. Syst. Signal Process.* **2017**, *84*, 485–498. [CrossRef]
22. Pfingstl, S.; Rios, J.I.; Baier, H.; Zimmermann, M. Predicting Crack Growth and Fatigue Life with Surrogate Models. *arXiv* **2020**, arXiv:2008.02324.
23. Pfingstl, S.; Zimmermann, M. On integrating prior knowledge into Gaussian processes for prognostic health monitoring. *Mech. Syst. Signal Process.* **2022**, *171*, 108917. [CrossRef]
24. Garay, F.; Huaman, W.; Vargas-Machuca, J. State of health diagnostic and remain useful life prognostic for lithium-ion battery by combining multi-kernel in Gaussian process regression. In Proceedings of the 2021 IEEE XXVIII International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Lima, Peru, 5–7 August 2021; pp. 1–4. [CrossRef]
25. Richardson, R.R.; Osborne, M.A.; Howey, D.A. Battery health prediction under generalized conditions using a Gaussian process transition model. *J. Energy Storage* **2019**, *23*, 320–328. [CrossRef]
26. Tagade, P.; Hariharan, K.S.; Ramachandran, S.; Khandelwal, A.; Naha, A.; Kolake, S.M.; Han, S.H. Deep Gaussian process regression for lithium-ion battery health prognosis and degradation mode diagnosis. *J. Power Sources* **2020**, *445*, 227281. [CrossRef]
27. Greenbank, S.; Howey, D. Automated feature selection for data-driven models of rapid battery capacity fade and end of life. *arXiv* **2021**, arXiv:2101.04440.
28. Damianou, A.; Lawrence, N.D. Deep Gaussian Processes. *Artif. Intell. Stat.* **2013**, *31*, 207–215.
29. Zeng, A.; Ho, H.; Yu, Y. Prediction of building electricity usage using Gaussian Process Regression. *J. Build. Eng.* **2020**, *28*, 101054. [CrossRef]
30. Chu, J.; Yuan, L.; Hu, Y.; Pan, L. Comparative Analysis for Interval Modeling Algorithms of Wind Turbine Power Curve. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *569*, 052028. [CrossRef]
31. Pandit, R.K.; Infield, D. Using Gaussian process theory for wind turbine power curve analysis with emphasis on the confidence intervals. In Proceedings of the 2017 6th International Conference on Clean Electrical Power (ICCEP), Santa Margherita Ligure, Italy, 27–29 June 2017; pp. 744–749. [CrossRef]
32. Li, Y.; Zheng, Y. Citywide Bike Usage Prediction in a Bike-Sharing System. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1079–1091. [CrossRef]
33. Gammelli, D.; Rodrigues, F.; Pacino, D.; Kurtaran, H.A.; Pereira, F.C. A Machine Learning Approach to Censored Bike-Sharing Demand Modeling. *Transp. Res. Board Annu. Meet. Proc.* **2020**, *2020*, 1.
34. Tobin, J. Estimation of Relationships for Limited Dependent Variables. *Econometrica* **1958**, *26*, 24–36. [CrossRef]
35. Zhang, L.; Liu, J.; Jiang, H.; Guan, Y. SensTrack: Energy-Efficient Location Tracking With Smartphone Sensors. *IEEE Sens. J.* **2013**, *13*, 3775–3784. [CrossRef]
36. Zhang, L.; Liu, J.; Jiang, H. Energy-efficient location tracking with smartphones for IoT. In Proceedings of the SENSORS, 2012 IEEE, Taipei, Taiwan, 28–31 October 2012; pp. 1–4. [CrossRef]

37. Islam, M.; Nooruddin, S.; Karray, F.; Muhammad, G. Internet of Things Device Capabilities, Architectures, Protocols, and Smart Applications in Healthcare Domain: A Review. *arXiv* **2022**, arXiv:2204.05921.
38. Ma, L.; Liu, J.; Sun, L.; Karimi, O.B. The Trajectory Exposure Problem in Location-Aware Mobile Networking. In Proceedings of the 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, Spain, 17–22 October 2011; pp. 7–12. [CrossRef]
39. Gonçalves, G.; Gomes, D.; Leoni, G.; Rosendo, D.; Moreira, A.; Kelner, J.; Sadok, D.; Endo, P. Optimizing the Cloud Data Center Availability Empowered by Surrogate Models. In Proceedings of the Hawaii International Conference on System Sciences, Maui, HI, USA, 7–10 January 2020. [CrossRef]
40. Lataniotis, C.; Marelli, S.; Sudret, B. UQLAB User Manual—Kriging (Gaussian Process Modelling). 2015. Available online: https://www.uqlab.com/kriging-user-manual (accessed on 30 March 2022).
41. Gönül, M.; Kutlar, O.A.; Calik, A.T.; Orcun Parlak, F. Prediction of oil dilution formation rate due to post injections in diesel engines by using Gaussian process. *Fuel* **2021**, *305*, 121608. [CrossRef]
42. Berger, B.; Rauscher, F.; Lohmann, B. Analysing Gaussian Processes for Stationary Black-Box Combustion Engine Modelling. *IFAC Proc. Vol.* **2011**, *44*, 10633–10640. [CrossRef]
43. Gutjahr, T.; Kleinegraeber, H.; Huber, T.; Kruse, T. Advanced Statistical System Identification in ECU-Development and Optimization. In Proceedings of the SAE 2015 Commercial Vehicle Engineering Congress, Chicago, IL, USA, 6–8 October 2015. [CrossRef]
44. Berger, B.; Rauscher, F. Robust Gaussian Process Modelling for Engine Calibration. *IFAC Proc. Vol.* **2012**, *45*, 159–164. [CrossRef]
45. Allgöwer, F.; Findeisen, R.; Nagy, Z. Nonlinear model predictive control: From theory to application. *J. Chin. Inst. Chem. Engrs* **2004**, *35*, 299–315.
46. Likar, B.; Kocijan, J. Predictive control of a gas–liquid separation plant based on a Gaussian process model. *Comput. Chem. Eng.* **2007**, *31*, 142–152. [CrossRef]
47. Belda, S.; Pipia, L.; Morcillo-Pallarés, P.; Verrelst, J. Optimizing Gaussian Process Regression for Image Time Series Gap-Filling and Crop Monitoring. *Agronomy* **2020**, *10*, 618. [CrossRef]
48. Verrelst, J.; Rivera, J.P.; Moreno, J.; Camps-Valls, G. Gaussian processes uncertainty estimates in experimental Sentinel-2 LAI and leaf chlorophyll content retrieval. *ISPRS J. Photogramm. Remote Sens.* **2013**, *86*, 157–167. [CrossRef]
49. Verrelst, J.; Alonso, L.; Camps-Valls, G.; Delegido, J.; Moreno, J. Retrieval of Vegetation Biophysical Parameters Using Gaussian Process Techniques. *IEEE Trans. Geosci. Remote Sens.* **2012**, *50*, 1832–1843. [CrossRef]
50. Verrelst, J.; Alonso, L.; Rivera Caicedo, J.P.; Moreno, J.; Camps-Valls, G. Gaussian Process Retrieval of Chlorophyll Content From Imaging Spectroscopy Data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2013**, *6*, 867–874. [CrossRef]
51. Moore, C.; Chua, A.; Berry, C.; Gair, J. Fast methods for training Gaussian processes on large datasets. *R. Soc. Open Sci.* **2016**, *3*, 160125. [CrossRef] [PubMed]
52. Fuglstad, G.A.; Lindgren, F.; Simpson, D.; Rue, H. Exploring a new class of non-stationary spatial gaussian random fields with varying local anisotropy. *Stat. Sin.* **2015**, *25*, 115–133. [CrossRef]
53. Li, P.; Chen, S. A review on Gaussian Process Latent Variable Models. *CAAI Trans. Intell. Technol.* **2016**, *1*, 366–376. [CrossRef]
54. Ihler, A.; Fischer, J., III; Willsky, A. Loopy Belief Propagation: Convergence and Effects of Message Errors. *J. Mach. Learn. Res.* **2005**, *6*, 905–936.
55. Sui, T.; Marelli, D.; Fu, M.; Lu, R. Accuracy Analysis for Distributed Weighted Least-Squares Estimation in Finite Steps and Loopy Networks. *Automatica* **2018**, *97*, 82–91. [CrossRef]

*electronics*

MDPI

*Article*

# Efficient Gaussian Process Calculations Using Chebyshev Nodes and Fast Fourier Transform

**Adrian Dudek and Jerzy Baranowski ***

Department of Automatic Control & Robotics, AGH University of Science & Technology, 30-059 Kraków, Poland; addudek@agh.edu.pl
* Correspondence: jb@agh.edu.pl

**Abstract:** Gaussian processes have gained popularity in contemporary solutions for mathematical modeling problems, particularly in cases involving complex and challenging-to-model scenarios or instances with a general lack of data. Therefore, they often serve as generative models for data, for example, in classification problems. However, a common problem in the application of Gaussian processes is their computational complexity. To address this challenge, sparse methods are frequently employed, involving a reduction in the computational domain. In this study, we propose an innovative computational approach for Gaussian processes. Our method revolves around selecting a computation domain based on Chebyshev nodes, with the optimal number of nodes determined by minimizing the degree of the Chebyshev series, while ensuring meaningful coefficients derived from function values at the Chebyshev nodes with fast Fourier transform. This approach not only facilitates a reduction in computation time but also provides a means to reconstruct the original function using the functional series. We conducted experiments using two computational methods for Gaussian processes: Markov chain Monte Carlo and integrated nested Laplace approximation. The results demonstrate a significant reduction in computation time, thereby motivating further development of the proposed algorithm.

**Keywords:** gaussian process; Chebyshev polynomials; fast Fourier transform; integrated nested Laplace approximations; Markov chain Monte Carlo



**Citation:** Dudek, A.; Baranowski, J. Efficient Gaussian Process Calculations Using Chebyshev Nodes and Fast Fourier Transform. *Electronics* **2024**, *13*, 2136. https://doi.org/10.3390/electronics13112136

Academic Editor: Olivier Sename

Received: 20 February 2024
Revised: 17 May 2024
Accepted: 24 May 2024
Published: 30 May 2024



## 1. Introduction

Gaussian processes (GPs) have gained significant popularity in the fields of machine learning and statistics, owing to their remarkable adaptability and capacity to capture intricate, non-linear associations. These non-parametric Bayesian models empower the inference of complex, non-linear functions from data while furnishing valuable uncertainty assessments. Essentially, GPs embody a stochastic process that posits a probability distribution over functions, ensuring that any finite collection of function values conforms to a joint Gaussian distribution.

In essence, a GP lays down a prior distribution over functions, subsequently refined through Bayesian inference as more data become available. Notably, GPs exhibit the unique capability of accommodating irregularly spaced and noisy data, as well as capturing inherent correlation structures within the data [1–3].

The utility of GP spans an array of tasks, encompassing regression [4–6], classification [7,8], and optimization [9,10]. These versatile models have found successful applications in diverse domains, including support for bike-sharing systems [11], computer vision [12], and even diesel engine optimization [13]. A prominent feature of GP is its ability to yield probabilistic predictions and effectively account for data noise, as previously noted.

However, the predominant drawback of employing GPs lies in their computational demands [3]. Considerable efforts have been dedicated to enhancing the computational efficiency of GPs, particularly when dealing with large datasets [14,15]. Prominent techniques

in this regard include sparse approximation methods and scalable algorithms, which aim to make GPs more accessible and computationally tractable [16,17].

In recent years, the integrated nested Laplace approximation (INLA) framework has emerged as a powerful and efficient approach for modeling GPs in various statistical and machine learning applications. INLA combines elements of Bayesian modeling with numerical approximations, offering a computationally tractable alternative to traditional Markov chain Monte Carlo (MCMC) methods for estimating GP parameters and making predictions [18]. INLA excels in providing accurate and rapid inference for GPs, particularly when dealing with large datasets or complex models [19]. It leverages a combination of deterministic numerical integration techniques and Laplace approximations to approximate the posterior distribution of GP parameters, such as the length scale and amplitude hyperparameters. This approach significantly reduces the computational burden associated with GP modeling, making it feasible to work with extensive datasets and intricate models [20].

Less attention is given to improving the efficiency of using GPs as generative models. In general, GPs are utilized as generative models within the larger algorithm framework. For instance, the authors of [21] demonstrate the technique for embedding unlabeled categorical data into a continuous space using GP. As a probabilistic generative model, it is capable of estimating densities. They present a generative classification model that operates as a supervised approach for labeled categorical data. Moreover, the authors of [22] developed a Gaussian process density sampler, an exchangeable model for nonparametric Bayesian density estimation. This model generates independent samples from a transformed function, derived from a GP prior. Their approach facilitates the inference of an unknown density from data through MCMC methods, which gives samples from both the posterior distribution over density functions and the predictive distribution in the data space. A good representation of the GP generative model, which was optimized through approximation, is presented in [23], where the authors address the problem of Koopman mode decomposition. Their generative GP model allows for the concurrent estimation of specific quantities and latent variables controlled by an unidentified dynamical system. Moreover, they propose an effective method for parameter estimation within the model, which utilizes low-rank approximations of covariance matrices. The optimization of the generative model in the mentioned case primarily relies on reducing the dimension of the covariance matrix, a commonly employed method that facilitates computation time reduction. An underexplored aspect of such optimizations involves decreasing the dimensions of the mentioned matrix in a manner that additionally provides further information for the generative model. Our proposed approach for selecting computation points for GP is based on the premise of not only reducing computation time but also enhancing the generative model's performance through the application of output in the form of functional series.

In this paper, we aim to explore the feasibility and efficiency of leveraging the Chebyshev property for efficient GP calculations. Our main contributions to this paper are as follows:

- **Selecting computational points for GP by using the properties of Chebyshev nodes.** The proposed approach seeks to mitigate the computational challenges associated with GPs by reducing the number of computation points and adopting a specific method to gather information for generating functions with series representation
- **An algorithm output in the form of a functional series, achieving low computational complexity with minimal loss of accuracy.** Normally, it is difficult to ensure the series representation, but our approach allows us to address this challenge with the use of fast Fourier transform (FFT) formulas to convert function values at Chebyshev points into coefficients of the Chebyshev series. This can be done with minimal loss of accuracy (1 bit) while facilitating effective interpolation of the series.
- **An algorithm that is especially susceptible to heavy computational methods.** Alleviating computational burden by reducing the covariance matrix within Chebyshev node calculations can enable the use of high-cost computational methods like MCMC sampling on a wider scale.

- **Improving the usability of GP generative models.** Ensuring the function series output with FFT conversion, with low cost and accuracy loss can lead to overall improved generative model capabilities, for example, in terms of classification.

The rest of the paper is organized as follows. We begin by presenting the fundamental theory behind GPs, generative models, and Chebyshev interpolation that we use in our solution as well as a description of computational methods used in the experiment section (MCMC and INLA framework). Subsequently, we detail the experiments conducted and the application of the proposed methodology to GP computations. Finally, we conclude with a discussion and summary of our findings.

## 2. Materials and Methods

This section of the article is dedicated to explaining the fundamental theory behind Gaussian processes (GPs), including their covariance functions and practical applications. Furthermore, we describe the theory and use of generative models in the context of GPs. We also describe generative models to highlight the application context of the method. Moreover, we focus on the components of Chebyshev interpolation, which we employ in our proposed process. This starts with a description of Chebyshev nodes and extends to the convergence of functional series, detailing the specifics of the algorithm for converting values at Chebyshev nodes into Chebyshev series coefficients using FFT. It also describes the computational methods used in the experimental section, such as Markov chain Monte Carlo (MCMC) methods and the integrated nested Laplace approximation (INLA) framework, which are employed for efficient Bayesian inference. Additionally, this section provides an overview of the algorithm based on the aforementioned theory, explaining how these elements are bound to address the problem.

### *2.1. Gaussian Process*

We need to provide a concise definition of GPs and explore their fundamental properties. A widely accepted and succinct definition of a GP comes from [2] and can be summarized as follows:

> *A Gaussian process is a collection of random variables, any finite number of which jointly follow a Gaussian distribution.*

To define a GP mathematically, let us introduce the mean function $m(x)$ and the covariance function $k(x, x')$ for a real process $f(x)$ as follows:

$$m(x) = \mathbb{E}[f(x)] \tag{1}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{2}$$

With these definitions, we can formally express a GP as follows:

$$f(x) \sim GP(m(x), k(x, x')) \tag{3}$$

The mean and covariance functions provide a complete characterization of the GP prior, describing the expected behavior of the function, $f$.

To define GP in another way, we can consider the following set of observations: $\boldsymbol{f} = \{f(\boldsymbol{x}_n)\}_{n=1}^{N}$; then the prior distribution of $\boldsymbol{f}$ follows a multivariate Gaussian distribution: $\boldsymbol{f} \sim \text{Normal}(\boldsymbol{\mu}, \boldsymbol{K})$, where $\boldsymbol{\mu} = \{\mu(\boldsymbol{x}_n)\}_{n=1}^{N}$, and $\boldsymbol{K}$ is the covariance matrix with elements $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The joint distribution of $f$ and a new variable $\tilde{f}$, assuming $\boldsymbol{\mu} = 0$, is expressed as follows:

$$p(\boldsymbol{f}, \tilde{f}) = \text{Normal}\left( \begin{bmatrix} \boldsymbol{f} \\ \tilde{f} \end{bmatrix} \middle| \boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{f,f} & \boldsymbol{k}_{f,\tilde{f}} \\ \boldsymbol{k}_{\tilde{f},f} & k_{\tilde{f},\tilde{f}} \end{bmatrix} \right) \tag{4}$$

where $\boldsymbol{k}_{f,\tilde{f}}$ represents the covariance between $\boldsymbol{f}$ and $\tilde{f}$ (creating a covariance matrix), and $k_{\tilde{f},\tilde{f}}$ is the prior variance of $\tilde{f}$ [2].

In a broader context, GPs are stochastic processes commonly used for modeling data observed over time, space, or both [1]. They can be viewed as generalizations of normal probability distributions, where each random variable (scalar or vector in the case of multivariate distributions) is described by a GP. Formally, a GP on a set $\mathcal{T}$ is a collection of random variables indexed by a continuous variable, denoted as $\{f(t) : t \in \mathcal{T}\}$, with the property that any finite subset of these random variables follows a multivariate Gaussian distribution [24,25].

To fully specify a GP, one needs to define the mean (often set to "0" for simplicity) and covariance functions. However, variations in the mean function can be considered for interpretability or prior specification purposes [1,2,26]. The covariance function, also known as the kernel function, quantifies the similarity between data points and is typically chosen from a set of predefined functions [27]. While choosing the appropriate covariance function is crucial, any function generating a positive definite covariance matrix can be used [28]. Nevertheless, creating new covariance functions that are both mathematically and practically useful can be challenging.

In this research, our attention is primarily directed toward exploring the intricacies and applications of two distinct kernels: the radial basis function (RBF) and the Matérn kernel. These kernels were chosen for their unique properties and relevance in the field of study, offering us the opportunity to delve deeper into their theoretical foundations and practical implementations. The RBF kernel is defined as follows:

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right) \tag{5}$$

where $l$ represents the characteristic length scale and $d(\cdot, \cdot)$ is the Euclidean distance. The RBF kernel's key property is its dependence, primarily on the distance from the specified point. In the case of a twice differentiable function, we use a kernel from the Matérn family. The Matérn kernel is a kind of generalization of the RBF function. The general formula for Matérn covariance functions is given by (6):

$$k(x_i, x_j) = \frac{1}{\Gamma(v)2^{v-1}}\left(\frac{\sqrt{2v}}{l}d(x_i, x_j)\right)^{v} K_v\left(\frac{\sqrt{2v}}{l}d(x_i, x_j)\right) \tag{6}$$

where $K_v$ is a modified Bessel function, $l$ is a length-scale parameter and $v$ is a parameter that allows changing smoothness of the function, which gives an ability to flexibly control the function in relation to the one we want to model. Of particular note are the frequently used parameter, $v$, with values of $3/2$ used for learning functions that are at least once differentiable, and $5/2$ for functions that are at least twice differentiable [2].

The kernel choice plays a pivotal role in determining the regularity of the function from the GP. Specifically, the Matérn kernel exhibits a notable characteristic where it is $k$ times differentiable. This differentiability property is integral in defining the smoothness of the function modeled by the Gaussian process. A higher degree of differentiability implies a smoother function, which can be crucial in various applications where the smoothness of the function has significant implications on the accuracy and reliability of the model's predictions or interpretations [2].

GPs play a significant role in the advancement of modern technologies and sciences, where they are used for prediction, optimization, or classification [3]. In practice, they are useful, for instance, in industrial anomaly detection [29], and crucial for ensuring production quality and continuity. GPs predict battery state parameters [30], which are relevant in electric vehicles and energy management [4]. In bike-sharing systems, they optimize bike allocation, enhancing efficiency [31]. Furthermore, GPs are applied in tracking and location, especially where GPSs fail [32]. In optimization areas, they aid in resource management, like in data center designs [9]. These processes are key in data analysis and statistical model-

ing, used across various fields, from image processing [12] to biomedical data analysis [33], demonstrating their versatility and importance in contemporary applications.

### *2.2. Generative Models*

A generative model is a type of statistical algorithm designed to generate new data similar to the data it was trained on, but not identical. These models learn the probability distribution of the training data, allowing them to create new instances that retain the statistical characteristics of the training set. Generative models are widely used in various fields such as natural language processing, image generation, speech synthesis, and others, where they can create realistic text, images, sounds, or other types of data [34].

In the context of GPs, we utilize function sampling from posterior distributions, enabling statistical inference based on available data. This method relies on data influencing the distribution of hyperparameters, allowing the generation of new data samples and creating a generative model [2]. An example of a generative model can be found in [29], where the described instance of diagnosing malfunctions in industrial processes. In this case, the GP model is trained with measurement data from both normal and healthy operation of equipment as well as from moments when faults occur. The model then becomes capable of generating more time series data, which consequently leads to the ability to classify new situations based on data depth. This approach exemplifies the application of GPs in predictive maintenance, where they assist in interpreting complex data patterns to identify potential issues before they become critical failures, enhancing both the efficiency and safety of industrial operations. Another example can be found in [35]. In this problem, we use generative models within a Gaussian Mixture to classify the faulty or healthy states of induction motor startups.

GPs provide a deeper understanding of data relationships, which is particularly useful in estimating hyperparameters and calculating posterior distributions [2]. In function modeling, it is crucial to focus not only on values at measurement points but also on the overall function trajectory. An important aspect of GPs is that we sample a set of points, not the entire function. The problems we are dealing with involve how to choose specific points at which to calculate the GP and how to generate functions from the model. A simple rule of thumb states that the more data points we have information on, the easier and better the interpolation will be, regardless of the method chosen. However, in the case of GP, significantly increasing the number of computational points (random variables) can lead to computational burdens [3]. Therefore, we must strive for a robust method of interpolating functions while maintaining as few computational points as possible for GP.

### *2.3. Chebyshev Interpolation*

Our proposed solution is to generate functions through a GP model in the form of a functional series, an example of which is the Chebyshev series. GP generates a point from a function. We want to transform those points into a Chebyshev series. Fundamentally, obtaining its coefficients is not a simple task, but each polynomial can be uniquely expressed as a finite Chebyshev series. When the polynomial is defined by its values at Chebyshev points, there is a unique linear relationship between these values and the coefficients of the Chebyshev expansion. This relationship allows for an efficient mapping, executable in $O(nlogn)$ operations using the FFT. This approach offers a computationally effective way of handling polynomial expressions and transformations [36].

In the context of generative modeling, where new functions are generated, the application of the Chebyshev series becomes particularly relevant. In such a case, it is very useful that the Chebyshev series are known for their convergence properties, particularly toward differentiable functions that are generated by the model. Moreover, the degree of approximation plays a crucial role in determining the nature and rate of this convergence [36]. The convergence of the Chebyshev series to the generated functions with the GP model can be mathematically represented by the below formula, where for integer $\nu \geq 1$, let $f$ and all its

*Electronics* **2024**, *13*, 2136



differentiable functions up to $f^{(\nu-1)}$ and the $\nu$ th differentiable $f^{\nu}$ be of bounded variation, $V$, then for any $n > \nu$, we have the following:

$$\|f - f_n\| \leq \frac{2V}{\pi\nu(n-\nu)^{\nu}} \tag{7}$$

Unfortunately, computing the values of the Chebyshev series can be an exceptionally demanding task. In such instances, we might resort to utilizing the Chebyshev interpolant as an alternative to the series itself [36]. The Chebyshev series and interpolants are quite similar in nature, and the convergence of the latter can be articulated through the following expression:

$$\|f - p_n\| \leq \frac{4V}{\pi\nu(n-\nu)^{\nu}} \tag{8}$$

As we can observe in the convergence formulas, the Chebyshev interpolation differs little in terms of accuracy, amounting to only twice the results, which means that in computer calculations, we only lose one bit of accuracy. This minimal loss of precision is compensated by efficiency and numerical stability, which are crucial in many engineering and scientific applications. Thus, we can speak of a significant consistency in the representation of the series and its interpolation using Chebyshev polynomials in terms of our problem. The optimal interpolation nodes are known as Fekete points, which for scalar interpolation are the Legendre nodes. They require solving a large-dimensional eigenvalue problem. Chebyshev nodes have low complexity but are approximately two times worse than Legendre nodes, equating to 1 bit of accuracy [36].

The next problem we must confront is how to obtain the Chebyshev coefficients. This aspect is crucial because the correct determination of these coefficients is fundamental to the effectiveness of the entire method. Thanks to the work of Ahmed and Fisher in 1970 [37], we understand that using interpolation formulas with the FFT on Chebyshev polynomial values can lead to the accurate determination of Chebyshev series coefficients. This process results in minimal loss of accuracy, around the order of one bit [36], due to Chebyshev interpolation. This implies that while efficiently computing the coefficients for a Chebyshev series through FFT, we maintain a high degree of precision, making this method highly valuable for accurate interpolations in various computational applications. Thus, our solution to the problem of function generativity is the use of the Chebyshev series form, where we can quickly calculate the series coefficients by utilizing FFT on Chebyshev node values. An important aspect is selecting the appropriate number of Chebyshev points, ensuring minimal information loss about the original function while avoiding overburdening the GP model with heavy calculations. We can apply a solution similar to that promoted in the modern approach to interpolation [36], where the accuracy of the interpolating polynomial stops at achieving machine precision. For differentiable functions, such as those from the Matérn family, Chebyshev coefficients gradually decrease to a certain level. This allows them to be used to determine the required precision. An example of such behavior can be seen in Figure 1, where we observe that the Chebyshev coefficients decrease as the degree of the Chebyshev series increases until it reaches machine precision. In our case, since we may not achieve machine precision accuracy, where subsequent values cease to be significant, we can use a threshold related to the noise variance, which we can observe as oscillations around a single value of the series coefficient, after a significant drop at the beginning.

The assumptions of our approach involve utilizing knowledge about where to apply specific points for constructing Chebyshev polynomials for the purpose of interpolating the Chebyshev series. This entails strategically selecting points that optimize the polynomial's ability to model the underlying function. Chebyshev polynomials are a sequence of polynomials that satisfy a specific orthogonality condition with respect to a weight function. The most commonly used weight function for Chebyshev polynomials is based on the inverse square root of the difference between the upper and lower bounds of the interval

over which the polynomials are defined. The $k$th Chebyshev polynomial can be expressed as the real part of the function, $z^k$, on the unit circle [36,37]:

$$\begin{aligned} x &= \frac{1}{2}\left(z + z^{-1}\right) = \cos\beta, \quad \beta = \cos^{-1} x, \\ T_k(x) &= \frac{1}{2}\left(z^k + z^{-k}\right) = \cos(k\beta). \end{aligned} \tag{9}$$

One notable property of Chebyshev polynomials is the presence of equally spaced roots along the defined interval. These roots are known as Chebyshev nodes or Chebyshev points and can be expressed as the real parts of these points [36]:

$$x_j = \operatorname{Re} z_j = \frac{1}{2}\left(z_j + z_j^{-1}\right), \quad 0 \le j \le n. \tag{10}$$

An alternative way to define Chebyshev's points is in terms of the original angles [36]:

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad 0 \le j \le n. \tag{11}$$

Chebyshev polynomials are valuable for interpolation, particularly due to their effectiveness compared to polynomial interpolants created through equally spaced points. The clustering of points near the ends of the interval plays a significant role. Chebyshev points offer the advantage of having each point at a similar average distance from the others, as measured by the geometric mean. The resulting polynomial interpolant using Chebyshev points is referred to as the Chebyshev interpolant [36].

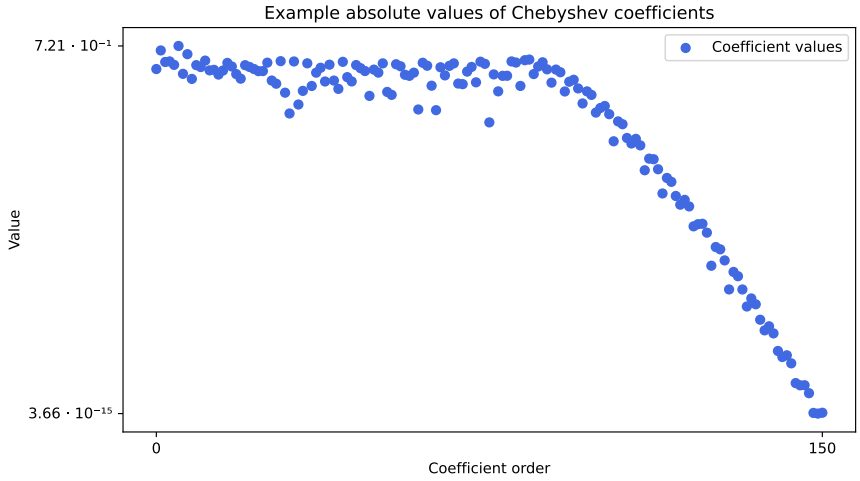



**Figure 1.** The plot presents Chebyshev series coefficient values compared to the order of the series. It illustrates the point at which the series reaches machine precision values with an interpolation algorithm, indicating which series order is required to achieve the desired accuracy of interpolation. For degrees up to approximately k = 80, the Chebyshev series struggles to accurately represent the function, f, as it cannot adequately capture the oscillations occurring at shorter wavelengths. However, beyond this point, there is rapid convergence in the series. By the time the degree reaches k = 150, the accuracy significantly improves to about 15 digits, leading to the truncation of the computed Chebyshev series at this point. This demonstrates the effectiveness of the Chebyshev series in approximating functions at higher degrees, where it becomes more capable of capturing the intricate details of the function.

Furthermore, Chebyshev polynomials can be exclusively represented in a finite Chebyshev series format. This means that the functions $T_0(x), T_1(x), \ldots, T_n(x)$ form the basis for $P_n$ (the Chebyshev series). The relationship between values at Chebyshev points and Chebyshev expansion coefficients is straightforward and can be achieved using the FFT in $O(n \log n)$ operations. The Chebyshev series can be defined as follows:

$$p_n \approx \sum_{k=0}^{n} a_k T_k(x), \tag{12}$$

where $T_k(x)$ represents the $k$th Chebyshev polynomial, and $a_k$ is the associated coefficient. The Chebyshev series exhibits several desirable properties, such as uniform convergence over the interval $[-1, 1]$ for any continuous function and rapid convergence for functions that are analytic or possess a smooth derivative [36,37].

### *2.4. Overview of Proposed Algorithm*

The structure of our proposed algorithm, based on the gathered information, comprises several key components:

1. *Data feeding and fitting the model:* Involves using relevant data to train and adjust the Gaussian process (GP) model, ensuring an accurate representation of the system or phenomenon.
2. *Generative Gaussian process model:* The core of the algorithm, utilizing statistical methods for generating predictions or simulations from input data.
3. *Generating values at Chebyshev points:* Calculating values at crucial Chebyshev points for the subsequent transformation process.
4. *FFT transformation to Chebyshev series:* Applying fast Fourier transform to convert values from Chebyshev points into a Chebyshev series for efficient computational processing.
5. *Series degree optimization with noise variance condition:* Optimizing the Chebyshev series degree while considering noise variance (oscillations), balancing accuracy and computational load.

To help understand the algorithm's workflow and the interconnections between different components, Figure 2 presents a flow chart illustrating the general assumptions of the algorithm. Each step in the flow chart represents a specific part of the algorithm.

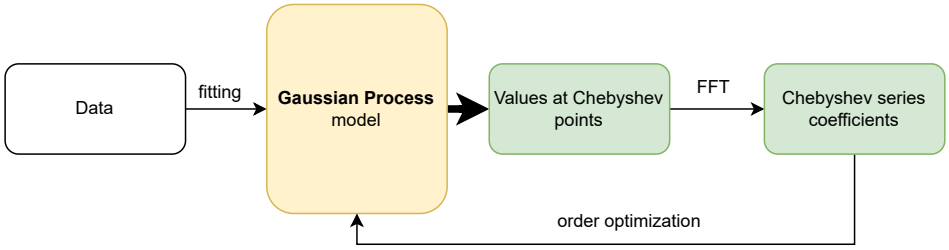



**Figure 2.** The flow chart visually outlines the structure and flow of the proposed algorithm. It begins with data fitting using a Gaussian process model, followed by generating values at Chebyshev points. The next step involves applying the FFT to these values, leading to the generation of Chebyshev series coefficients. The final stage of the algorithm is the optimization of the order of the series, a crucial step to enhance the algorithm's efficiency and accuracy.

### *2.5. Computational Methods*

The current state of GP modeling reflects both its robust potential in statistical learning and the computational challenges inherent in its implementation. GPs are flexible non-parametric models that offer a principled way of encoding prior beliefs about functions, incorporating uncertainty, and handling various types of data. However, the computational demands of GPs, particularly with large datasets, pose significant challenges.

One of the primary computational challenges in GP modeling arises from the need to compute the inverse and determinant of the covariance matrix, a process that is typically cubic in the number of data points $(O(N))$ [2]. This complexity limits the scalability of standard GP methods to large datasets. Moreover, the problem of direct and analytical calculation of the hierarchical model of Gaussian processes has led to the exploration of

alternative solution methods [3]. To address these challenges, several numerical methods and approximations have been developed.

One such method is the Markov chain Monte Carlo (MCMC) sampling method. MCMC is a powerful and widely used computational technique in statistics, machine learning, and various scientific fields. It is particularly valuable when dealing with complex probability distributions and high-dimensional spaces where traditional numerical methods become impractical or inefficient. MCMC provides a way to generate a sequence of random samples from a target probability distribution, allowing us to approximate various characteristics of that distribution [38].

In our research, we are using a Stan model as a probabilistic programming framework that leverages MCMC methods for Bayesian inference. Stan allows us to specify statistical models using a flexible modeling language and then employs MCMC techniques of Hamiltonian Monte Carlo (HMC) to sample from the posterior distribution of model parameters. This combination of Stan's modeling language and MCMC algorithms enables sophisticated and robust Bayesian inference, making it a powerful tool for a wide range of applications. For instance, in our case, it is particularly effective for the hierarchical modeling of GP [39].

The implementation of the GP generative model in Stan involves dealing with multivariate Gaussian processes and generative algorithms. This includes specifying a series of inputs and outputs and defining the probability of outputs conditioned on their inputs. Stan's approach to GPs is characterized by its focus on statistical efficiency and the handling of model conditioning and curvature. This is particularly relevant in Bayesian modeling contexts where the inference about the data-generating process is made using posterior distributions derived from a priori assumptions and observed data. Stan allows direct use of MCMC within its framework and generates data [39].

Building a Gaussian process (GP) model in Stan involves several key steps, from specifying the model to executing the simulation. Here is a general overview of the process we used:

1. Define the data.
2. Specify the GP prior.
3. Model the likelihood.
4. Define hyperparameters.
5. Parameter estimation.
6. Posterior inference.
7. Making predictions and generation of new samples.

Initially, we define both the input and output data for the model. Given that the model is generative in nature, we declare how many samples are required. In selecting priors, a crucial aspect is the choice of the covariance function, which is employed in constructing the covariance matrix. A common practice, also adopted in our approach, is the use of Cholesky decomposition. The modeling of our likelihood is based on declaring a normal distribution, complete with appropriate variance and potential noise. The hyperparameters of our covariance function are treated as priors over hyperparameters and are inferred from the data. The estimation of these parameters is carried out through MCMC sampling. Once the entire model is defined, posterior inference from the model becomes possible. This allows us to obtain the required parameters along with their uncertainties, enabling the use of the model to generate new responses and their associated uncertainties.

MCMC methods within the Stan framework are known for their high precision and computational complexity. These methods, while detailed and accurate, demand considerable computational resources. In contrast, there are alternative approaches that offer faster computational performance with a trade-off in the amount of information extracted from the model. One prominent alternative is the INLA method. The tool employed in our second modeling framework is the R-INLA package, which provides comprehensive implementations for solving problems using sparse matrices, as described in the INLA methodology. INLA, or the integrated nested Laplace approximation, was introduced

by Rue, Martino, and Chopin in 2009 as an alternative to traditional MCMC methods for approximate Bayesian inference [40].

INLA is specifically tailored to models that can be expressed as latent Gaussian Markov random fields (GMRFs) due to their advantageous computational properties. Without going into detail, GPs (especially Matérn ones) are solutions to certain stochastic partial differential equations. Those solutions obtained with finite element methods can be considered GMRFs [41]. In the INLA framework, the observed variables $y = (y_1, \ldots, y_N)$ are modeled using an exponential family distribution. The mean $\mu_i$ (for observation $y_i$) is linked to the linear predictor $\eta_i$ through an appropriate link function. The linear predictor encompasses terms related to covariates and various types of random effects, while the distribution of the observed variables $y$ depends on certain hyperparameters $\theta$.

The distribution of the latent effects $x$ is assumed to follow a GMRF with a zero mean and a precision matrix $\mathbf{Q}(\theta)$, which is dependent on the hyperparameter vector $\theta$ [42]. The likelihood of the observations, given the vector of latent effects and the hyperparameters, can be expressed as a product of likelihoods for individual observations:

$$\pi(\mathbf{y} \mid \mathbf{x}, \theta) = \prod_{i \in \mathcal{I}} \pi(y_i \mid \eta_i, \theta) \tag{13}$$

In this context, the latent linear predictor is denoted as $\eta_i$, which is one of the components of the vector $x$ containing all latent effects. Set $\mathcal{I}$ includes the indices of all the observed values of $y$, although some of these values may not have been observed.

The primary goal of the INLA approach is to estimate the posterior marginal distributions of the model's effects and hyperparameters. Model effects are the components of the model that capture the underlying structure and relationships within the data. This includes fixed effects, which are the deterministic part of the model related to the covariates, and random effects, which account for spatial or temporal correlations, incorporating variation produced by variables in the model that are not the primary focus of the study. Hyperparameters are the parameters of the kernel used in the model. This goal is achieved by leveraging the computational advantages of GMRF and utilizing the Laplace approximation for multidimensional integration. The joint posterior distribution of the effects and hyperparameters can be expressed as [42]:

$$\pi(\mathbf{x}, \theta \mid \mathbf{y}) \propto \pi(\theta) |\mathbf{Q}(\theta)|^{1/2} \exp\left\{ -\frac{1}{2} \mathbf{x}^\top \mathbf{Q}(\theta) \mathbf{x} + \sum_{i \in \mathcal{I}} \log(\pi(y_i \mid x_i, \theta)) \right\} \tag{14}$$

Here, the precision matrix of the latent effects is denoted as $\mathbf{Q}(\theta)$, and its determinant is represented by $|\mathbf{Q}(\theta)|$. Additionally, $x_i = \eta_i$ for values of $i$ that are in the set $\mathcal{I}$ [42].

The computation of the marginal distributions for the latent effects and hyperparameters involves integrating over the hyperparameter space, requiring a reliable approximation of the joint posterior distribution of the hyperparameters [40]. This is accomplished by approximating $\pi(\theta \mid \mathbf{y})$ as $\tilde{\pi}(\theta \mid \mathbf{y})$ and using it to approximate the posterior marginal distribution of the latent parameter $x_i$:

$$\tilde{\pi}(x_i \mid \mathbf{y}) = \sum_k \tilde{\pi}(x_i \mid \theta_k, \mathbf{y}) \times \tilde{\pi}(\theta_k \mid \mathbf{y}) \times \Delta_k \tag{15}$$

The weights $\Delta_k$ correspond to a vector of hyperparameter values $\theta_k$ in a grid, and the specific method for calculating the approximation $\tilde{\pi}(\theta_k \mid \mathbf{y})$ can vary depending on the chosen approach [42].

The primary distinction for Gaussian process (GP) generative models lies in the nature of the output from the modeling approach. In the case of an MCMC-based sampling model in Stan, complete information is obtained in the form of samples at specific points, which are then used to determine the underlying distribution. This approach in Stan allows for a direct and comprehensive understanding of the data distribution, as it provides specific instances (samples) of the model's output [39].

On the other hand, INLA provides information primarily about the distribution of the random variables. Instead of generating samples directly from the distribution values, INLA approximates the posterior distribution of both the latent field and hyperparameters using Laplace approximation methods. This approximation leads to a more efficient computational process but offers a less direct insight into the specific instances of the model's output [41].

## 3. Results

Numerical experiments were conducted using the efficient GP computation algorithm we developed. The process for each scenario consists of the following steps:

1. Selecting and defining the tested function.
2. Generating a small subset of samples with specified noise from the function as data input.
3. Creating a Gaussian process model.
4. Fitting of the Gaussian process model.
5. Generating samples (distributions) at Chebyshev points from the GP model.
6. Converting samples (distributions) into Chebyshev series coefficients.
7. Optimizing the degree of the Chebyshev series.

Approaches vary between the MCMC (HMC) sampling method and the INLA framework. With MCMC, we obtain 4000 samples at each random variable, from which we gather information about the distribution at each point, while with INLA, we only obtain information about the distribution at certain points (for this reason, INLA can run more efficiently). A detailed chart for the algorithm used in the numerical experiments can be seen in Figure 3.

In the initial stage, two different functions are defined for testing purposes. The first function is a simple—at least twice differentiable, i.e., used for Matérn kernels—function and is defined as follows:

$$f(x) = |\sin(x)|^3 \tag{16}$$

The second one is an infinitely differentiable function (used for RBF kernels) and defined as follows:

$$f(x) = e^x + e^{x^2} \tag{17}$$

For these defined functions, a small number of points (20 in our experiments) was generated and then joined with random values from a Gaussian distribution (mean $\mu = 0$; standard deviation $\sigma = 0.1$) to introduce noise. Figures 4 and 5 illustrate the defined functions and the noisy samples generated for them, used in the presented experiment. To simplify subsequent operations with Chebyshev nodes, the domain was set to a range of $[-1, 1]$.

The next stage involves defining and creating a Gaussian process model. At first, we use the GP model created in the Stan, which allows us to create an accurate model based on defined priors and generate samples using MCMC tailored to the specifics of the problem. In our case, the GP model is characterized by a mean of 0. We also use two different covariance functions, one for each case of the function. For the case of an infinitely differentiable function, we use the radial basis function (RBF), also known as the squared exponential (SE) kernel. Also, as we deal with the twice differentiable function, we set the Matérn value to $5/2$. Both of these kernels are the most common in terms of GP usage. Our model in Stan was set up based on general guidelines available in [39]. Our priors were set as normal distributions, we used Cholesky decomposition for our covariance matrix, and generated results for random variables (computation points) in the sample generating block.

*Electronics* **2024**, *13*, 2136



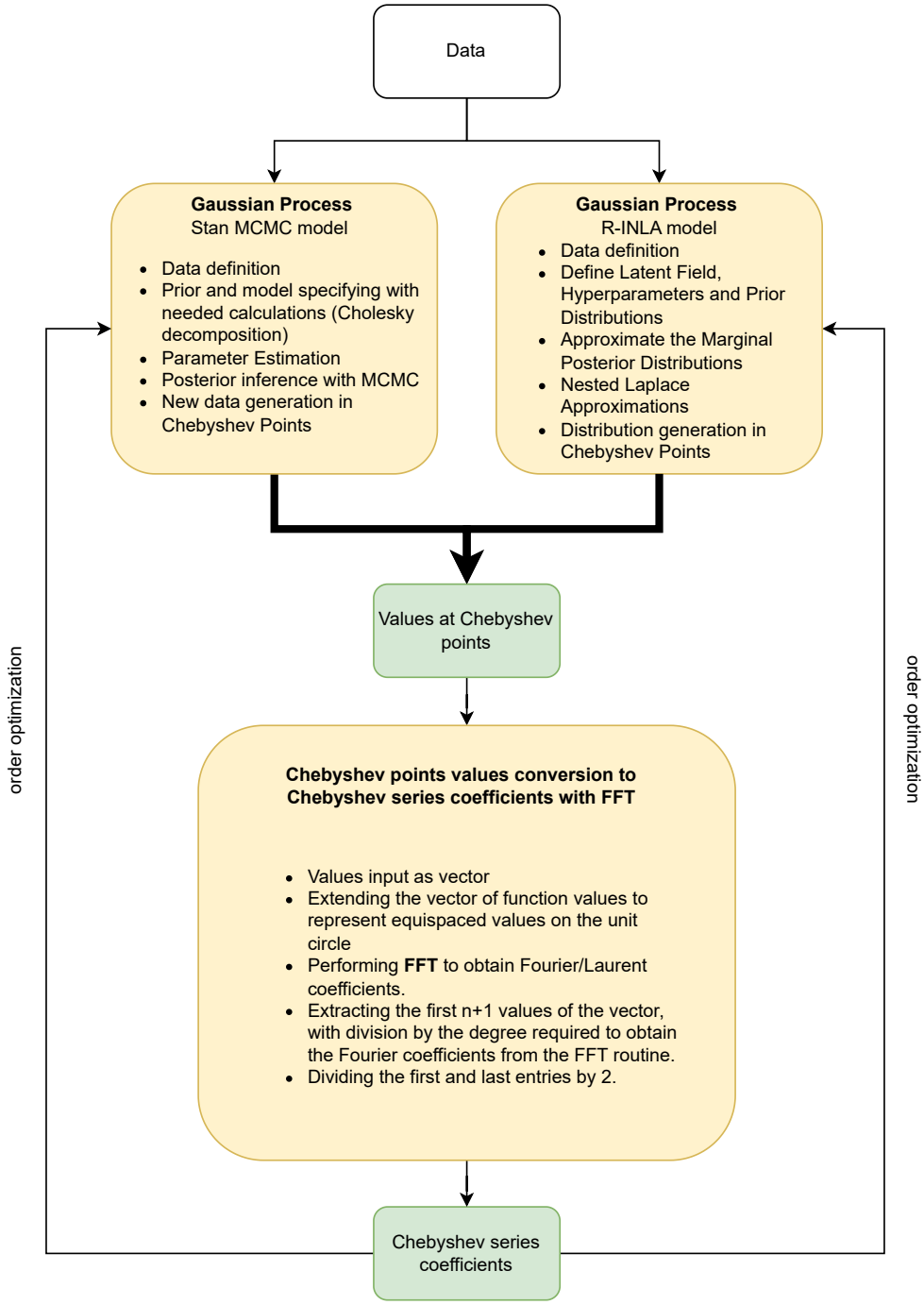



**Figure 3.** The provided flowchart effectively illustrates the intricate steps and progression of the designated algorithm. It initiates with the phase of data fitting, which is conducted using a Gaussian process model. This model is executed in two distinct manners: firstly, by employing the integrated nested Laplace approximation (INLA) method, and secondly, through the use of the Stan framework with Markov chain Monte Carlo (MCMC) techniques. The primary objective of these generative models is to facilitate the generation of values at specific Chebyshev points. Following this, the algorithm advances to the implementation of the fast Fourier transform (FFT). This transformative process is integrated into the script and is pivotal in deriving the Chebyshev series coefficients from the aforementioned values. Concluding the algorithm's sequence is the optimization phase, where the series order is meticulously adjusted. This final step is instrumental in elevating the overall efficacy and precision of the algorithm, ensuring optimal performance and accurate outcomes.

*Electronics* **2024**, *13*, 2136



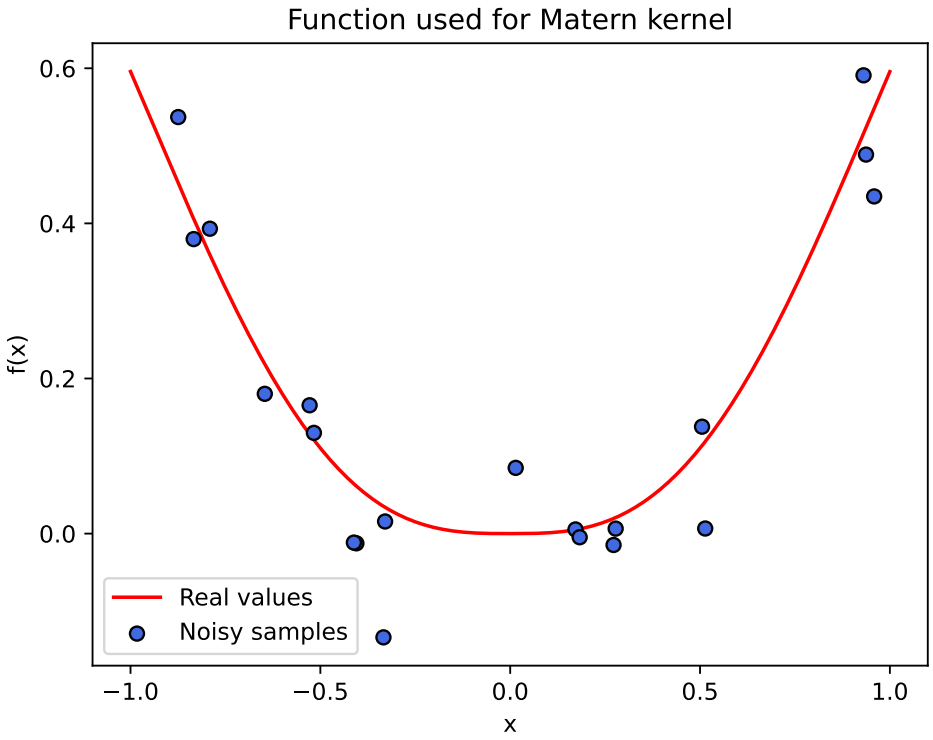



**Figure 4.** The plot shows an example of the twice differentiable function (red line) with sampled noisy values (blue) used for model fitting.

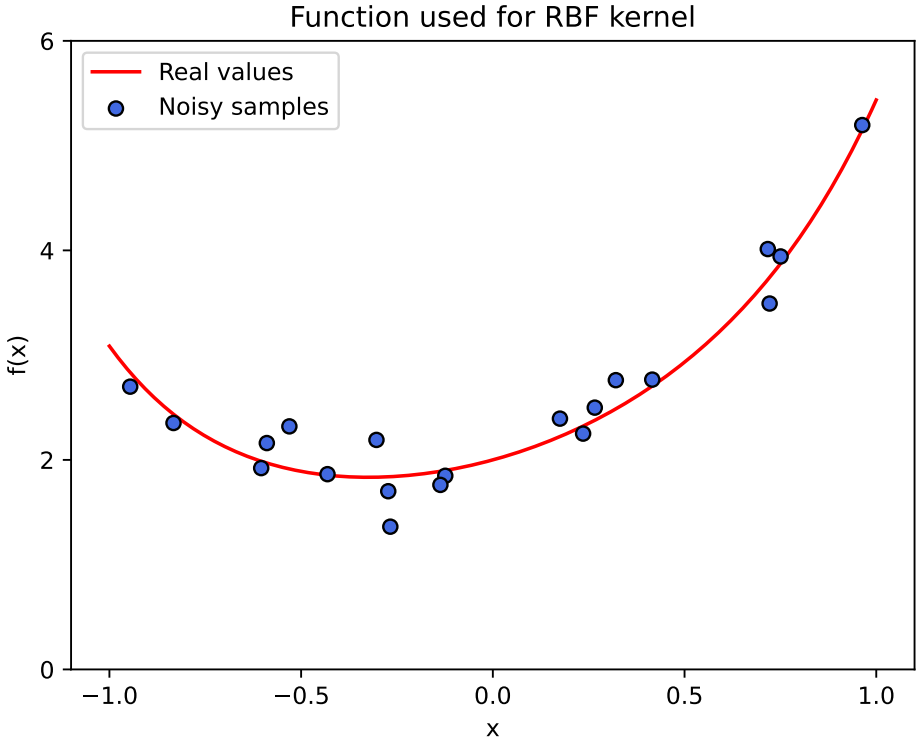



**Figure 5.** The plot shows an example of the infinitely differentiable function (red line) with sampled noisy values (blue) used for model fitting.

In the case of the INLA framework, since we use the stochastic partial differential equation (SPDE) approach, we can only set covariance parameters to use the Matérn family function. But as Matérn is a variation of the RBF function, we can set the parameter $v$ to

high values to approximate RBF function behavior. In the implementation of R-INLA, we first generate a one-dimensional mesh based on which we generate matrix A (the operation matrix on the mesh). As a prior for the similarity matrix, we choose the identity matrix (for random effects). Priors for hyperparameters, as in the case of Stan, are also normal distributions. All created objects are enclosed in what is called a stack, and then a model is created from which we infer the distributions. More on this can be found in [41].

The models were fitted using a small subset of generated noisy data from the original functions. The standard deviation of added noise was $\sigma = 0.1$. It is important to note that the number of Chebyshev nodes used for sample generation directly impacts the degree of the Chebyshev series obtained later, which is closely related to the optimization of this degree, as discussed below. To obtain appropriate preliminary conclusions, the number of Chebyshev nodes we used varied from 10 to 200. Such a discrepancy in values allowed us to examine not only the optimized degree of the Chebyshev series but also, for example, the computational efficiency.

Once the function values were calculated at the Chebyshev nodes, we utilized the FFT to convert them into Chebyshev series by calculating their coefficients, and a script was developed for this purpose. The $k$th Chebyshev polynomial on the unit interval can be interpreted as the real part of the monomial $z^k$ on the unit disc in the complex plane, given by $z^k = Re(z^k) = \frac{1}{2}(z^k + z^{-k})$. Thus, when adding n+1 Chebyshev polynomials, a truncated Laurent series in the variable $z$ was obtained, with the same coefficients for the $z^k$ and $z^{-k}$ terms. Similarly, the Chebyshev points on the interval $[-1, 1]$ can be seen as the real parts of equispaced nodes on the unit circle.

This connection between the unit interval and the unit circle allowed us to utilize Fourier analysis [37]. A truncated Laurent series with equal coefficients for the $z^k$ and $z^{-k}$ terms is equivalent to a Fourier series in the variable $s$, where $z = \exp(i \cdot s)$. Therefore, Fourier and Laurent coefficients are identical, and the coefficient vector is symmetric since the same factor multiplies the $z^k$ and $z^{-k}$ terms. In this context, the Chebyshev coefficients correspond to the first n + 1 terms of this vector, with the first and last coefficients divided by 2.

The script based on information introduced in [36,43] was implemented and performed via the following steps:

1. We gathered function values from the model results into vectors.
2. We extended the vectors of function values to represent equispaced values on the unit circle, progressing anticlockwise, starting from an $x$ value of 1.
3. We performed FFT to obtain Fourier/Laurent coefficients. Since the Chebyshev coefficients are real-valued, the real part of this vector was taken to eliminate any spurious imaginary components due to rounding errors.
4. We extracted the first $n + 1$ values of the vector, adjusting by the degree required to obtain the Fourier coefficients from the FFT routine.
5. We divided the first and last entries by 2.

The script can also be seen in Figure 3 as a part of the whole algorithm.

With the calculated Chebyshev series coefficients for the generated values (4000 for each node in the case of MCMC) from the model, we obtained the marginal distribution of the coefficients. In the case of the INLA framework, our conversion script was based on the mean values of the distribution in each node, resulting in single coefficient values from the FFT, in contrast to the MCMC distribution. The previously mentioned challenge was selecting the optimal degree of the Chebyshev series, which subsequently determined the calculation of the GP in the appropriate number of Chebyshev nodes. The method for estimating the optimal degree involved evaluating the behavior of coefficient values based on the degree. Consequently, we assessed the significance of the coefficients to determine a cutoff threshold beyond which the remaining coefficients could be considered negligible. This threshold could be reached when values approached machine precision or oscillated around a value. In our case, we decided to use the value based on our noise variance and set the threshold to $t = 0.1 \cdot \sigma^2$. This step reduces the Chebyshev series degree. Since we

were dealing with distributions, we used the expected values of the coefficients to optimize the series order needed to accurately represent the original function.

The results of conducted tests for different orders, ranging from 10 to 200, can be seen below. We gathered the plots of results from models (mean values with confidence interval) and coefficient values together. In Figures 6 and 7, we can see results for the MCMC sampling approach with the use of the Matérn kernel with at least twice differentiable functions. Figures 8 and 9 show the results of the same model but for RBF kernels and infinitely differentiable functions. The results from the INLA framework can be seen in Figures 10 and 11 for at least twice differentiable functions and in Figures 12 and 13 for infinitely differentiable functions. We also gathered information about the average computing time for each experiment in different combinations of kernels, frameworks, and Chebyshev node counts; they can be seen in Table 1.

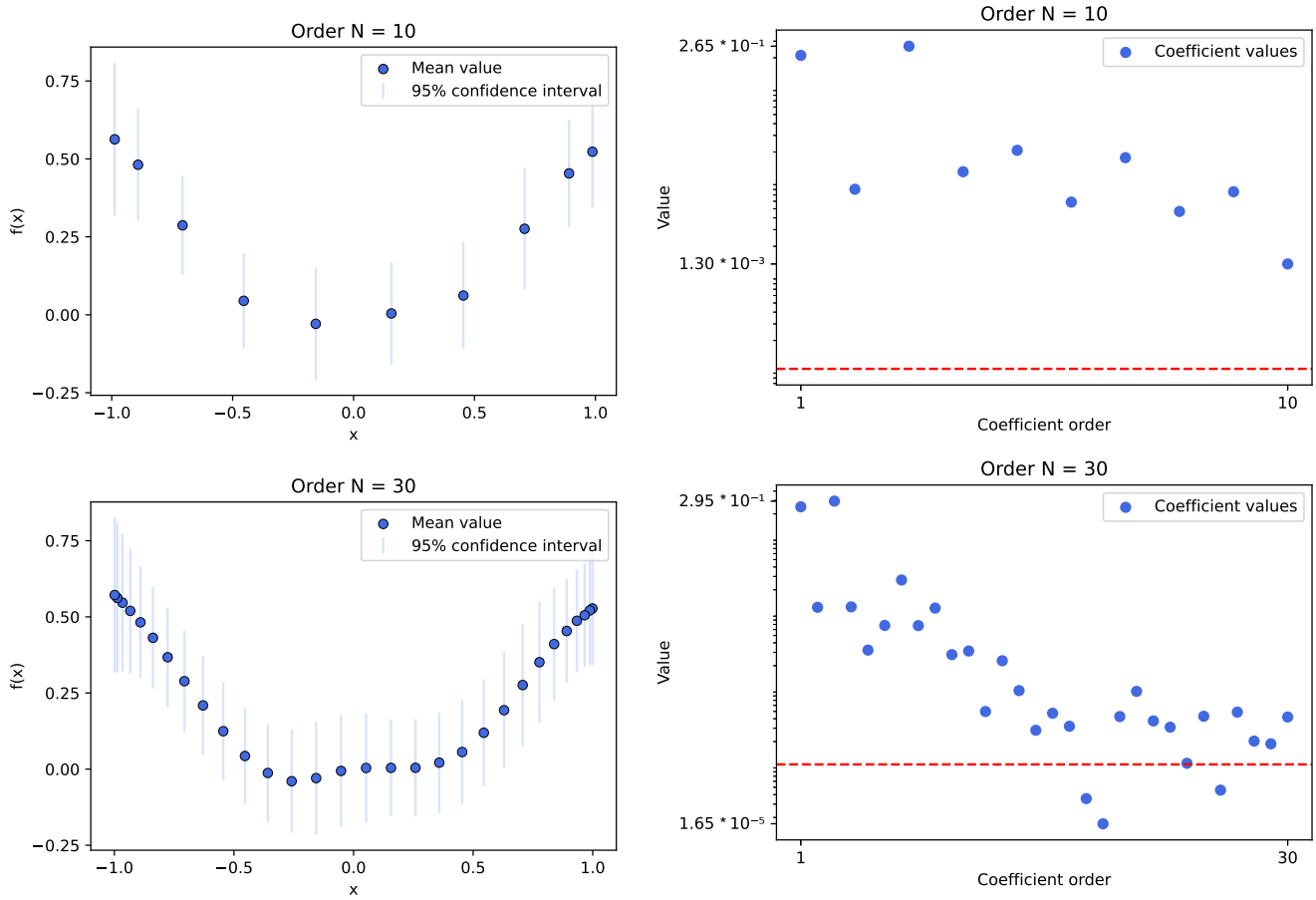



**Figure 6.** The plots present results for the experiment conducted using the *MCMC sampling* method, with a Gaussian process model that employs a *Matérn kernel*. Experiments use sample data from selected functions and the random variables are set in *10 and 30 Chebyshev nodes*. The left plots show the mean values of calculated samples as well as the 95% confidence interval received from the sampled distribution. The right plots show the behavior of Chebyshev series coefficient values compared to their order. The red line shows the selected threshold for optimization ($0.1 \cdot \sigma^2$). We can deduce that both results are nowhere near machine precision, so we should rely on the selected threshold. In the case of only 10 Chebyshev nodes, we did not even reach the threshold value, whereas with 30 nodes, sample number 18 did. As a result of our optimization, we should take the value around 18 as the order of the Chebyshev series. Moreover, the computation times for the small number of nodes were acceptable for real-life cases, ranging from several seconds to a dozen seconds.

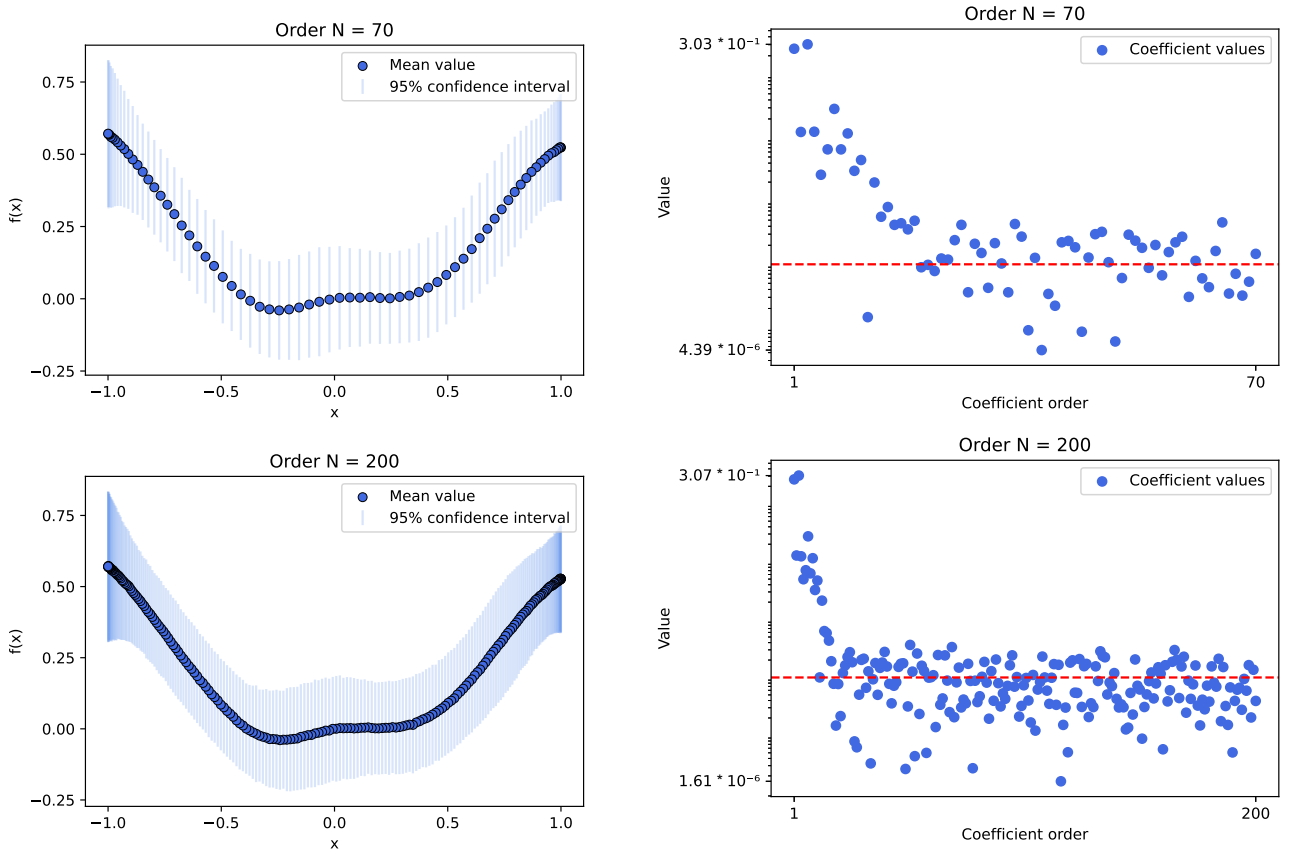



**Figure 7.** The plots present results for the experiment conducted using the *MCMC sampling* method, with a Gaussian process model that employs a *Matérn kernel*. Experiments use sample data from selected functions and the random variables are set in *70 and 200 Chebyshev nodes*. The left plots show the mean values of calculated samples as well as the 95% confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values relative to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). We can deduce that oscillation around the threshold starts early, around the order of 30, so our previous assumption of selecting the 18th order should be correct. Moreover, the values decrease only slightly up to the order of $10^6$, which suggests that they probably won't reach machine precision very soon. These experiments were conducted primarily to observe the behavior after reaching the threshold and to verify if the previous assumption was correct. Additionally, they allowed us to check the computation times for the number of nodes. Unfortunately, these times were unacceptable, ranging from several minutes to 10 min.

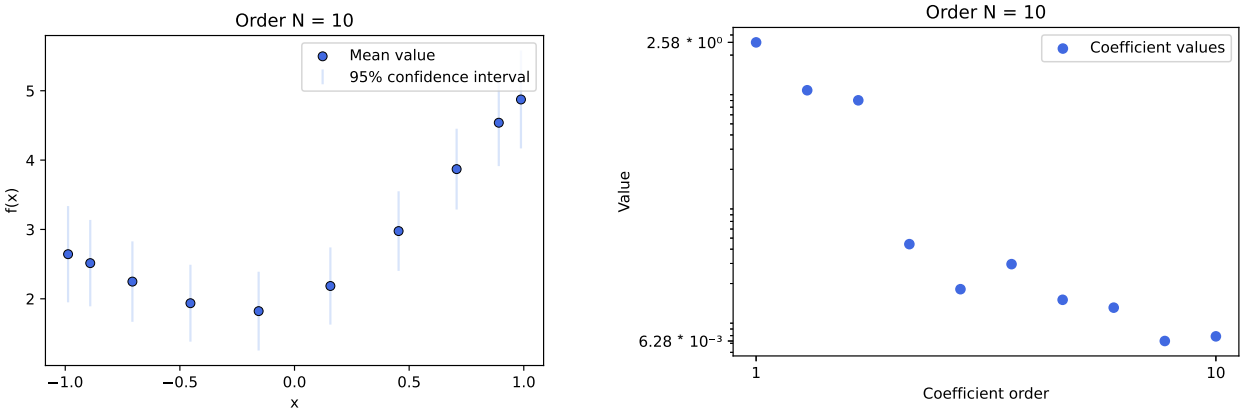



**Figure 8.** *Cont.*

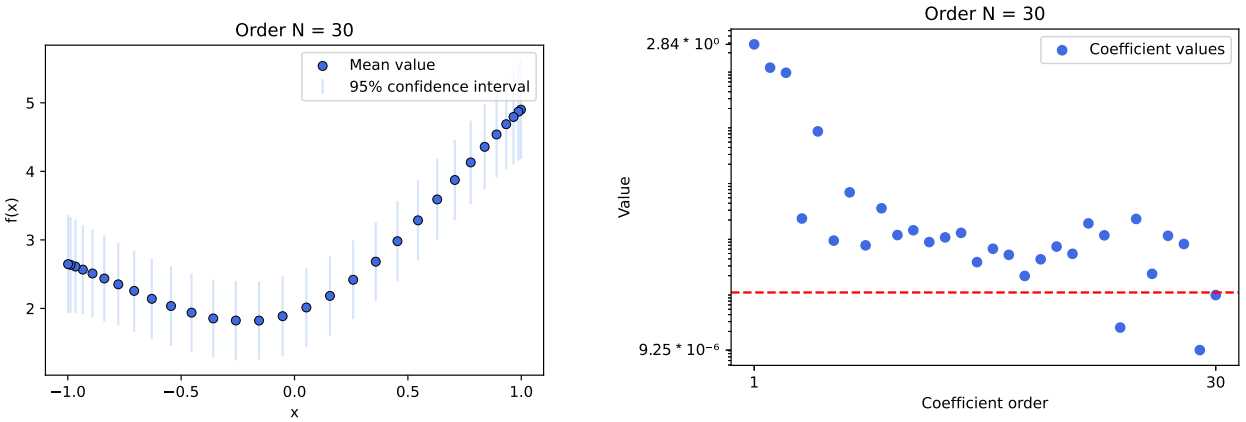



**Figure 8.** The plots present results for the experiment conducted using the *MCMC sampling* method, with a Gaussian process model that employs an *RBF kernel*. Experiments use sample data from a selected function, with random variables set at *10 and 30 Chebyshev nodes*. The left plots show the mean values of calculated samples as well as the 95% confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values relative to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). The first 10 coefficients are highly informative; none reached the selected threshold. When we increased the number of selected nodes, it was clear that the 24th coefficient crossed the threshold. Contrary to the previous case, this function required more coefficients to gather sufficient information based on our assumptions. Moreover, as with the other MCMC case, computation times for this number of nodes were acceptable and similar to the Matérn ranging from several seconds to a maximum of a dozen seconds.

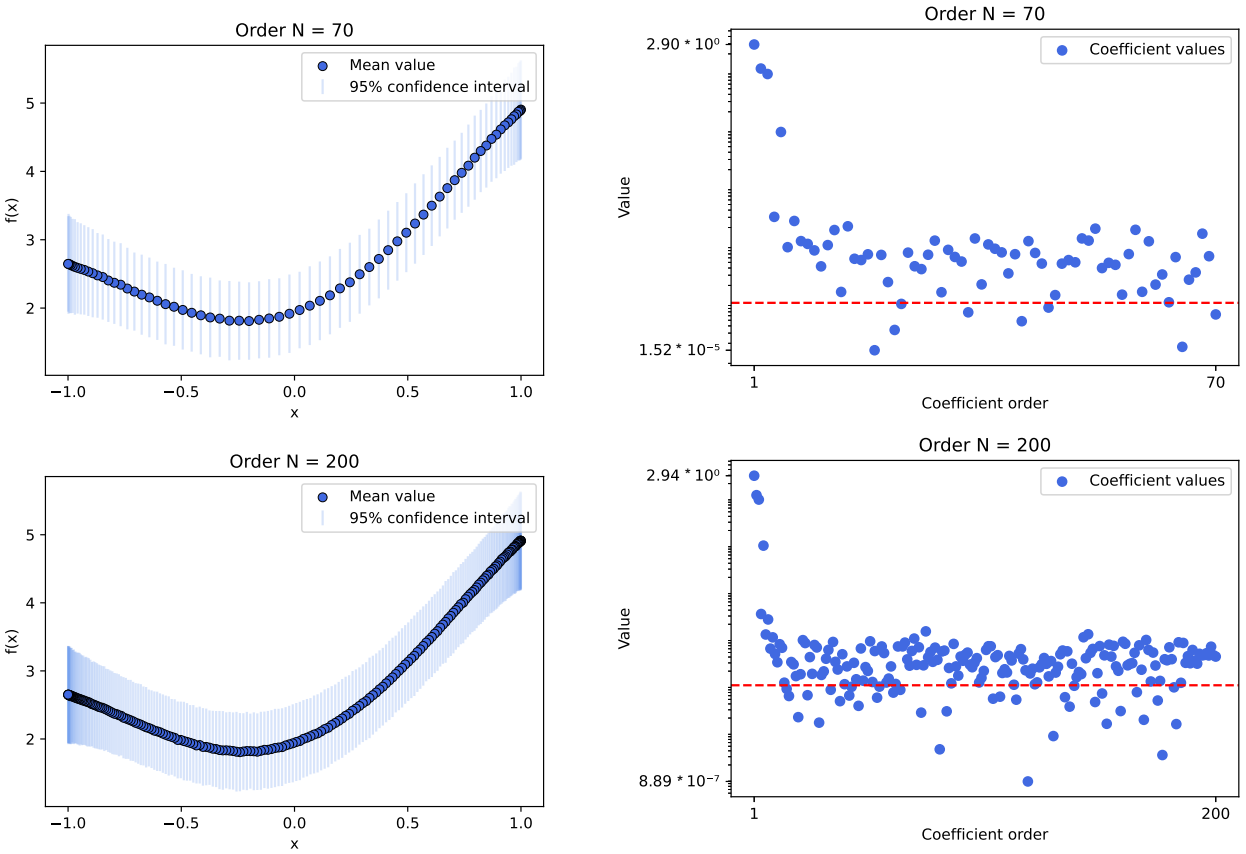



**Figure 9.** The plots present results for the experiment conducted using the *MCMC sampling* method, with a Gaussian process model that employs an *RBF kernel*. The experiments use sample data from a selected function, with random variables set at *70 and 200 Chebyshev nodes*. The left plots show the

mean values of calculated samples, as well as the 95% confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values relative to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). We can deduce that adding more nodes to the GP model does not actually help gather more information. Oscillations around the threshold start around the order of 30 and do not seem to reach machine precision. Additionally, the computation times for this number of nodes were unacceptable, ranging from several minutes to a dozen minutes.

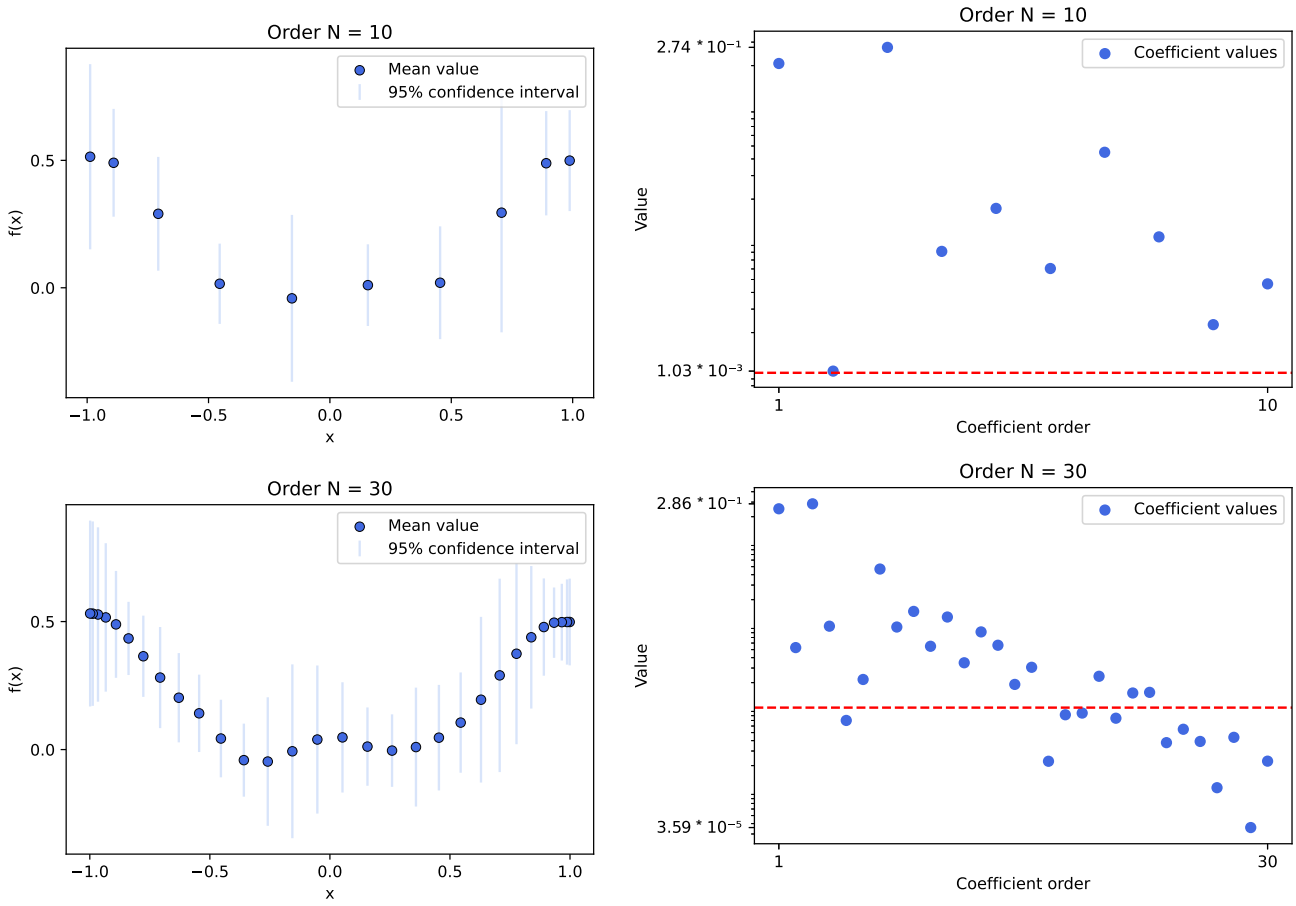



**Figure 10.** The plots present results for the experiment conducted using the *INLA framework* method, with a Gaussian process model that employs a *Matérn kernel*. Experiments utilized sample data from a selected function with random variables set at *10 and 30 Chebyshev nodes*. The left plots show the mean values of calculated samples as well as the 95% confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values in relation to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). Contrary to MCMC experiments, the INLA framework, in this case, managed to catch the threshold at the 2nd order, highlighting the importance of checking higher orders, especially since no other coefficient reached the selected threshold. For the 30th order, the 5th sample could be considered as the cutoff for optimization, but the next value below this is the 16th coefficient, with a clearly visible decreasing trend between the 6th and 16th coefficients. We can deduce that oscillation slightly starts around the order of 17, but we need to verify this with higher orders. The computation times for the node numbers using INLA were excellent and completed within several seconds; this was predictable due to the INLA approximation properties.

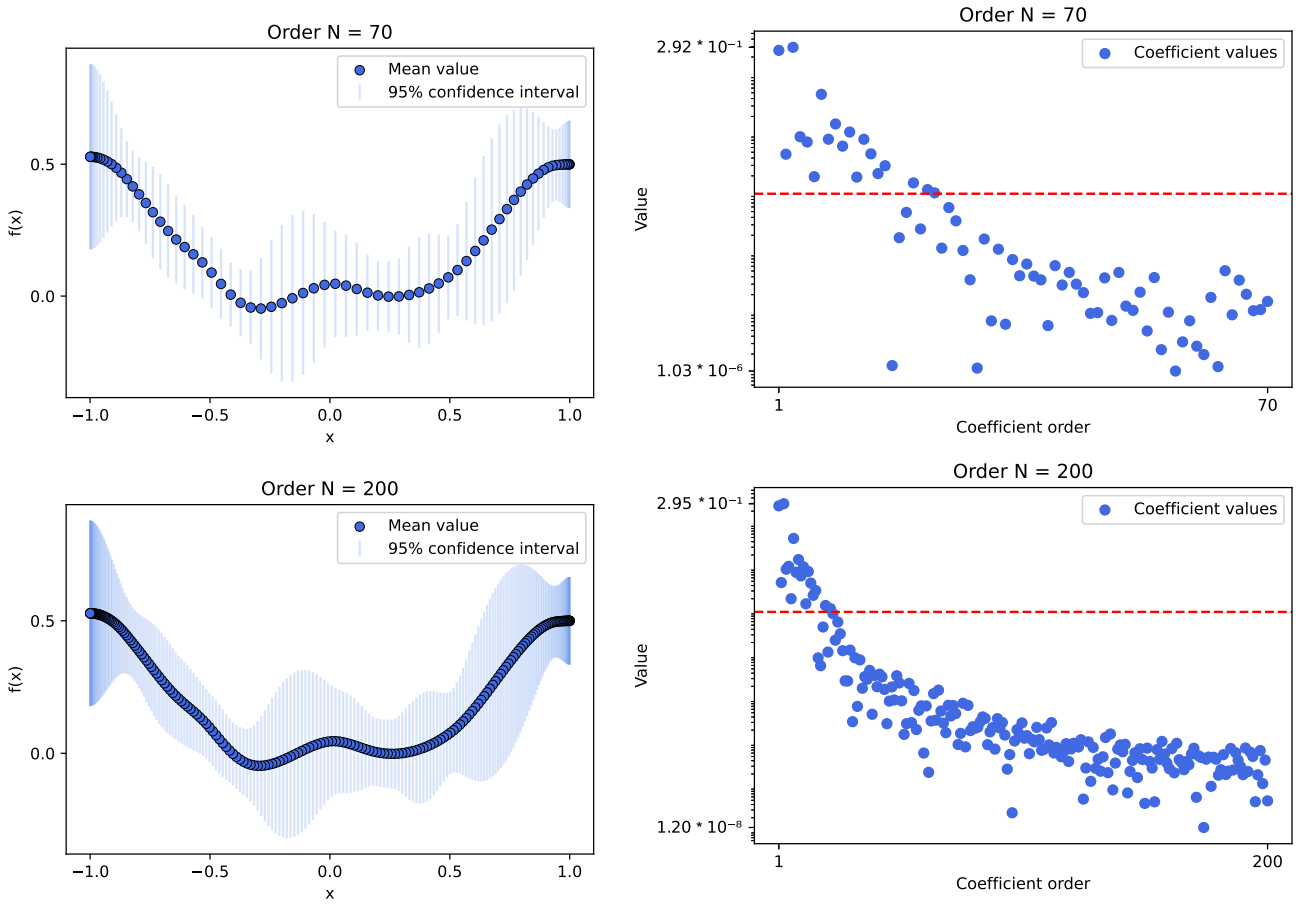



**Figure 11.** The plots present results for the experiment conducted using the *INLA framework* method, with a Gaussian process model that employs a *Matérn kernel*. Experiments utilize sample data from a selected function, with random variables set at *70 and 200 Chebyshev nodes*. The left plots show the mean values of calculated samples along with the 95% confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values relative to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). Based on previous experiments with a lesser value or order and the information gathered here, we can deduce that values truly cross the threshold around the order of 20. Moreover, coefficient values are slowly decreasing and do not start to oscillate, but in our case, it makes no sense to aim for values below the threshold. As expected, due to the INLA approximation properties, the computation times for any number of nodes were excellent, conducted within several seconds.

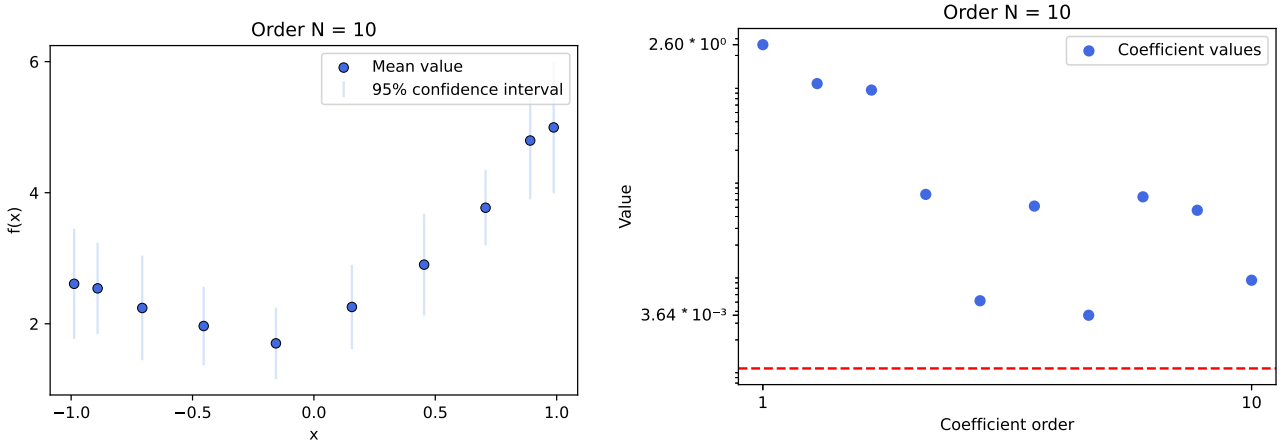



**Figure 12.** *Cont.*

*Electronics* **2024**, *13*, 2136



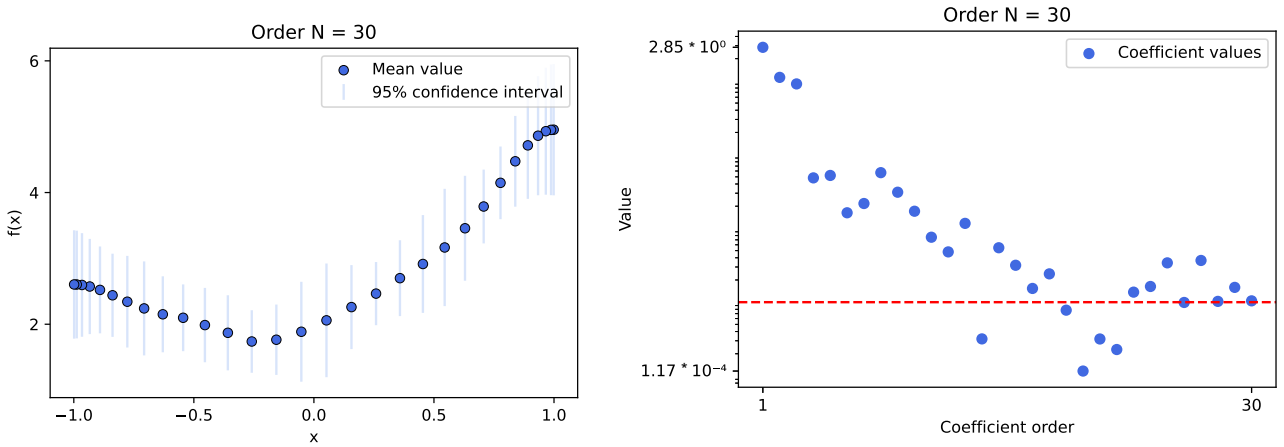



**Figure 12.** The plots present results for the experiment conducted using the *INLA framework* method, with a Gaussian process model that employs a modified Matérn kernel, simulating the *RBF* one. Experiments utilized sample data from a selected function with random variables set at *10 and 30 Chebyshev nodes*. The left plots show the mean values of calculated samples along with the 95% confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values relative to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). In this case, the INLA framework mimics the behavior observed in previous cases with low-order coefficients, where values reach the threshold quickly but continue to show a decreasing trend. We can deduce that the required value is likely around the order of 20. The computation times for this number of nodes were excellent, conducted within several seconds.

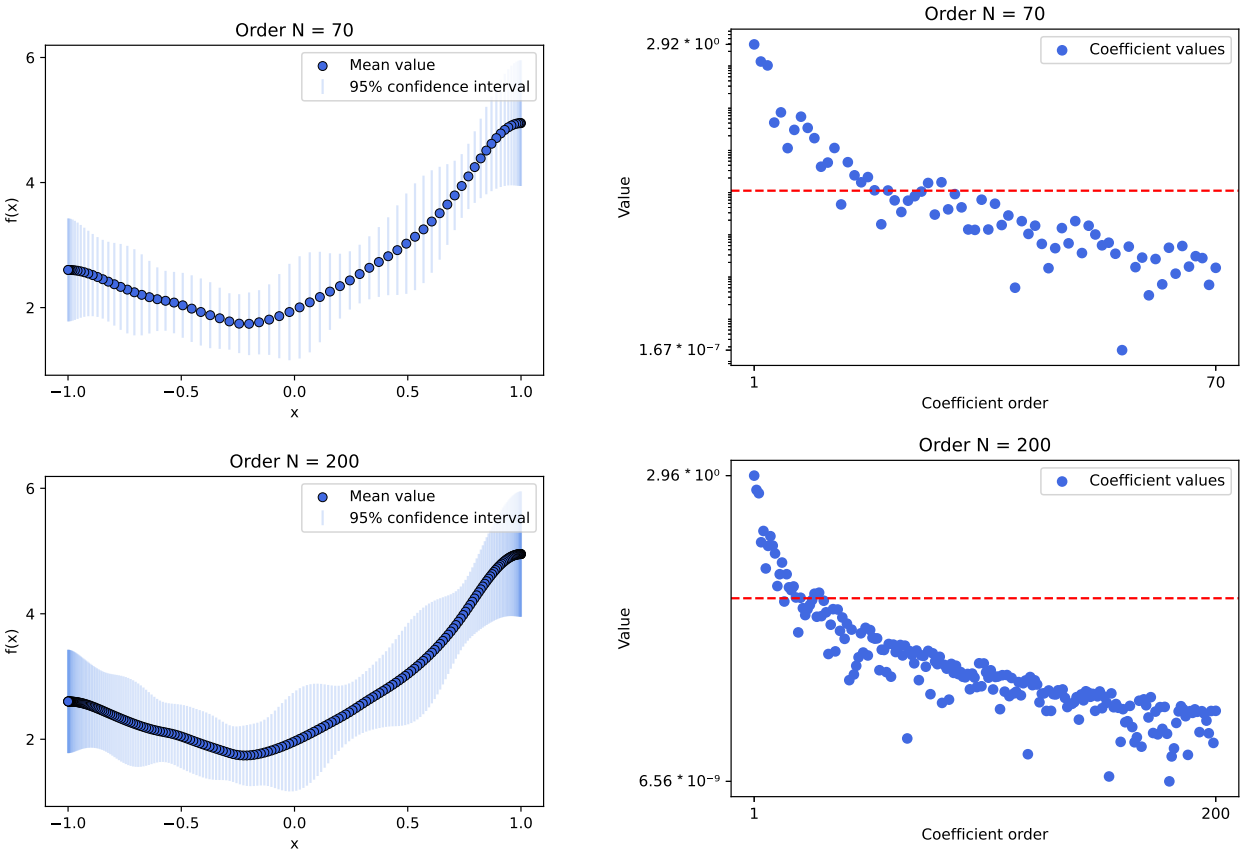



**Figure 13.** The plots present results for the experiment conducted using the *INLA framework* method, with a Gaussian process model that employs a modified Matérn kernel, simulating the *RBF* one. Experiments utilized sample data from selected functions, with random variables set at *70 and 200 Chebyshev nodes*. The left plots show the mean values of calculated samples along with the 95%

confidence interval derived from the sampled distribution. The right plots illustrate the behavior of Chebyshev series coefficient values relative to their order. A red line indicates the selected threshold for optimization ($0.1 \cdot \sigma^2$). We can deduce that all INLA cases behave similarly: initially, the values decrease rapidly, reaching the threshold, and then the downward trend slows down. The computation times for this number of nodes were excellent, conducted within several seconds.

**Table 1.** The table presents average computation times for various selected experiments using different numbers of samples: 10, 20, 30, 50, 70, 100, 150, and 200 for Matérn and RBF kernels, using MCMC and INLA frameworks. The time for MCMC methods significantly increases depending on the number of samples for the GP model, whereas for INLA, it remains practically unchanged. This is primarily due to the initial assumption of optimizing computations for INLA. The kernel choice did not have an impact on the computation time.

| Samples | MCMC: Matérn | MCMC: RBF | INLA: Matérn | INLA: RBF |
|---|---|---|---|---|
| **10** | 9.6 s | 10.1 s | 3.1 s | 2.4 s |
| 20 | 15 s | 12 s | 2.5 s | 3 s |
| 30 | 24.6 s | 27.8 s | 3.5 s | 3.1 s |
| 50 | 40 s | 51 s | 2.7 s | 3 s |
| 70 | 1 min 40 s | 2 min 10 s | 2.7 s | 2.8 s |
| 100 | 3 min 11 s | 2 min 51 s | 3 s | 2.9 s |
| 150 | 6 min 40 s | 7 min 51 s | 3.1 s | 2.7 s |
| 200 | 10 min 30 s | 13 min 11 s | 3.8 s | 4 s |

The overall conclusion regarding MCMC approaches is that there is a clear barrier to the optimal degree level of the Chebyshev series. A clear cut-off line of the threshold of useful information can be used but can also be achieved by catching the oscillations around the same value of the series coefficient. For the case of the Matérn kernel, the value of oscillations is slightly below the threshold, and for the RBF kernel, it is a bit above the threshold value. Both approaches provide us with the value of nodes we should use; both were around 10–30 orders. It is important to check the optimization of the order because any decrease in these values directly impacts the size of the covariance matrix and, thus, the computation time, losing very little information.

In the case of the R-INLA framework approach, we can conclude that despite achieving high Chebyshev degree values for the coefficients and observing a decreasing tendency, we do not reach machine precision, and the values do not oscillate around any value. In such a situation, we can use the selected threshold, which was useful in the previous experiment with MCMC, where the optimal cut-off values of the necessary coefficients were identified. Moreover, it is important to understand that the INLA framework is optimized to calculate Gaussian random fields (multivariate GPs), so decreasing the number of nodes does not significantly impact execution time.

This difference in results comes from a different calculation method. MCMC uses sampling at each point, so we receive more information, such as the values of each generated sample and chain, from which probability distributions are later calculated. Due to this possibility, MCMC takes into account entire value vectors, not just distribution information, for calculations related to the Chebyshev series. This allows us to optimize the degree of the series with FFT conversion. In the case of R-INLA, the SPDE approach only provides information about the distribution at given points. Therefore, we can only use data on the expected value and standard deviation when calculating values for the Chebyshev series. It is also important to consider the execution times for model fitting and data generation. MCMCs are known for their long and complex calculations but for cases involving 10 to 50 random variables, the execution time did not exceed several dozen seconds. However,

for 200 variables, it took almost 15 min on a moderate computer. We did not encounter a similar problem with R-INLA, as it is optimized for calculations on sparse matrices; the execution times for all attempts did not exceed several seconds.

By employing the selected Chebyshev nodes, we obtained the expected results of the GP model in the form of Chebyshev series coefficients, rather than computing the GP at every point. It is also crucial to emphasize that, regardless of the chosen framework or the degree of the results, all are presented in the form of coefficients. This form allows us to encode information in the form of the Chebyshev series itself, which serves as an approximation of the original function and can help in generating more data for generative models. Thus, it carries additional information about the original function, unlike other frequently used methods. It is also important to note that, unlike MCMC, R-INLA is already an approximation method, so the use of subsequent optimizations for simple one-dimensional cases did not bring much improvement. However, in the case of the MCMC approach, our method can facilitate its use in scenarios where it was previously infeasible, especially where generative models are needed.

## 4. Discussion

In the field of computational process optimization for GPs, one of the more significant and recognizable solutions is sparse GP. This approach involves strategically selecting a subset of informative points, known as inducing points, to represent the underlying data. The main advantage lies in addressing the computational challenges associated with traditional GPs, particularly when dealing with large datasets. By employing sparse GPs, the model's scalability is greatly improved, allowing for efficient processing of extensive datasets while maintaining the flexibility and power of GPs in capturing complex relationships [16,17].

One way to enhance the performance in representing underlying functions through sparse models of GPs is through rectangularization. In situations where data are sparse, avoiding overfitting and optimizing Gaussian process regression (GPR) hyperparameters becomes challenging, especially in high-dimensional spaces where data sparsity cannot be practically resolved by adding more data. The success or failure of GPR applications hinges on the optimal selection of hyperparameters. The authors propose a method that facilitates parameter optimization through the rectangularization of the GPR-defining equation. They use a 15-dimensional molecular potential energy surface as an example; they illustrate the effectiveness of this approach in achieving hyperparameter tuning even with very sparse data [44].

Other works focus on optimizing the performance of online updating for sparse GPs. Research presented in [45] delves into the relationship between a class of sparse-inducing point GP regression methods and Bayesian recursive estimation, enabling Kalman filter-like updates for online learning. Unlike prior focus on the batch setting, this study concentrates on mini-batch training. Leveraging the Kalman filter formulation, the proposed approach recursively propagates analytical gradients of the posterior over mini-batches, resulting in faster convergence and superior performance. The approach presented in [46] was developed for both the fully independent training conditional (FITC) and the partially independent training conditional (PITC) approximations, enabling the inclusion of a new measurement point $(x_n + 1)$ in $O(m^2)$ time, where m represents the size of the set of inducing inputs. The online nature of the algorithms enables the forgetting of earlier measurement data, resulting in a memory space requirement of $O(m^2)$ for both FITC and PITC.

Works that do not rely on sparse GPs can also be found. In [12], the authors propose a way to optimize GPs for time series gap-filling in satellite image processing by replacing the per-pixel optimization step with cropland-based pre-calculations for GPR hyperparameters $\theta$.

In [47], the authors present a novel algorithm in terms of GP optimization: BKB (budgeted kernelized bandit). It is designed for optimization under bandit feedback. BKB achieves near-optimal regret and convergence rates with nearly constant per-iteration complexity. Notably, it makes no assumptions about the input space or GP covariance. The

algorithm combines a kernelized linear bandit approach (GP-UCB) with randomized matrix sketching using leverage score sampling. The authors demonstrate that randomly sampling inducing points based on posterior variance accurately provides a low-rank approximation of the GP, maintaining reliable variance estimates and confidence intervals.

In the last example, the authors explore an alternative global optimization framework called optimistic optimization, which is computationally more efficient. This approach leverages prior knowledge of the search space's geometry through a dissimilarity function. The study aims to integrate the conceptual advantages of Bayesian Optimization with the computational efficiency of optimistic optimization. This is achieved by mapping the kernel to a dissimilarity, resulting in the development of an optimistic optimization algorithm [48].

The practical application of the proposed solution can be easily illustrated through existing examples, such as in the previously mentioned problem of anomaly detection in industrial processes [29]. This problem involves several steps: data acquisition, generating new samples using g = Gaussian processes (GPs), and classification through data depth. While other aspects of the issue would remain unchanged, for the generative model, our presented solution could be applied. In the original problem, the authors used a GP implementation based on the ML-II solution. By applying our algorithm, MCMC sampling could be used in such a problem, potentially providing more information about the generation of functions. Another example is the previously discussed issue of using a Gaussian mixture in diagnosing the state of an engine based on its startup characteristics [35]. Here, the authors employ several models aimed at generating new samples for classification. As in the previous case, it would be valuable to explore the application of our proposal, which relies on optimally selecting the degree of the Chebyshev series and Chebyshev points as random variables, thereby enabling the use of MCMC sampling.

As presented by employing various methods, including sparse GP variations, we can address the issue of time-consuming computations in GP modeling. In contrast to the standard sparse approach, our proposal relies on an innovative method characterized by two key features: a reduction in computation time and the ability to reconstruct the original function. The first feature is ensured through the imposition of domain constraints in computations, where we model the GP using a limited set of random variables at selected Chebyshev nodes. The second feature arises from the form of the Chebyshev series, achieved by transforming function values at Chebyshev nodes using FFT, allowing us to obtain coefficients of the Chebyshev series. This representation provides information about the original function. Additionally, by analyzing coefficients of different degrees, we can guarantee a minimal, meaningful number of Chebyshev nodes. Our experiments confirm that our algorithm significantly reduces computation time, especially in the context of sampling-based methods, while maintaining unique properties not encountered in standard sparse approaches.

## 5. Conclusions

The incorporation of the Chebyshev property to enhance the computation of the GP is a topic frequently overlooked. Over the years, one of the primary challenges associated with GPs has been their computational complexity, particularly when dealing with a large number of points. This complexity is exacerbated when employing the MCMC method for sample generation, often resulting in impractical computation times.

In response to this issue, our proposed approach aims to mitigate the computational burden by reducing the calculations of random variables to the appropriate number of Chebyshev nodes. Moreover, our study focuses on leveraging the values at Chebyshev nodes (and their marginal distribution) gathered from the generative model to construct the Chebyshev series. Chebyshev series are recognized for their strong convergence characteristics, especially in relation to differentiable functions produced by the model. Computing the values of the Chebyshev series can be a highly challenging task so we are using the Chebyshev interpolant as a substitute for the series itself with just one bit loss of accuracy. With conversion based on FFT, we can manage to transform distributions (values)

of random variables in Chebyshev nodes to Chebyshev series coefficients. The critical challenge addressed was the judicious selection of the optimal degree for the Chebyshev series, a parameter that influences the computation of the GP within an appropriate number of Chebyshev nodes. To determine the optimal degree, we employed a method that involved assessing the behavior of coefficient values based on the degree, subsequently establishing a cutoff threshold based on the significance of coefficients. This threshold was identified when values approached machine precision or exhibited oscillations around a similar value, leading to a reduction in the Chebyshev series degree.

Our proposed solution is also characterized by the feature of receiving more information about the original function. By transforming the function values at Chebyshev nodes using FFT, we can reduce the amount of Chebyshev points and obtain the Chebyshev interpolation of the generated function. This form of Chebyshev representation can help us generate more variations from the model, with respected uncertainty from the GP model. This approach works very well for generative models. These models are designed to map new data samples based on collected information from the original function. Using MCMC sampling in such cases is useful and our approach allows its application in a reasonable time. Moreover, thanks to the use of the after-mentioned Chebyshev interpolation, we have the possibility of transferring additional information about the function through appropriate coefficients.

From numerical experiments, we can conclude significant differences between the results between MCMC and R-INLA frameworks. These differences arise from their distinct calculation methods. MCMC, using sampling at each point, considers entire value vectors, providing more information for Chebyshev series calculations. This enables the optimization of the series degree. On the other hand, R-INLA, utilizing an SPDE approach, offers information solely about the distribution at given points, limiting the utilization of data to expected values and standard deviations in the context of calculating values for the Chebyshev series.

Furthermore, considering the execution times of the algorithm is crucial. MCMC, known for lengthy and intricate calculations, exhibited significant variations. For 10 random variables, the execution time remained within several seconds, whereas for 200, it extended to about 11 min. In contrast, R-INLA, designed for optimized and approximated calculations on sparse matrices, consistently demonstrated shorter execution times, not exceeding one minute in all attempts. The utilization of the selected Chebyshev nodes in our script facilitated the acquisition of expected results for the GP generative model in the form of Chebyshev series coefficients, avoiding the need for computing the GP at every individual point. Preliminary results indicate a substantial reduction in calculation time in the case of MCMC-based calculations. In both cases, it is possible to utilize a reduced number of nodes for computations without significant loss in the structures of the reconstructed functions from the models. This promising outcome underscores the potential efficiency gains achievable through our proposed method, especially in terms of an MCMC-based approach. Our solution allows us to use the MCMC sampling in cases where it was not possible before due to the problem of computation time. For the R-INLA case, the reduction in computational time is not as spectacular as in the MCMC case, but that outcome should be expected from the already approximate calculations; therefore, the use of the method may seem unnecessary.

It is worth noting that our suggested approach shows promise but necessitates further testing and validation on real case scenarios. Future considerations should delve into exploring alternative, less frequently used covariance functions for GPs, adjusting priors and model settings to better align with specific problem nuances, and implementing scalability measures to ensure the versatility and applicability of the proposed solution across diverse domains. As our research progresses, it is imperative to validate and refine the proposed approach to contribute meaningfully to the optimization of GP computations.

**Author Contributions:** Conceptualization, A.D. and J.B.; methodology, J.B.; resources, A.D. and J.B.; writing—original draft preparation, A.D.; writing—review and editing, A.D. and J.B.; visualization, A.D.; supervision, J.B.; project administration, J.B.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by AGH's Research University Excellence Initiative under the project "Interpretable methods of process diagnosis using statistics and machine learning" and by the Polish National Science Centre project "Process Fault Prediction and Detection", contract no. UMO-2021/41/B/ST7/03851.

**Data Availability Statement:** Data are contained within the article

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| GP | Gaussian process |
| MCMC | Markov chain Monte Carlo |
| INLA | integrated nested Laplace approximation |
| FFT | fast Fourier transform |
| GPR | Gaussian process regression |
| HMC | Hamiltonian Monte Carlo |
| GMRF | Gaussian Markov random field |
| RBF | radial basis function |
| SE | squared exponential |
| SPDE | stochastic partial differential equation |
| FITC | fully independent training conditional |
| PITC | partially independent training conditional |
| BKB | budgeted kernelized bandit |

## References

1. Davis, R.A. Encyclopedia of Environmetrics, Gaussian Process. In *Encyclopedia of Environmetrics*; American Cancer Society: Hoboken, NJ, USA, 2006. [CrossRef]
2. Rasmussen, C.; Williams, C.K.I. *Gaussian Processes in Machine Learning*; MIT Press: Cambridge, MA, USA, 2006.
3. Dudek, A.; Baranowski, J. Gaussian Processes for Signal Processing and Representation in Control Engineering. *Appl. Sci.* **2022**, *12*, 4946. [CrossRef]
4. Zeng, A.; Ho, H.; Yu, Y. Prediction of building electricity usage using Gaussian Process Regression. *J. Build. Eng.* **2020**, *28*, 101054. [CrossRef]
5. Gogolashvili, D.; Kozyrskiy, B.; Filippone, M. Locally Smoothed Gaussian Process Regression. *Procedia Comput. Sci.* **2022**, *207*, 2717–2726. [CrossRef]
6. Das, S.; Roy, S.; Sambasivan, R. Fast Gaussian Process Regression for Big Data. *Big Data Res.* **2018**, *14*, 12–26. [CrossRef]
7. Rodrigues, F.; Pereira, F.; Ribeiro, B. Gaussian Process Classification and Active Learning with Multiple Annotators. In Proceedings of the 31st International Conference on Machine Learning, PMLR, Bejing, China, 22–24 June 2014; Xing, E.P., Jebara, T., Eds.; PMLR: New York, NY, USA, 2014; Volume 32, pp. 433–441.
8. Ibna Kaisar, T.; Zaman, K.; Khasawneh, M.T. A New Approach to Probabilistic Classification Based on Gaussian Process and Support Vector Machine. *Comput. Ind. Eng.* **2023**, *174*, 109719. [CrossRef]
9. Gonçalves, G.; Gomes, D.; Leoni, G.; Rosendo, D.; Moreira, A.; Kelner, J.; Sadok, D.; Endo, P. Optimizing the Cloud Data Center Availability Empowered by Surrogate Models. In Proceedings of the 53rd Hawaii International Conference on System Sciences, Maui, HI, USA, 7–10 January 2020; Hawaii International Conference on System Sciences; HICSS 2020. [CrossRef]
10. Sanabria-Borbón, A.C.; Soto-Aguilar, S.; Estrada-López, J.J.; Allaire, D.; Sánchez-Sinencio, E. Gaussian-Process-Based Surrogate for Optimization-Aided and Process-Variations-Aware Analog Circuit Design. *Electronics* **2020**, *9*, 685. [CrossRef]
11. Li, Y.; Zheng, Y. Citywide Bike Usage Prediction in a Bike-Sharing System. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1079–1091. [CrossRef]
12. Belda, S.; Pipia, L.; Morcillo-Pallarés, P.; Verrelst, J. Optimizing Gaussian Process Regression for Image Time Series Gap-Filling and Crop Monitoring. *Agronomy* **2020**, *10*, 618. [CrossRef]
13. Gönül, M.; Kutlar, O.A.; Calik, A.T.; Orcun Parlak, F. Prediction of oil dilution formation rate due to post injections in diesel engines by using Gaussian process. *Fuel* **2021**, *305*, 121608. [CrossRef]
14. Liu, H.; Ong, Y.S.; Shen, X.; Cai, J. When Gaussian Process Meets Big Data: A Review of Scalable GPs. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4405–4423. [CrossRef] [PubMed]

15. Liu, H.; Cai, J.; Ong, Y.S.; Wang, Y. Understanding and comparing scalable Gaussian process regression for big data. *Knowl.-Based Syst.* **2019**, *164*, 324–335. [CrossRef]
16. Bottou, L.; Chapelle, O.; DeCoste, D.; Weston, J. Approximation Methods for Gaussian Process Regression. In *Large-Scale Kernel Machines*; MIT Press: Cambridge, MA, USA, 2007; pp. 203–223.
17. Quiñonero-candela, J.; Rasmussen, C.E.; Herbrich, R. A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.* **2005**, *6*, 2005.
18. Gómez Rubio, V.; Rue, H. Markov Chain Monte Carlo with the Integrated Nested Laplace Approximation. *Stat. Comput.* **2018**, *28*, 1033–1051. [CrossRef]
19. Lindgren, F.; Rue, H. Bayesian spatial modelling with R-INLA. *J. Stat. Softw.* **2015**, *63*, 1–25. [CrossRef]
20. Van Niekerk, J.; Krainski, E.; Rustand, D.; Rue, H. A new avenue for Bayesian inference with INLA. *Comput. Stat. Data Anal.* **2023**, *181*, 107692. [CrossRef]
21. Lv, F.; Yang, G.; Zhu, W.; Liu, C. Generative classification model for categorical data based on latent Gaussian process. *Pattern Recognit. Lett.* **2017**, *92*, 56–61. [CrossRef]
22. Adams, R.P.; Murray, I.; MacKay, D.J.C. The Gaussian Process Density Sampler. In Proceedings of the 21st International Conference on Neural Information Processing Systems (NIPS'08), Red Hook, NY, USA, 8–10 December 2008; pp. 9–16.
23. Kawashima, T.; Hino, H. Gaussian Process Koopman Mode Decomposition. *Neural Comput.* **2023**, *35*, 82–103. [CrossRef] [PubMed]
24. Riutort-Mayol, G.; Bürkner, P.C.; Andersen, M.R.; Solin, A.; Vehtari, A. Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming. *arXiv* **2020**, arXiv:2004.11408. https://doi.org/10.48550/ARXIV.2004.11408.
25. Simpson, D. Un Garçon pas Comme les Autres (Bayes): Yes but What Is a Gaussian Process? or, Once, Twice, Three Times a Definition; or A Descent into Madness. 2021. Available online: https://dansblog.netlify.app/posts/2021-11-03-yes-but-what-is-a-gaussian-process-or-once-twice-three-times-a-definition-or-a-descent-into-madness/yes-but-what-is-a-gaussian-process-or-once-twice-three-times-a-definition-or-a-descent-into-madness.html (accessed on 20 November 2023).
26. Garnett, R. *Bayesian Optimization*; Cambridge University Press: Cambridge, UK, 2022.
27. Blum, M.; Riedmiller, M. Optimization of gaussian process hyperparameters using Rprop. In Proceedings of the ESANN 2013 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 24–26 April 2013; pp. 339–344.
28. Raes, W.; Dhaene, T.; Stevens, N. On The Usage of Gaussian Processes for Visible Light Positioning With Real Radiation Patterns. In Proceedings of the 2021 17th International Symposium on Wireless Communication Systems (ISWCS), Berlin, Germany, 6–9 September 2021; pp. 1–6. [CrossRef]
29. Baranowski, J.; Dudek, A.; Mularczyk, R. Transient Anomaly Detection Using Gaussian Process Depth Analysis. In Proceedings of the 2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR), Amber Baltic Hotel, Miedzyzdroje, Poland, 23–26 August 2021; pp. 221–226. [CrossRef]
30. Dudek, A.; Baranowski, J. Modelling of Li-Ion battery state-of-health with Gaussian processes. *Arch. Electr. Eng.* **2023**, *72*, 643–659. [CrossRef]
31. Gammelli, D.; Rodrigues, F.; Pacino, D.; Kurtaran, H.A.; Pereira, F.C. A Machine Learning Approach to Censored Bike-Sharing Demand Modeling. In Proceedings of the Transportation Research Board, Annual Meeting Proceedings, Transportation Research Board National Cooperative Highway Research Program, Walter E. Washington Convention Center, Washington, DC, USA, 12–16 January 2020; Volume 2020.
32. Zhang, L.; Liu, J.; Jiang, H.; Guan, Y. SensTrack: Energy-Efficient Location Tracking with Smartphone Sensors. *IEEE Sens. J.* **2013**, *13*, 3775–3784. [CrossRef]
33. Cheng, L.; Ramchandran, S.; Vatanen, T.; Lietzén, N.; Lahesmaa, R.; Vehtari, A.; Lähdesmäki, H. An additive Gaussian process regression model for interpretable non-parametric analysis of longitudinal data. *Nat. Commun.* **2019**, *10*, 1798. [CrossRef] [PubMed]
34. Lamb, A. A Brief Introduction to Generative Models. *arXiv* **2021**, arXiv:2103.00265. http://arxiv.org/abs/2103.00265.
35. Jarzyna, K.; Rad, M.; Piatek, P.; Baranowski, J. Bayesian Fault Diagnosis for Induction Motors During Startup in Frequency Domain. In *Lecture Notes in Networks and Systems, Proceedings of the Advanced, Contemporary Control—Proceedings of the XXI Polish Control Conference, Gliwice, Poland, 26–29 June 2023, Volume 2*; Pawelczyk, M., Bismor, D., Ogonowski, S., Kacprzyk, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2023; Volume 709, pp. 14–24. [CrossRef]
36. Trefethen, L.N. *Approximation Theory and Approximation Practice*; SIAM Philadelphia USA: Philadelphia, PA, USA, 2012; pp. I–VII, 1–305.
37. Ahmed, N.; Fisher, P. Study of algorithmic properties of chebyshev coefficients. *Int. J. Comput. Math.* **1968**, *2*, 307–317. [CrossRef]
38. Monterrubio-Gómez, K.; Wade, S. On MCMC for variationally sparse Gaussian processes: A pseudo-marginal approach. *arXiv* **2021**, arXiv:2103.03321. http://arxiv.org/abs/2103.03321.
39. Stan Development Team. Stan User's Guide. 2024. Available online: https://mc-stan.org/users/documentation/ (accessed on 5 January 2024).
40. Rue, H.; Martino, S.; Chopin, N. Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations. *J. R. Stat. Soc. Ser. B* **2009**, *71*, 319–392. [CrossRef]

41. Rue, H.; Held, L. Gaussian Markov Random Fields: Theory and Applications. In *Gaussian Markov Random Fields*; Chapman and Hall/CRC: New York, NY, USA, 2005; Volume 104. [CrossRef]
42. Krainski, E.; Gómez Rubio, V.; Bakka, H.; Lenzi, A.; Castro-Camilo, D.; Simpson, D.; Lindgren, F.; Rue, H. *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*; Chapman and Hall/CRC: New York, NY, USA, 2018. [CrossRef]
43. Mark Richardson. Chebfun and FFT Example. 2024. Available online: https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/23972/versions/22/previews/chebfun/examples/approx/html/ChebfunFFT.html (accessed on 6 November 2023).
44. Manzhos, S.; Ihara, M. Rectangularization of Gaussian process regression for optimization of hyperparameters. *Mach. Learn. Appl.* **2023**, *13*, 100487. [CrossRef]
45. Schürch, M.; Azzimonti, D.; Benavoli, A.; Zaffalon, M. Recursive estimation for sparse Gaussian process regression. *Automatica* **2020**, *120*, 109127. [CrossRef]
46. Bijl, H.; van Wingerden, J.W.; B. Schön, T.; Verhaegen, M. Online sparse Gaussian process regression using FITC and PITC approximations. *IFAC-PapersOnLine* **2015**, *48*, 703–708. [CrossRef]
47. Calandriello, D.; Carratino, L.; Lazaric, A.; Valko, M.; Rosasco, L. Gaussian Process Optimization with Adaptive Sketching: Scalable and No Regret. In *Proceedings of Machine Learning Research, Proceedings of the Thirty-Second Conference on Learning Theory, Phoenix, AZ, USA, 25–28 June 2019*; Beygelzimer, A., Hsu, D., Eds.; PMLR: New York, NY, USA, 2019; Volume 99, pp. 533–557.
48. Grosse, J.; Zhang, C.; Hennig, P. Optimistic Optimization of Gaussian Process Samples. *arXiv* **2023**, arXiv:2209.00895.

# Transient Anomaly Detection Using Gaussian Process Depth Analysis

Jerzy Baranowski, Adrian Dudek, Rafał Mularczyk
Department of Automatic Control & Robotics
AGH University of Science & Technology
Kraków, Poland
{jb,addudek,mularczyk}@agh.edu.pl

Abstract—Effective and reliable monitoring and diagnostics of process control installations is of utmost importance, as they are important part of world's economy. Detection of faulty behavior fits really well into this problem. Most of known results provides good techniques for analysing steady states, but have issues with the transients.

In this paper we propose a new method for detecting anomalies in transient states using combination of Gaussian Processes and data depth functions. We create a model of a transient signal and using its parameters also a model for candidate for fault analysis. Their similarity is verified using data depth. We present how such application works for a water tank system and different types of data depth.

Index Terms—transient state, fault detection, Gaussian Process, data depth.

## I. Introduction

Effective and reliable monitoring and diagnostics of process control installations is of utmost importance, as they are important part of world's economy. Algorithms for fault detection and isolation allow extension of system lifetime, reduction in operation interruption and can lead to significant savings. The main difficulty in their development is that process installations have a high level of complexity, are usually nonlinear and influenced by stochastic disturbances and parameter variations. Therefore, approaches based on first principles models are difficult or even impossible to use on a wider scale. That is why methods based on statistical models or machine learning are those most researched.

Typically, machine learning, data-driven models are providing complicated 'black-box' models, which are not transparent and hard to interpret. This is flaws are less prevalent in statistical approaches wich is a cause of their dominance of statistical approaches in the field. Both of those groups however suffer from the typical situation that the real data for system faults is extremely rare, and even if present is often incomplete. That is why it is crucial to develop methods that can handle issues of non-representative or missing data.

Work partially financed from AGH's Subvention for Scientific research and partially from AGH's Research University Excellence Initiative under project "Interpretable methods of process diagnosis using statistics and machine learning".

Regardless of the overall data quality, a matter that is generally consistent in process diagnostic analysis is the data character. Measurements are often heterogeneous, in the form of time series, with differing sampling periods and varying noise levels. Time series diagnostics are usually based on extracting features and with that reduction of dimensionality as signals are represented as vectors of numbers. Unfortunately usual approach to feature extraction is to obtain certain typical statistical measures (such like, mean, standard deviation, kurtosis, median, peak-to-peak) in both time and frequency domain, and hope that they will contain enough information about the signal [1], [2]. This negatively influences reliability and efficiency of diagnostic models as it is very hard to verify.

The field of fault detection and diagnosis is dominated by statistical methods, mostly based on variants of Principal Component Analysis or kernel methods. Machine learning methods are being investigated, but a consensus is yet to be reached, as methodologies range from kNN classifiers, through SVD to neural networks [3]–[7]. Probabilistic approaches either cover PCA [8], [9] or kernel methods like CVA [10], [11], with some notable exceptions for Fisher discriminant analysis [12], [13] and more advanced dynamic PCA and probabilistic mixtures [14], [15]. Regardless of selected methodology, most of research considers control charts for steady states. In case of transient diagnostics authors (e.g. [16]) mostly use autoregressive moving average models (or their variants). which is a significant limitation both because of linearity and non-locality (it is difficult to capture localized behavior).

In this paper we want to fill the gap in the transient detection using new approach consisting of:

- modelling of transient states (both healthy and suspect of faulty) using Gaussian Processes,
- using distribution obtained with a Gaussian Process analyze the data depth distribution for healthy and faulty signals
- difference in the depth is an anomaly indicator.

The rest of the paper is organized as follows. First we present basic theory of Gaussian Processes and data depth. Then we present the experimental setup and application

of proposed methodology to a problem of fault detection in water tank system. Paper ends with conclusions.

## II. Gaussian Process

Gaussian Process (GP) is a stochastic process used for modeling data, which were observed over time, space or both [17]. The main feature that characterize GP is that it is a generalization of normal probability distributions, where each of them describes a random variable (scalar or vector if we deal with multivariate distribution) [18]. What distinguish GP from other processes is its property to be fully determined by mean and covariance function [17] [18]. Usually, the mean function is fixed to zero, which is an easy condition to fulfill in most cases. Covariance function (also called kernel function) represents a similarity between data points. We chose it to reflects the prior beliefs about function, which is about to be predicted [19]. Kernel functions have free parameters called hyperparameters. They allow to flexible customization of the GP. In a process of learning, we should optimize these parameters for specific problem [18] [19].

According to Carl Edward Rasmussen, GP can be simplified to definition:

> A Gaussian process is a collection of random variables, any Gaussian process finite number of which have a joint Gaussian distribution [18].

Summing up, if we define mean function $m(x)$ and the covariance function mean function $k(x, x')$ of a real process $f(x)$ as:

$$m(x) = E[f(x)] \tag{1}$$

$$k(x, x') = E[f(x) - m(x))(f(x') - m(x'))] \tag{2}$$

Then the GP can be defined as:

$$f(x) \sim GP(m(x), k(x, x')) \tag{3}$$

Usage of GP is growing in popularity in statistic and machine learning due to ease of implementation and wide application possibilities, especially in issues which lacks of comprehensive data or when it is difficult to obtain them. Known examples of application are predicting with GP model (kriging) [20] and usage in Bayesian neural networks [21].

## III. Data depth

Tukey initiated the concept of data depth in 1975 (called Tukey depth, half space depth, or location depth) [22]. In the following years, it was expanded to include additional data depth types like Euclidean depth, $L^p$ depth, Mahalanobis depth, projection depth or Oja depth [23] [24].

The depth of the data was used to analyze the multivariate control charts. Such facilitation allows for easier control of multidimensional processes [25].

Another important element is improving the quality of nonparametric tests. [26] use different depth measurements to analyze the best and compare with known methods. The analyzed analysis considers single and multidimensional cases.

It relates much of the data depth to continuous functions that are not very realistic. Nagy and Ferraty [27] use functional data analysis to represent discontinuous data.

It was also used to analyze functional data [28] [29]. By analyzing function curves, it is much easier to detect distant data. In addition, finding anomalies is easier by using the data depth derived from the functional mean.

At the very beginning, let's introduce the definition of the depth function. It is the basis of each of the data depth methods.

Definition 1:

We define the data depth, the distance of the measurements from the center of the point $x \in R^d$ regarding a distribution function F. The outermost observations have lower values than the near-center data. They are determined by the depth function. This way, you can identify which data is anomalous throughout the process. First, let's start with the definition of the depth function.

Let the mapping $D(\cdot;\cdot) : \mathrm{R}^d \times \mathcal{F} \to \mathrm{R}^1$ be bounded, non-negative, and meet the assumptions:

1) $D(Ax+b; F_{AX+b}) = D(x; F_X)$ holds for any random vector $X$ in $\mathrm{R}^d$, any $d \times d$ nonsingular matrix $A$, and any $d$-vector $b$;
2) $D(\theta; F) = sup_{x \in \mathrm{R}^d} D(x; F)$ holds for any $F \in \mathcal{F}$ having center $\theta$;
3) for any $F \in \mathcal{F}$ having deepest point $\theta, D(x; F) \leq D(\theta + \alpha(x - \theta); F)$ holds for $\alpha \in [0, 1]$
4) $D(x; F) \to 0$ as $\|x\| \to \infty$, for each $F \in \mathcal{F}$.

Then $D(\cdot; F)$ is called a statistical depth function [30].

This article focuses on two types of data depth (Euclidean and Mahalanobis depth).

### A. Euclidean depth

The first and simplest method of determining data depth is the Euclidean depth. It is based on the Euclidean distance.

$$ED(x; F) = \left(1 + \|y - \bar{x}_j\|^2\right)^{-1} \tag{4}$$

where $\bar{x}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{ji}$

### B. Mahalanobis depth

Another method of determining the distance between two points $x$ and $y$ in $R^d$ was proposed by Mahalanobis

$$d_M^2(x, y) = (x - y)' M (x - y) \tag{5}$$

where $M$ is positive definite $d \times d$ matrix.

Based on the Mahalanobis distance, the depth function can be defined as

$$MHD(x;F) = \left(1 + d^2_{\Sigma(F)}(x, \mu(F))\right)^{-1} \quad (6)$$

where $\Sigma(F)$ is covariance matrix of $F$ and $\mu(F)$ is the mean.

## IV. Fault detection algorithm

Main purpose of the paper is to develop and use fault detection algorithm which combines Gaussian Process usage and Data Depth Analysis. At first, we divide data obtained during the experiment into two sets: normal and faulty case, based on whetever fault was simulated in a given run or not. We pass some healthy data to fit GP in order optimize our model and obtain hyperparameters. Then we sample more exemplary healthy data from GP for the purpose of calculating depth to determine the range in which the values of healthy data lie.

Using the same hyperparameters, which we obtain during optimisation on healthy data, we then fit GP again and generate enormous amount of new samples, this time based on whole sets of healthy and failure cases. For each sample we calculated depth data value and we plot histogram, all relative to healthy, exemplary samples. That makes it possible to compare obtained results to example healthy cases and classify individual process as a failure or a healthy one.

## V. Experiments

### A. Water tank system

The system under consideration is a set of three hydraulic tanks connected in a cascade. Each of them has a different shape: rectangular, trapezoidal and quarter-circular. It shows the diagram of the considered system in the figure. In order to keep the water in the system, there is an additional buffer tank. I am at the very bottom of the system under consideration. It is connected to the last of the tanks. A water pump is connected to it. After starting, it sets in motion the water flow in the system, filling the tanks from the top. The entire system is closed, which allows a continuous flow of water in the system. Water can flow through two valves installed between the individual tanks, which are at the bottom of each tank. These are manual valves (ensuring a continuous flow of water between tanks) and electromagnetic valves (simulating anomalies in selected tanks). Pressure sensors MPX2010GP provides water level (in indirect way). A dedicated measurement interface routes all signals from the system to the computer. The interface unit also acts as a power amplifier, allowing the user to control the speed of the pump and valves by changing the PWM duty cycle. The computer dedicated to the laboratory system is equipped with a universal RT DAC 4 PCI digital-to-analog I / O expansion card, which measures the analog water level signals and supplies the digital PWM control signal to the DC pump motor. Mounting the water height sensors in the tank, they move with the flow of water, which affects data collection. Causes noise and inconclusive readings. Using the MATLAB / Simulink environment, we calibrate the sensors before each system startup. This allows you to reduce the measurement error. The water level in the system is controlled. Each water defect is completed [31] [32].

Fig. 1. Photo of a system of three water tanks connected in a cascade. the entire system is connected to a computer that controls the pump and solenoid valves.

The figures show examples of a healthy water flow in the system (figure 2) and a run with some anomaly at some point in time (figure 3).

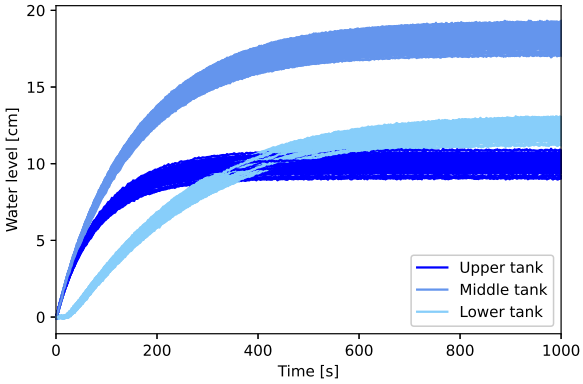



Fig. 2. An example of correct water flow through a system of water reservoirs. The water level in individual water reservoirs depending on the time is presented.

### B. Usage of Gaussian Process

Key concept of using GP in our experiment was to increase amount of possessed data in order to detect the faulty systems by data depth usage. It is worth to notice that count of generated data can be tremendous, compared to quantity of measured one, which is required to correct depth analysis.

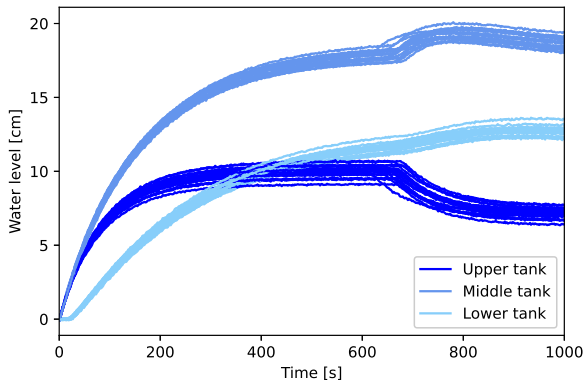



Fig. 3. An example of water flow through a system of water reservoirs with an anomaly. By using a solenoid valve, a failure was simulated on the upper tank. The water level in individual water reservoirs depending on the time is presented.

1) Data acquisition: At first, we analysed all measurements and transformed them if needed. Each data set comprises 20 cases of measured water level in three water tanks over time. For our case study, we focus on 2 data sets measured on the first tank where one consists of normal trajectories and second one of faulty ones. For the sake of appropriate data representation, we subtracted the average level of healthy measurements in each point of time from both data sets, in order to make the data represent deviation from average level instead of raw readings. From that transformed data set, we gathered 21 evenly distributed in time samples from each case and packed them into different subsets of data.

2) Gaussian Process Preparation: Before fitting any data, it is required to form a prior distribution. Here, the mean of GP is set to 0 and we have chosen the covariance function from a set of most used ones, which fits our prior believes. That is the Rational Quadratic kernel. It can be considered as a scale mixture (an infinite sum) of Radial Basis Function kernels with different characteristic length scales. The free parameters (hyperparameters), that describe this function are length-scale $l$ and scale mixture $\alpha$. Both hyperparameters were set to 1 before any optimizing.

3) Gaussian Process Fitting: To get the concrete results it is required to optimize kernel and its hyperparameters. The method we used to accomplish this task is a usage of algorithm, that minimize the calculated negative log likelihood. During training on healthy data set, optimizer equipped with Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LM-BFGS) algorithm was running in order to minimize mentioned log likelihood and receive kernel parameters. To avoid local minima, we restarted optimizer 20 times per one data subset. Calculated $l$ parameter was $1.95e{+}03$ and $\alpha$ equal to 0.0882.

We then executed fitting process on both healthy and faulty data sets using constant kernel hyperparameters obtained during the optimization process. In order to receive more exact values, we added little noise to samples. Exemplary trajectories of GP Regression with confidence interval against measured data and used samples can be seen on figure 4 for healthy case and on figure 5 for faulty ones, respectively.

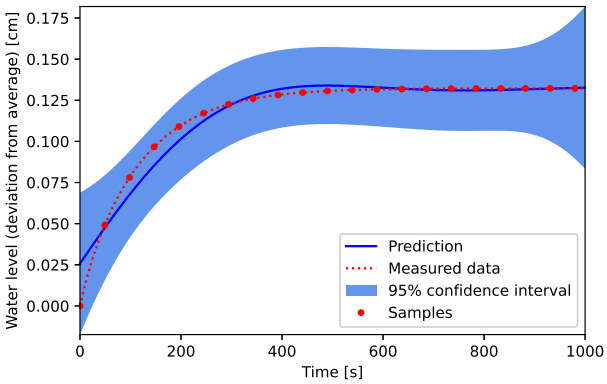



Fig. 4. Example of Gaussian Process fitted in healthy samples with usage of optimized hyperparameters for kernel. Red dots represent samples used in fitting process and red dashes the complete data set of measured data (the real trajectory of example). Blue line and field shows expected value (prediction) and confidence interval of predicted data. Prediction fits real trajectory almost perfectly, because kernel was optimized on healthy representations. At the end of plot we can see that confidence interval stretches. It is caused due to lack of data near the end of trajectory.

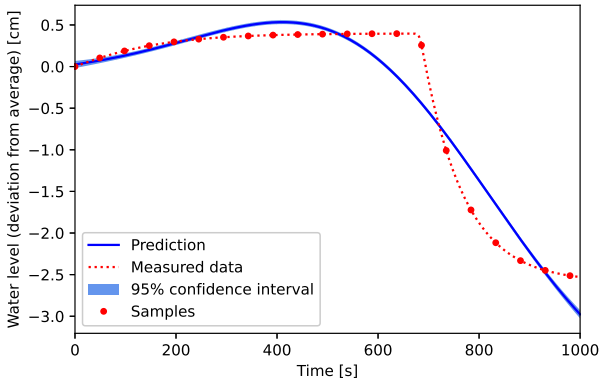



Fig. 5. Example of Gaussian Process fitted in faulty samples with usage of optimized hyperparameters for kernel. Red dots represent samples used in fitting process and red dashes the complete data set of measured data (the real trajectory of example). Blue line and field shows expected value (prediction) and confidence interval of predicted data. Prediction does not fit real trajectory with high accuracy and sometimes mismatches even out of shown confidence interval. This is caused due to set of hyperparameters, which were especially optimized to predict healthy trajectories, but it does not detract from the usefulness of GP for generating desired samples.

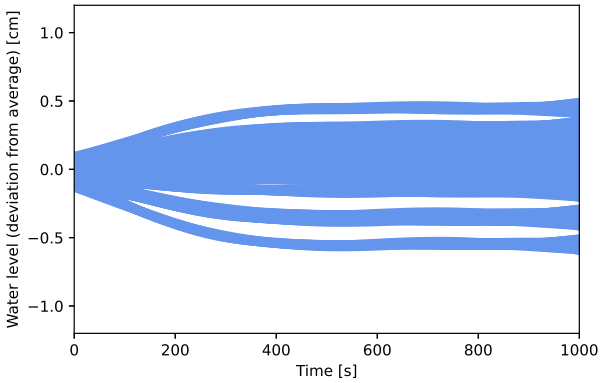



Fig. 6. Healthy cases trajectories used for data depths. Blue field represents only data, which were obtained by usage of Gaussian Process sampling from fitted models on 20 examples. The amount of trajectories, which were generated from only 21 samples per each of 20 healthy data subsets, comes to about 200 000. As we can see, all healthy data joint together does not deviate from mean over 0.5 cm during normal state.

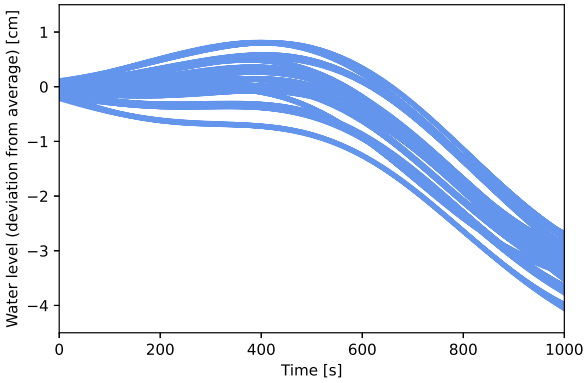



Fig. 7. Faulty cases trajectories used for data depths. Blue field represents only data, which were obtained by usage of Gaussian Process sampling from fitted models on 20 examples. The amount of trajectories, which were generated from only 21 samples per each of 20 faulty data subsets comes to about 200 000. The strong tendency to keep reasonable deviation from mean at the begging, just to fall into fault state after, is seen among all of processed data. This case shows that even with lack of faulty data faulty states can be predicted and trajectories generated in sufficient quantity with GP usage to make any deep analysis.

As seen, GP adapts prediction based on sampled data, to real trajectory, especially well for healthy cases. It does not represents prediction for faulty cases highly accurate, because the hyperparameters were calculated and optimized for only at healthy cases. At the and of examples the confidence interval stretches, because of lack of data.

4) Sampling from Gaussian Process: For each data subset, after its fitting, we realized GP main purpose and generated new samples from GP. More specifically, we

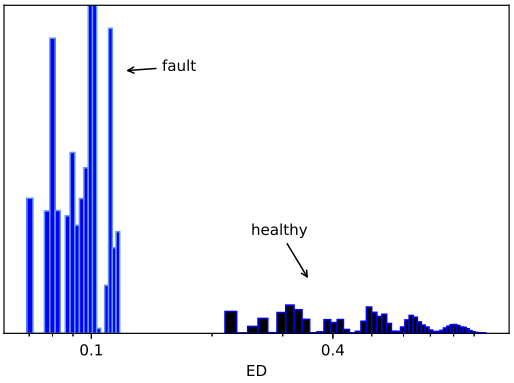



Fig. 8. Histogram of data depth values by Euclidean method. We presented values for healthy data and data with anomaly. There is a visible difference of values between the healthy values and those with an anomaly. The horizontal axis is presented in logarithmic form to show small values.

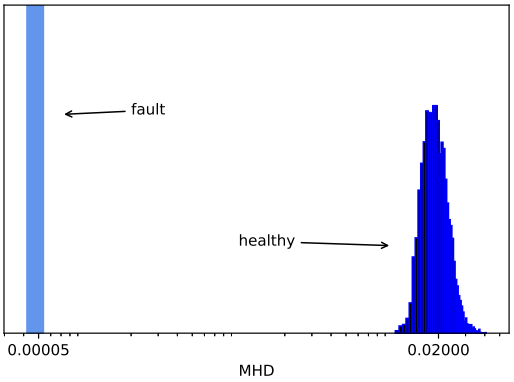



Fig. 9. Histogram of data depth values by Malahanobis method. We presented values for healthy data and data with anomaly. There is a visible difference of values between the healthy values and those with an anomaly. The horizontal axis is presented in logarithmic form to show small values.

created, for each data subset, 10000 trajectories with over 50 samples each. Plot of joint all together generated data is shown on figure 6 for healthy cases and on figure 7 for faulty ones.

## C. Fault detection using Data depth

Data depth methods were used to detect anomalies in the system. The figures 8 and 9 show the histograms of the depth values for the two selected methods. They illustrate the values for health data and data with anomaly. Thanks to this presentation of data, it is quite easy to distinguish data from each other. With the Mahalanobis depth, there is a significant difference in value. The depth values for the healthy data were around 0.02, while for the faults it was 0.00005. Using the Euclidean depths, the difference is

not that significant, but it is also easy to detect anomaly values from healthy data.

## VI. Conclusion

This paper includes very early stage of the research. Proposed approach shows promise but requires further testing. Main issue that was observed while working with depth statistics is that their values are not really interpretable. They are dimensionless and while they can be used for discrimination, increasing dimensions might increase sensitivity. Further research will include considering different depths and analysis of their properties. Regarding Gaussian Process models in this paper we have considered more machine learning type approach with point estimates of hyperparameters. Further work will use some kind of probabilistic computation in order to better capture the uncertainity.

## References


[1] A. Stief, R. Tan, Y. Cao, J. Ottewill, N. Thornhill, and J. Baranowski, "A heterogeneous benchmark dataset for data analytics: Multiphase flow facility case study," Journal of Process Control, vol. 79, pp. 41–55, 2019.

[2] A. Stief, J. R. Ottewill, J. Baranowski, and M. Orkisz, "A pca and two-stage bayesian sensor fusion approach for diagnosing electrical and mechanical faults in induction motors," IEEE Transactions on Industrial Electronics, vol. 66, no. 12, pp. 9510–9520, 2019.

[3] L. Meneghetti, M. Terzi, S. Del Favero, G. A. Susto, and C. Cobelli, "Data-driven anomaly recognition for unsupervised model-free fault detection in artificial pancreas," IEEE Transactions on Control Systems Technology, vol. 28, no. 1, pp. 33–47, Jan 2020.

[4] D. Jung, "Data-driven open-set fault classification of residual data using bayesian filtering," IEEE Transactions on Control Systems Technology, vol. 28, no. 5, pp. 2045–2052, 2020.

[5] S. Chen, J. Yu, and S. Wang, "One-dimensional convolutional auto-encoder-based feature learning for fault diagnosis of multivariate processes," Journal of Process Control, vol. 87, pp. 54 – 67, 2020.

[6] S. Cho and J. Jiang, "A fault detection and isolation technique using nonlinear support vectors dichotomizing multi-class parity space residuals," Journal of Process Control, vol. 82, pp. 31 – 43, 2019.

[7] J. Feng and K. Li, "Mrs-knn fault detection method for multirate sampling process based variable grouping threshold," Journal of Process Control, vol. 85, pp. 149 – 158, 2020.

[8] S. Zhang, C. Zhao, and F. Gao, "Incipient fault detection for multiphase batch processes with limited batches," IEEE Transactions on Control Systems Technology, vol. 27, no. 1, pp. 103–117, Jan 2019.

[9] Q. Jiang, X. Yan, and B. Huang, "Neighborhood variational bayesian multivariate analysis for distributed process monitoring with missing data," IEEE Transactions on Control Systems Technology, vol. 27, no. 6, pp. 2330–2339, Nov 2019.

[10] H. Luo, K. Li, O. Kaynak, S. Yin, M. Huo, and H. Zhao, "A robust data-driven fault detection approach for rolling mills with unknown roll eccentricity," IEEE Transactions on Control Systems Technology, vol. 28, no. 6, pp. 2641–2648, 2020.

[11] Y. Liu and M. Xie, "Rebooting data-driven soft-sensors in process industries: A review of kernel methods," Journal of Process Control, vol. 89, pp. 58 – 73, 2020.

[12] K. A. Palmer and G. M. Bollas, "Sensor selection embedded in active fault diagnosis algorithms," IEEE Transactions on Control Systems Technology, vol. 29, no. 2, pp. 593–606, 2021.

[13] Y. Zeng, Z. Jia, W. Liang, and S. Gu, "Fault diagnosis based on variable-weighted separability-oriented subclass discriminant analysis," Computers & Chemical Engineering, vol. 129, p. 106514, 2019.

[14] H. Kodamana, R. Raveendran, and B. Huang, "Mixtures of probabilistic pca with common structure latent bases for process monitoring," IEEE Transactions on Control Systems Technology, vol. 27, no. 2, pp. 838–846, March 2019.

[15] Y. Wan, V. Puig, C. Ocampo-Martinez, Y. Wang, E. Harinath, and R. D. Braatz, "Fault detection for uncertain lpv systems using probabilistic set-membership parity relation," Journal of Process Control, vol. 87, pp. 27 – 36, 2020.

[16] Y. Lin, U. Kruger, F. Gu, A. Ball, and Q. Chen, "Monitoring nonstationary and dynamic trends for practical process fault diagnosis," Control Engineering Practice, vol. 84, pp. 139–158, Mar. 2019.

[17] R. A. Davis, Encyclopedia of Environmetrics, Gaussian Process. American Cancer Society, 2006.

[18] C. Rasmussen and C. K. I. Williams, Gaussian Processes in machine learning. MIT Press, 2006.

[19] M. Blum and M. Riedmiller, "Optimization of gaussian process hyperparameters using rprop," 2013, pp. 339–344, cited By 26.

[20] C. Lataniotis, S. Marelli, and B. Sudret, UQLAB User Manual – Kriging (Gaussian process modelling), 07 2015.

[21] S. Hu, M. Lam, X. Xie, S. Liu, J. Yu, X. Wu, X. Liu, and H. Meng, "Bayesian and gaussian process neural networks for large vocabulary continuous speech recognition," vol. 2019-May, 2019, pp. 6555–6559, cited By 3.

[22] J. Tukey, "Mathematics and the picturing of data." Proceedings of the International Congress of Mathematicians, vol. 2, p. 523±531, 1975.

[23] Y. Zuo, "Multivariate trimmed means based on data depth," in Statistical Data Analysis Based on the L1-Norm and Related Methods, Y. Dodge, Ed. Basel: Birkhäuser Basel, 2002, pp. 313–322.

[24] K. Mosler, Depth statistics. Berlin: Springer, 2013, pp. 17–34.

[25] S. Idris, L. Wachidah, T. Sofiyayanti, and E. Harahap, "The control chart of data depth based on influence function of variance vector," Journal of Physics: Conference Series, vol. 1366, p. 012125, nov 2019.

[26] S. Chenouri, C. G. Small, and T. J. Farrar, "Data depth-based nonparametric scale tests," The Canadian Journal of Statistics / La Revue Canadienne de Statistique, vol. 39, no. 2, pp. 356–369, 2011.

[27] S. Nagy and F. Ferraty, "Data depth for measurable noisy random functions," Journal of Multivariate Analysis, vol. 170, pp. 95–114, 2019, special Issue on Functional Data Analysis and Related Topics.

[28] A. Nieto-Reyes and H. Battey, "A topologically valid definition of depth for functional data," Statistical Science, vol. 31, no. 1, pp. 61–79, 2016.

[29] I. Gijbels and S. Nagy, "On a general definition of depth for functional data," Statistical Science, vol. 32, no. 4, pp. 630–639, 2017.

[30] Y. Zuo and R. Serfling, "General notions of statistical depth function," The Annals of Statistics, vol. 28, no. 2, pp. 461–482, apr 2000.

[31] J. Baranowski, W. Bauer, M. Zagórowska, A. Kawala-Janik, T. Dziwiński, and P. Piątek, "Adaptive non-integer controller for water tank system," in Theoretical Developments and Applications of Non-Integer Order Systems. Springer, 2016, pp. 271–280.

[32] J. Baranowski and A. Tutaj, "State and parameter estimation in a hydraulic system — moving horizon approach," in Proceedings of 13th International Power Electronics and Motion Control Conference EPE-PEMC, Poznań„ 1–3 September 2008, pp. 1455–1462.

# Mixture Based Classifier Using Gaussian Processes for Induction Motor Diagnosis

Adrian Dudek, Kacper Jarzyna, and Jerzy Baranowski
*Dept. of Automatic Control and Robotics*
*AGH University of Kraków*
Kraków, Poland
addudek@agh.edu.pl, kjarzyna@student.agh.edu.pl, jb@agh.edu.pl

***Abstract*—This paper explores the use of Gaussian Processes (GPs) and Gaussian Mixture Models (GMMs) for diagnosing induction motors. GPs provide flexible, non-linear models that handle noisy data, while GMMs offer robust probabilistic classification by modeling data as mixtures of Gaussian distributions. By integrating Bayesian inference with GMMs and utilizing Stan for complex model management, the study enhances classification accuracy. Experimental data from induction motors under various conditions were analyzed, identifying patterns indicative of motor health. The approach, leveraging synthetic data generation, demonstrates effectiveness in proactive maintenance and fault detection, reducing downtime and costs.**

***Index Terms*—Gaussian Processes, Gaussian Mixture Models, Probabilistic Classification, Induction Motor Diagnostics, Bayesian Inference**

## I. Introduction

Statistical techniques play a crucial role in the diagnostics of induction motors by offering a quantitative and systematic way to detect potential operational issues. These techniques enable engineers and technicians to process and evaluate extensive data from the motor, gaining valuable insights into its performance, health status, and possible faults. The primary advantages of employing statistical methods for diagnosing induction motors include enhanced accuracy, reliability, and operational efficiency. By effectively analyzing data to pinpoint issues at an early stage, proactive maintenance and repairs can be planned, thus preventing severe breakdowns, reducing downtime, and lowering overall expenses. Furthermore, statistical methods help in understanding the underlying causes of problems, facilitating the development and application of more targeted and efficient corrective measures.

Probabilistic classification is an approach within statistics that involves predicting the probabilities of various possible categorical outcomes rather than making a single definitive classification. This method is particularly beneficial when the decision-making process requires knowledge about the certainty of predictions. Probabilistic classifiers, such as logistic regression [1], Gaussian Mixture Models (GMMs) [2], and neural networks [3], calculate the likelihood of each class based on the input features, providing a rich, quantified understanding of potential outcomes. In this article, the authors focus on the use of GMMs in terms of classification. For each model component, it was proposed to use the Bayesian inference model of Gaussian Processes (GPs).

GPs are powerful tools in machine learning and statistics, known for their remarkable flexibility and ability to model complex, non-linear relationships. These non-parametric Bayesian models are ideal for deriving complex functions from data and providing crucial uncertainty estimates. A GP posits a probability distribution over functions, ensuring that any finite collection of function values adheres to a joint Gaussian distribution. Essentially, GPs establish a prior distribution over functions, which is refined through Bayesian inference as additional data becomes available. This method allows GPs to effectively handle irregularly spaced and noisy data and to capture the underlying correlation structures within the data. The references for these capabilities include [4]–[6].

Moreover, GPs are employed in a variety of tasks including regression [7]–[9], classification [10], [11], and optimization [12], [13]. They have been successfully applied in diverse areas such as supporting bike-sharing systems [14], enhancing computer vision [15], and optimizing diesel engines [16]. A significant feature of GPs is their ability to yield probabilistic predictions, which effectively account for noise in the data. In our case, we are using the GP as a probabilistic classification for GMMs.

The contributions of our paper are the following:

- We constructed a Bayesian Gaussian mixture model, where mixing components are Gaussian Processes
- We inferred the parameters of the mixture using frequency characteristics of induction motor startup currents
- We used methods of recovering cluster probability to create a classifier allowing the recognition of faulty and healthy motors.

The rest of the paper is organized as follows: First, we describe the methods used, including Gaussian Processes, the Gaussian mixture model classifier, and the probabilistic programming language Stan. Next, we move on to the considered system, the data we have collected, and how we are augmenting it for learning. Finally, we present the results, describing our model structure exactly and providing the results of its operation. The paper ends with conclusions.

Work partially realised in the scope of project titled "Process Fault Prediction and Detection". Project was financed by The National Science Centre on the base of decision no. UMO-2021/41/B/ST7/03851. Part of work was funded by AGH's Research University Excellence Initiative under project "DUDU - Diagnostyka Uszkodzeń i Degradacji Urządzeń".

## II. Methods

### A. Gaussian Process

Gaussian Processes (GPs) are essential for understanding the stochastic processes used to model data observed over time, space, or both [4]. A GP is typically defined as a collection of random variables where any finite subset has a joint Gaussian distribution. This definition, widely accepted and utilized, emphasizes GPs as generalizations of normal probability distributions. Each component in a GP can describe a random variable, either scalar or multivariate. The universally recognized definition of a GP is stated as follows:

> *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

The behavior of a function $f$ modeled by a GP is fundamentally characterized by its mean function $m(x)$ and its covariance function $k(x, x')$, which are defined respectively as:

$$m(x) = E[f(x)] \tag{1}$$

$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))] \tag{2}$$

Consequently, a GP can be formally represented as:

$$f(x) \sim GP(m(x), k(x, x')) \tag{3}$$

Assuming $\boldsymbol{f} = \{f(\boldsymbol{x}_n)\}_{n=1}^{N}$, the prior distribution of $\mathbf{f}$ is a multivariate Gaussian distribution $\boldsymbol{f} \sim Normal(\boldsymbol{\mu}, \boldsymbol{K})$, where $\boldsymbol{\mu} = \{\mu(\boldsymbol{x}_n)\}_{n=1}^{N}$ and $\mathbf{K}$ is the covariance matrix, with $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The joint distribution of existing values $f$ and new predictions $\tilde{f}$ follows a multivariate Gaussian distribution, detailed as:

$$p(\boldsymbol{f}, \tilde{f}) = \text{Normal}\left(\begin{bmatrix} \boldsymbol{f} \\ f^* \end{bmatrix} \mid \mathbf{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{f},\boldsymbol{f}} & \boldsymbol{k}_{\boldsymbol{f},\tilde{f}} \\ \boldsymbol{k}_{\tilde{f},\boldsymbol{f}} & k_{\tilde{f},\tilde{f}} \end{bmatrix}\right) \tag{4}$$

where $\boldsymbol{k}_{\boldsymbol{f},\tilde{f}}$ denotes the covariance between $f$ and $\tilde{f}$, and $k_{\tilde{f},\tilde{f}}$ is the prior variance of $\tilde{f}$ [17].

The ability to make inferences from the GP model is bolstered by adding real-world data from the process being modeled. GPs are uniquely determined by their mean and covariance functions, where the mean is often set to zero to simplify the model unless specific prior knowledge suggests otherwise [5], [18]. The choice of the covariance function is crucial as it must generate a positive definite covariance matrix, and ideally, it should reflect the prior beliefs about the data's underlying processes [19], [20]. Designing and selecting an effective covariance function, however, can be challenging yet critical for the success of the GP model.

In our study, we are closely examining two special types of functions known as kernels: the Radial Basis Function (RBF) and the Matérn kernel. We chose these kernels because they have unique features that are very useful for our work, and they allow us to deeply understand both the theory behind them and how they can be used in practice.

The first one, the RBF kernel, is defined by the formula:

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right), \tag{5}$$

where $l$ is known as the characteristic length scale, which helps determine how much one data point influences another, and $d(\cdot, \cdot)$ measures the straight-line distance between two points $x_i$ and $x_j$. A key feature of the RBF kernel is that it mainly depends on how far apart the points are.

For functions that are smooth and can be differentiated multiple times, we use a special type of RBF kernel called the *Matérn kernel*. This kernel is a bit more complex and can be tailored more flexibly to fit the data we are studying. The Matérn kernel is described by:

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right), \tag{6}$$

where $K_\nu$ is a modified Bessel function, $l$ is the length scale parameter, and $\nu$ controls how smooth the function is. Common values for $\nu$ are $3/2$, for functions that are smooth enough to differentiate once, and $5/2$, for functions smooth enough to differentiate twice [5].

### B. Gaussian Mixture Model classifier

Gaussian Mixture Models (GMMs) are a sophisticated statistical tool widely used in the field of data classification due to their robustness and efficiency in modeling complex distributions. A GMM is a probabilistic model that assumes the data is generated from a mixture of several Gaussian distributions, each with its own mean and covariance. This model is particularly useful in scenarios where the data exhibits heterogeneity, suggesting the presence of multiple subpopulations within the overall dataset. Each component in a GMM can capture different statistical properties of the data, allowing the model to accommodate varied data patterns that simple Gaussian models might miss [21], [22].

In the context of classification, GMMs offer significant advantages. They provide a soft-clustering approach, assigning probabilities to each data point for belonging to one of the Gaussian components, rather than forcing a hard classification. This probabilistic assignment is crucial in real-world applications where data may not be distinctly separable into clear classes. For instance, in image processing or speech recognition, where the boundaries between different categories can be blurred, the flexibility of GMMs allows for more nuanced classifications. Moreover, the parameters of the GMM (means, variances, and mixture coefficients) are fitted through the model based on the observed data [21], [22].

In the context of this paper, we consider that each of the mixture components is a Gaussian process with its own distribution of hyperparameters. In such a case, each of the mixture components is its own functional representation of signals, either faulty or healthy, and class probability can be recovered.

### C. Stan

Stan is a robust and innovative computational framework primarily for Bayesian inference, born out of the need to man-

age complex models and improve the accuracy and efficiency of statistical analyses. It is built on a foundation that utilizes sophisticated algorithms like Hamiltonian Monte Carlo (HMC) and the No-U-Turn Sampler (NUTS), which are advanced forms of Markov Chain Monte Carlo (MCMC) methods [23]. These algorithms are particularly effective at navigating the complex geometries of high-dimensional parameter spaces often encountered in statistical models, ensuring that sampling is both thorough and efficient. In our case, we used the HMC approach to sampling.

The development of Stan was driven by the statistical community's requirement for a more accessible, scalable, and efficient way to apply Bayesian methods to real-world problems. It was designed to be user-friendly, incorporating a high-level programming language that simplifies the definition of complex models and makes Bayesian analysis more accessible to non-statisticians. Stan's programming language allows users to write their models once and then fit them with any of the multiple algorithms available, facilitating a broad range of statistical analyses without the need for algorithm-specific coding [23].

Stan also supports a variety of interfaces known as 'wrappers', which cater to users familiar with different programming environments. Among the most popular are RStan for R users or CmdStanPy for Python environments. CmdStanPy, which was used by us, allows Python users to run Stan models by interfacing directly with CmdStan, the command-line interface of Stan. These interfaces significantly expand the accessibility of Stan, allowing it to be seamlessly integrated into the data analysis workflows of users across various platforms.

## III. Considered system

### A. System

Our work centers on the diagnostic assessment of induction motors during startup, prioritizing the identification of anomalous behaviors over specific waveform characteristics. We utilized a computational setup comprising four identical induction motors (seen in figure 5), each subjected to varying degrees of damage. Specifically, we considered motors in healthy condition, with single broken rotor bars, and with two broken rotor bars. Pairing the motors enhanced inertia moments and facilitated comparative analysis. The setup can be seen in the figure. Each motor is a closed construction and a supply voltage of 400 V in star configuration and 230 V in delta configuration. Manufactured by Indukta1, these motors had a rated power of 8.4 kW in delta configuration and 4.8 kW in star configuration, with a declared nominal efficiency of 82.0.

### B. Collected data

Each data sample consists of startup currents under different voltage conditions (ranging from 100 V to 150 V) and was acquired using standard acquisition cards and LEM transducers. The sampling frequency remained consistent at 1150.78 Hz throughout the experiments.

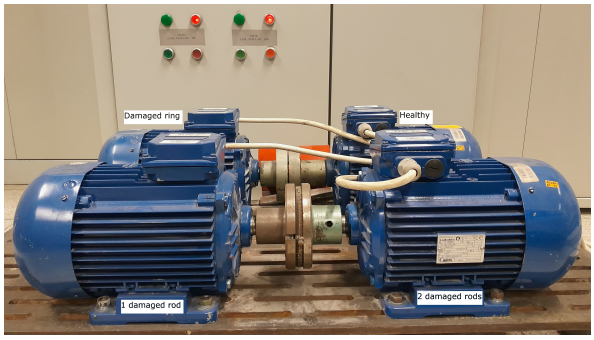



Fig. 1. Setup of four motors used for data collection. One of them was healthy, and the rest was damaged in different ways: broken rotor bar, two rotor bars broken, broken end ring. In this analysis, we focused on the broken rod diagnostics.

To account for variations induced by different startup voltages, we employed spectrogram analysis to unify the data in the frequency domain. Spectrograms consistently revealed distinct patterns indicative of motor health status in the lower frequency bands, as seen in figure 2.

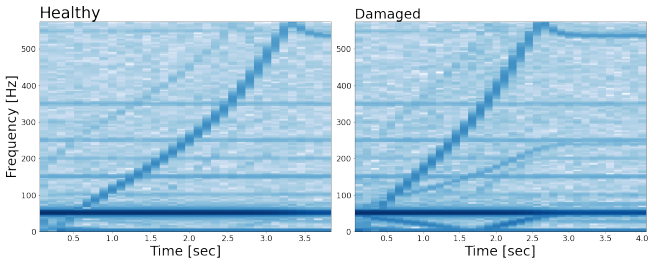



Fig. 2. Comparison of spectrograms of two example unfiltered signals. Components caused by rotational speed and supply frequency are identical in both healthy and damaged signals, however, we can see differences in the lower band.

A Notch filter was implemented to isolate relevant features and mitigate interference, specifically targeting the dominant 50 Hz component originating from the mains. The filter, designed using MATLAB/Simulink's Filterbuilder tool, was implemented as a 6th order system in SOS Direct Form II. Post-filtering, we employed FFT to focus only on lower band frequency responses within the 50 Hz. It exhibited notable disparities between healthy and damaged motors, as illustrated in figure 3.

### C. Supplementing data

In many real-world scenarios, data scarcity poses a significant challenge, limiting the effectiveness of traditional machine learning approaches. This limitation is particularly pronounced in our domain, where the available dataset is relatively small. To overcome this challenge and enable comprehensive analysis, we employ a generative modeling approach.

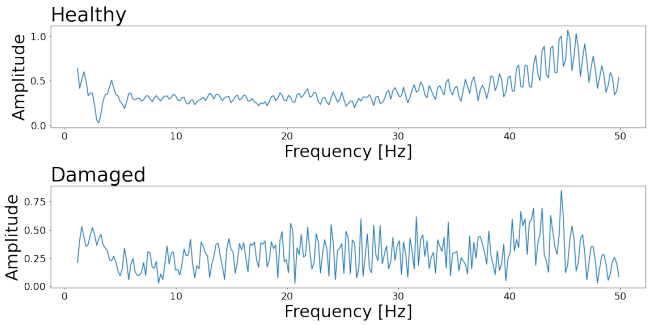



Fig. 3. Comparison of Fourier transforms of two example signals, after filtration and cutting off frequencies over 50 Hz.

Our data generation process is described by the following equation:

$$\begin{aligned} Y_n &\sim \text{Normal}(\mu_n, \sigma) \\ \mu_n &= \sum_{k=1}^{K} \beta_k \phi_k(t_n) \end{aligned} \tag{7}$$

Here, $Y_n$ represents magnitudes at selected frequencies, $n = 1, \ldots, N$, with uncertainties modeled by a normal distribution with standard deviation $\sigma$. The mean $\mu_n$ is expressed as a linear combination of B-splines $\phi_k(t_n)$ over $K$ knot points. This formulation captures both the distribution of individual measurements and the underlying structure of the data through the B-spline basis functions.

To address the challenge of data scarcity, we utilize a generative model implemented using Stan. This approach allows us to create synthetic frequency responses, thereby augmenting our dataset with additional samples for training and testing purposes. By generating synthetic data, we can mitigate the risk of overfitting while ensuring that all signals are adequately represented in our analysis.

The efficacy of our methodology is demonstrated through ribbon plots, as depicted in Figure 4. These plots visually represent the results of model training and illustrate the robustness of our approach in handling data scarcity and facilitating comprehensive analysis across multiple data types.

## IV. Results

### A. Model structure

In Stan, we implemented GPs by specifying a covariance function, which is essential for defining the shape and smoothness of the data generating process. One common approach is to use the Cholesky decomposition of the covariance matrix. This method is numerically stable and computationally efficient, making it ideal for handling the large matrices often encountered in GP modeling. The Cholesky factorization helps in transforming the problem into a lower-dimensional space, which simplifies sampling and improves the efficiency of the inference process. The covariance matrix itself is constructed based on a chosen kernel function, which defines the relationships between different points in the data.

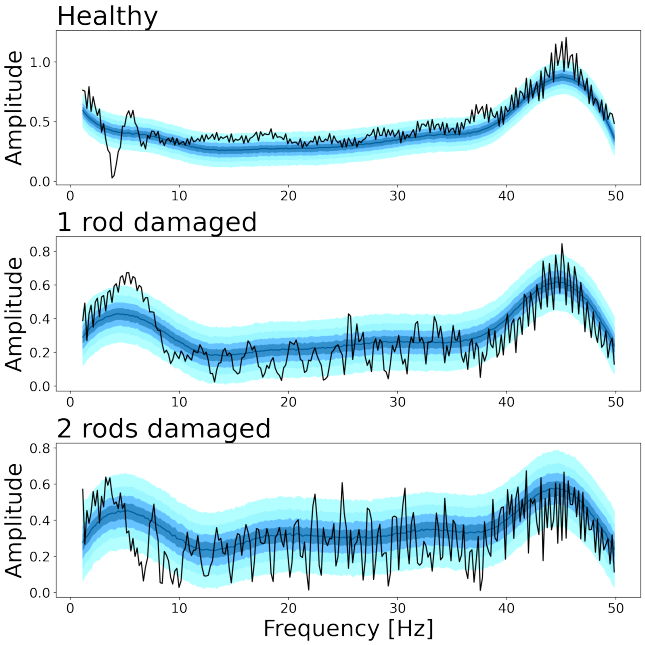



Fig. 4. To generate diverse data for training the classifier, we constructed models representing the magnitude response of each signal class using spline representation. We assumed a normal distribution for each spectrum. The figure showcases ribbon plots for different quantiles, with the median displayed in the center. Additionally, to provide a comparison, an example of the Fast Fourier Transform (FFT) from one of the signal classes is included.

Finite mixture models, particularly Gaussian Mixture Models, are used in Stan to model data that is assumed to be generated from several sub-populations, each described by its own distinct statistical distribution. In GMMs, each component of the mixture can be modeled as a Gaussian distribution with its own mean and variance. Stan handles these models effectively through its probabilistic programming language, allowing the specification of both the mixture components and the mixing proportions.

In Stan, constructing a GMM involved defining the mixture components as parameters and specifying the log-likelihood for each component in the mixture with a multinormal distribution. Since Stan optimizes computation by working with continuous parameters and log probabilities, it handles the discrete nature of mixture assignments implicitly by marginalizing over these discrete parameters. This marginalization increases the computational efficiency, especially important in models with a large number of mixture components or complex hierarchical structures.

With the samples obtained from Stan, it's possible to estimate the expected values of various functions of random variables, facilitating responses to a broad spectrum of statistical inquiries. Labeled data serve to initially shape the components of the mixture distribution, whereas unlabeled data help refine these structures and deepen understanding of the population's characteristics. The developed classifier undergoes testing with real data. In this research, the classifier was primarily trained

using data generated from the described generative model, supplemented by a small portion of random data samples from each category. Additionally, a mix of largely generated and some real unlabeled data was introduced to further bolster the classifier's ability to adapt. Each data category included a total of 30 samples, with only 2 being employed in the training process.

Training a Bayesian mixed model essentially involves sampling 4000 random mixed models from the conditional distribution provided by the available data, using 1000 samples for each of four Markov chains. For each model, and for the signal (frequency response), the probability of reconstructed class is computed. This procedure enables the calculation of all relevant statistics concerning the distribution of predictions.

### *B. Results*

The GMM using GP for classification, as implemented through the Stan framework, was fitted and tested on the dataset. Employing a total of 4000 samples generated by Stan for each chain to produce probabilistic classifications, the algorithm was primarily trained on synthetically generated data, which closely mimics real-world scenarios because these were generated with a spline model learned on real data. Moreover, to simulate real-world scenarios, we also added to the training set unlabeled data to represent the population as a whole. The test dataset consisted of real data to test the model for identifying healthy and damaged conditions, specifically targeting one or two damaged rods. Data were used in the frequency domain with a sampling rate of 50 samples in the range of 50Hz. The RBF and Matern3/2 kernels were used for GP while testing.

For each sample, the classifier outputs the probability of belonging to a particular class—either healthy, one rod damaged, or two rods damaged. Given the relatively small dataset utilized for these tests, results were presented as percentage probabilities for each sample across multiple runs of the classifier to ensure statistical significance and reliability of the outcomes. This approach helps in accounting for variability and potential biases in the model due to the limited data size.

The result of such classification is depicted in the plot (figure 5) below, which illustrates the probabilistic nature of the GMM outputs. Each point on the graph represents a specific sample's probability, distributed across the two possible conditions - healthy or damaged. The classifier distinguished between the different states of rod damage, highlighting its robustness and accuracy in handling small, complex datasets. This visual representation provides intuitive insight into the distribution of probabilities, facilitating easier interpretation and verification of the model's performance. As GP are Bayesian in nature, it is for the best to interpret results as distributions, with expected values with theirs 95% confidence intervals. When we consider that, we can not only declare the classification for the case, but also the confidence of such classification as damaged or healthy case. Following excamples shows that the Mixture Model is confident when labeling healthy data, but in case of damaged examples it sometimes considers them as possible healthy cases and with few examples as healthy cases. As a Bayesian inference it its better to consider that this way, which acknowledging confidence interval for the result and not only consider numerical metrics calculated based on the expected value.

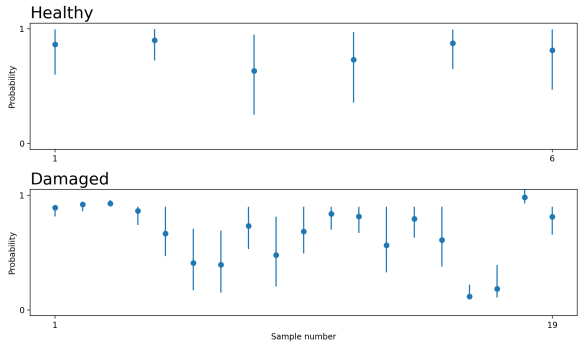



Fig. 5. The plot displays probabilistic classifications for a set of samples categorized into two classes: healthy and damaged. Each plot shows the probability distribution of each sample belonging to its respective class across multiple tests. The blue dot represents the mean values, while the blue bar presents the 95% confidence interval

## V. Conclusions

In this paper, we have presented a new option for the classification of faults, i.e., a mixture model using Gaussian Processes. Our results are obviously very preliminary. The main advantage of this kind of solution lies in handling uncertainty coming from unbalanced data. At this moment we can see how such uncertainty is visible in model predictions.

At this stage of research, we see few main directions for improvement:

- Computational inefficiency - multiple mixture components in the form of GPs require multiple Cholesky factorizations, which even with modest size can blow up during sampling.
- Proper sample size for classification with GPs - at the moment it is more or less trial and error, a more systematic approach is needed.
- Increase the number of categories in the mixture and increase the classifier's efficiency.

All these issues will be a focus of future work, and certain ideas will be introduced in order to handle them, including, among others, approximations of GPs with Gaussian mean random fields or using appropriate approximations in Hilbert spaces. Also, information criteria can be used to improve the modeling representation.

## References

[1] M. Maalouf, "Logistic regression in data analysis: An overview," *International Journal of Data Analysis Techniques and Strategies*, vol. 3, pp. 281–299, 07 2011.

[2] I. Prabakaran, Z. Wu, C. Lee, B. Tong, S. Steeman, G. Koo, P. J. Zhang, and M. A. Guvakova, "Gaussian mixture models for probabilistic classification of breast cancer," *Cancer Research*, vol. 79, no. 13, p. 3492–3502, Jul. 2019. [Online]. Available: http://dx.doi.org/10.1158/0008-5472.CAN-19-0573

[3] P. Zhang, "Neural networks for classification: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 30, pp. 451 – 462, 12 2000.

[4] R. A. Davis, *Encyclopedia of Environmetrics, Gaussian Process*. American Cancer Society, 2006.

[5] C. Rasmussen and C. K. I. Williams, *Gaussian Processes in machine learning*. MIT Press, 2006.

[6] A. Dudek and J. Baranowski, "Gaussian processes for signal processing and representation in control engineering," *Applied Sciences*, vol. 12, no. 10, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/10/4946

[7] A. Zeng, H. Ho, and Y. Yu, "Prediction of building electricity usage using gaussian process regression," *Journal of Building Engineering*, vol. 28, p. 101054, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S235271021930662X

[8] D. Gogolashvili, B. Kozyrskiy, and M. Filippone, "Locally smoothed gaussian process regression," *Procedia Computer Science*, vol. 207, pp. 2717–2726, 2022, knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022.

[9] S. Das, S. Roy, and R. Sambasivan, "Fast gaussian process regression for big data," *Big Data Research*, vol. 14, pp. 12–26, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214579617301909

[10] F. Rodrigues, F. Pereira, and B. Ribeiro, "Gaussian process classification and active learning with multiple annotators," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 433–441. [Online]. Available: https://proceedings.mlr.press/v32/rodrigues14.html

[11] T. Ibna Kaisar, K. Zaman, and M. T. Khasawneh, "A new approach to probabilistic classification based on gaussian process and support vector machine," *Computers & Industrial Engineering*, p. 109719, 2023.

[12] G. Gonçalves, D. Gomes, G. Leoni, D. Rosendo, A. Moreira, J. Kelner, D. Sadok, and P. Endo, "Optimizing the cloud data center availability empowered by surrogate models," 01 2020.

[13] A. C. Sanabria-Borbón, S. Soto-Aguilar, J. J. Estrada-López, D. Allaire, and E. Sánchez-Sinencio, "Gaussian-process-based surrogate for optimization-aided and process-variations-aware analog circuit design," *Electronics*, vol. 9, no. 4, 2020.

[14] Y. Li and Y. Zheng, "Citywide bike usage prediction in a bike-sharing system," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 02 2019.

[15] S. Belda, L. Pipia, P. Morcillo-Pallarés, and J. Verrelst, "Optimizing gaussian process regression for image time series gap-filling and crop monitoring," *Agronomy*, vol. 10, no. 5, 2020. [Online]. Available: https://www.mdpi.com/2073-4395/10/5/618

[16] M. Gönül, O. A. Kutlar, A. T. Calik, and F. Orcun Parlak, "Prediction of oil dilution formation rate due to post injections in diesel engines by using gaussian process," *Fuel*, vol. 305, p. 121608, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016236121014897

[17] G. Riutort-Mayol, P.-C. Bürkner, M. R. Andersen, A. Solin, and A. Vehtari, "Practical hilbert space approximate bayesian gaussian processes for probabilistic programming," 2020. [Online]. Available: https://arxiv.org/abs/2004.11408

[18] R. Garnett, *Bayesian Optimization*. Cambridge University Press, 2022, in preparation.

[19] M. Blum and M. Riedmiller, "Optimization of gaussian process hyperparameters using rprop," 2013, pp. 339–344, cited By 26.

[20] W. Raes, T. Dhaene, and N. Stevens, "On the usage of gaussian processes for visible light positioning with real radiation patterns," in *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, 2021, pp. 1–6.

[21] H. Wan, H. Wang, B. Scotney, and J. Liu, "A novel gaussian mixture model for classification," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 3298–3303.

[22] K. J. Lee, L. Guillemot, Y. L. Yue, M. Kramer, and D. J. Champion, "Application of the gaussian mixture model in pulsar astronomy - pulsar classification and candidates ranking for the fermi 2fgl catalogue: Application of the gaussian mixture model," *Monthly Notices of the Royal Astronomical Society*, vol. 424, no. 4, p. 2832–2840, Jul. 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1365-2966.2012.21413.x

[23] Stan Development Team, "Stan user's guide," https://mc-stan.org/users/documentation/, 2024, accessed: 05.01.2024.

ARCHIVES OF ELECTRICAL ENGINEERING VOL. 72(3), pp. 643–659 (2023)

DOI 10.24425/aee.2023.146042

# Modelling of Li-Ion battery state-of-health with Gaussian processes

ADRIAN DUDEK, JERZY BARANOWSKI ✉

*Department of Automatic Control and Robotics, AGH University of Science and Technology Kraków, Poland*

*e-mail: {addudek/✉jb}@agh.edu.pl*

(Received: 22.09.2022, revised: 13.04.2023)

**Abstract:** The problem of lithium-ion cells, which degrade in time on their own and while used, causes a significant decrease in total capacity and an increase in inner resistance. So, it is important to have a way to predict and simulate the remaining usability of batteries. The process and description of cell degradation are very complex and depend on various variables. Classical methods are based, on the one hand, on fitting a somewhat arbitrary parametric function to laboratory data and, on the other hand, on electrochemical modelling of the physics of degradation. Alternative solutions are machine learning ones or non-parametric ones like support-vector machines or the Gaussian process (GP), which we used in this case. Besides using the GP, our approach is based on current knowledge of how to use non-parametric approaches for modeling the electrochemical state of batteries. It also uses two different ways of dealing with GP problems, like maximum likelihood type II (ML-II) methods and the Monte Carlo Markov Chain (MCMC) sampling.

**Key words:** lithium-ion batteries, state-of-health, Gaussian process, diagnostics

## 1. Introduction

Lithium-ion batteries are currently standard for energy storage in portable consumer electronic devices such as mobile phones, notebooks, tablets and cameras [1]. Batteries are a complex chemistry and material system, and their electrical properties will change as they are used over time [2]. For example, their capacity will decrease, and their resistance will increase. Every improve of their capacity and lifespan requires major improvements in its design and chemistry [3]. In the case of such widespread use and drawback of constant wear, there is a need to predict its state-of-health (SoH).

The diagnostic of the SoH of a lithium battery is necessary to ensure the reliability and safety of the battery energy storage system and it also can minimize maintenance cost [4]. Battery aging is important to consider when predicting the battery's state of health, as it can show reduced capacity and increased resistance. The estimate of the remaining useful life is a key indicator for the management and performance of the battery and can also be seen as the time that elapses from now until the end of the battery's total useful life [5]. The SoH parameter is expressed as the ratio of the current cell capacity to its initial capacity (new cell), usually expressed in percentage. Most often, a cell is considered to be worn out when the SoH is below 80% [31].

Battery parameter prediction is mainly divided into methods based on physical-mathematical models (model-based) and data-driven models [32]. The model-based method considers the batteries aging as an empirical model process or as an electrical/electrochemical model. Unfortunately, it is a challenging task to develop a reasonable model that balances estimation accuracy and computational cost (due to the complex electrochemical process) [27]. That's why we often turn to a data-driven model.

Predicting lithium-ion battery degradation during design and operation is a significant challenge. Machine learning models have such a problem because the problem of black-box limitations still exists. Many mechanisms cause degradation, which may interact in various ways. A variety of factors affect it, including calendar time, high power usage, low-temperature usage, and combinations of these. However, battery end users can only measure a limited range of battery parameters. It is difficult to prioritize the mechanisms that drive ageing for a battery and use case [6].

There were several ways to predict battery health using data-driven methods such as: neural networks [7], support vector machines [8] and the Gaussian process [9–11, 29]. The approach proposed in our paper is based on a GP model in order to diagnose the SoH of the battery. We used an open-source dataset from the NASA Ames Research Centre regarding batteries, from which we selected features for GP fitting. Our GP models are based on two approaches of calculation: ML-II and MCMC. The first one comprises maximizing log-likelihood to optimize hyperparameters, and the second one is based on sampling the distributions. We also considered the multi-kernel approach of the GP to check the improvement of the results.

The novelty of our work consists in specific approaching and applicating Gaussian processes for use in the SoH estimation of Li-Ion batteries. The very concept of Gaussian processes has been well known for a long time, but it has been gaining popularity in recent years [23]. However, numerous ways of its implementation, variations and use are still under extensive research. [28, 30, 31]. As we mentioned, our proposal is based on the use of Markov Chains to sample data using the model we created earlier using STAN. This approach is not often used in cases of implementation of solutions using Gaussian processes [23]. For comparison, we also use python library implementations based on the aforementioned ML-II solution, which requires much less preparation.

Our motivation was to create an algorithm that would be able to diagnose the SoH of batteries using partial and time estimations of battery capacity changes based on the Gaussian process. The database used allowed for the study of a wide range of batteries applied in different conditions. In addition, various settings (including the mentioned multikernel approach) allowed us to check the quality of solutions for various initial assumptions. Moreover, we could examine the general

quality of solutions less frequently used in GP implementations (like MCMC) and broaden the knowledge of the topic and encourage its further exploration.

The rest of the paper is organized as follows: first, we present a basic theory of the GP. Then, we discuss the dataset used in detail, as well as present the GP models. The next section introduces the results of our experiments and comparison of methods. The paper ends with conclusions.

## 2. Gaussian process

According to Rasmussen and Williams [12], the Gaussian process (GP) can be defined as *a collection of random variables, any Gaussian process finite number of which have a joint Gaussian distribution*. Such a definition captures the essence of the Gaussian process. If we have a real process $f(x)$, then we can define the mean function $m(x)$ and the covariance function $k(x, x')$ with the use of the following formulas:

$$m(x) = E\left[f(x)\right], \tag{1}$$

$$k(x, x') = E\left[f(x) - m(x)f(x') - m(x')\right]. \tag{2}$$

Then the definition of the GP can be formulated as:

$$f(x) \sim GP\left(m(x), k(x, x')\right). \tag{3}$$

There is only one way to fully determine the GP, which is to declare the mean and covariance function [13, 14]. The mean in most cases is set to "0", because such a setting can be useful, simplify matters and is not difficult to fulfill. There are times when we would like to change the mean e.g., for a better model interpretability or the specification of our prior, but most of the time it can be left as 0 [12, 14]. The covariance function (also called the kernel function) represents a similarity between data points [15]. Usually, covariance is chosen from a set of already defined functions. In general, we should choose at least one that represents prior beliefs about the problem. However, any function that generates a positive definite covariance matrix is acceptable as a covariance function [16]. Moreover, it can be a challenging task to create new covariance functions that have practical value but are also correct.

The most basic and common kernel function is the Radial Basis Function (RBF). It is also known as the squared exponent (SE) and is defined by Eq. (4):

$$k\left(x, x\right) = \exp\left(-\frac{d\left(x, x\right)^2}{2l^2}\right). \tag{4}$$

Its main feature is that its value is usually only dependent on the distance from the specified point. The kernel used by the RBF algorithm is based on the length scale of $l$. Also $d(\cdot, \cdot)$ is the Euclidean distance. The RBF is infinitely differentiable. That means the GP with this kernel has a mean square derivative for all orders, despite that the RBF is very smooth. However, strong smoothness assumptions are unrealistic, and using the kernel from the Matérn family is recommended [12].

In the cases of battery health diagnostics and overall modeling, one of the most used kernels, besides for instance the RBF, are those from the Matérn family. The Matérn kernel family is a kind

of generalization of the RBF function. The general formula for Matérn covariance functions is given by Eq. (5):

$$k\left(x\ ,x\ \right)=\frac{1}{\Gamma(\nu)2^{\ -1}}\left(\frac{\sqrt{2\nu}}{l}d\left(x\ ,x\ \right)\right)^{\ }K^{\ }\left(\frac{\sqrt{2\nu}}{l}d(x\ ,x\ )\right), \tag{5}$$

where: $K$ is the modified Bessel function, $l$ is the length-scale parameter, $d(\cdot,\cdot)$ is the Euclidean distance and $\nu$ is the parameter, which allows one to change the smoothness of the function, giving an ability to flexibly control the function in relation to the one we want to model. These kernels are frequently used with parameter $\nu$ values of 3/2 applied in learning functions, which are at least once differentiable and 5/2 for functions at least twice differentiable [12].

In the case of modeling the electrochemical state of batteries, it is worth mentioning the Rational Quadratic (RQ) kernel, which can be seen as a scale mixture (an infinite sum) of RBF kernels with different characteristic length scales. Hyperparameters describing RQ are the length scale $l$ and the scale mixture parameter $\alpha$. The RQ kernel function is given by Formula (6):

$$k\left(x\ ,x\ \right)=\left(1+\frac{d\left(x\ ,x\ \right)^{2}}{2\alpha l^{2}}\right)^{-}\ , \tag{6}$$

where in addition to hyperparameteres, $d(\cdot,\cdot)$, as previously, is the Euclidean distance.

Each of the selected covariance functions has a set of free parameters called hyperparameters. These values need to be determined in order to use the covariance function. Handling this task can be one of the main difficulties in using the GP. The ML-II method is widely used at present, and we calculate the parameters by edge likelihood optimization. Its advantage is that it is easy to use and the calculation process is not complicated, and its main disadvantage is its low precision. For example, it can suffer from the problem of multiple local maxima. There are also solutions to calculate hyperparameters such as statistical methods like the MCMC sampling of the declared prior distribution [12]. We used both approaches to solve the problem of diagnosis of battery cells to gain the SoH result, but more importantly, to research the benefits and drawback of each techniques, as well as future possibilities.

The GP is an approach that is quickly gaining popularity in recent years. The algorithm works best for problems with high dimensions, small samples, and nonlinearity [17]. The GP is a probabilistic technique for nonlinear regression that computes a posterior degradation estimate by constraining the prior distribution to fit the available training data. Recently, the GPR method has been favored by researchers because it is a probabilistic prediction model under the Bayesian framework [18]. Moreover, the most distinctive about the GP is that it provides not only the prediction but also a confidence interval as well as the distribution of all calculated values [19].

## 3. Data and methods

### 3.1. Battery dataset

The general problem of every lithium-ion battery diagnostics is the data related to it, more specifically, its overall lack of large, reliable, and comprehensive sets of data [20]. Some studies connected to this field of interests have made use of an open-source dataset from the NASA AMES

research center, and much of this data shows an approximately constant degradation rate [21], so it could be easily used as a starting point in our research. Because of the aforementioned data issue, we also used one of the NASA resources, which is randomized battery usage dataset [22].

The dataset consists of several *18650 Li-Ion batteries* (28 in total), which were divided into 7 groups, based on the factors in which they were charged and discharged. Specifics about differences in load patterns can be seen in Table 1.

Table 1. Overview of load patters in NASA AMES battery dataset

| Group | Description |
|---|---|
| 1 | – Repeatedly charged to 4.2 V using a **randomly** selected duration<br>– Discharged to 3.2 V using a **randomized** sequence of discharging currents<br>– Randomized sequence of discharging currents between 0.5 A and 4 A<br>– Duration of charging between 0.5 h and 3 h<br>– Room temperature |
| 2 | – Repeatedly charged to 4.2 V using a **preselected** duration<br>– Discharged to 3.2 V using a **preselected** sequence of discharging currents<br>– Sequences of discharging currents between 0.5 A and 4 A<br>– Room temperature |
| 3 | – Operated using a sequence of charging/discharging currents<br>– Sequences between –4.5 A and 4.5 A<br>– Each loading period lasted 5 min<br>– Room temperature |
| 4 | – Repeatedly charged to 4.2 V using a **randomly** selected duration<br>– Discharged to 3.2 V using a **randomized** sequence of discharging currents<br>– Randomized sequence of discharging currents between 0.5 A and 5 A<br>– Customized probability distribution skewed towards selecting **higher** currents<br>– Room temperature |
| 5 | – Repeatedly charged to 4.2 V using a **randomly** selected duration<br>– Discharged to 3.2 V using a **randomized** sequence of discharging currents<br>– Randomized sequence of discharging currents between 0.5 A and 5 A<br>– Customized probability distribution skewed towards selecting **lower** currents<br>– Room temperature |
| 6 | – Repeatedly charged to 4.2 V using a **randomly** selected duration<br>– Discharged to 3.2 V using a **randomized** sequence of discharging currents<br>– Randomized sequence of discharging currents between 0.5 A and 5 A<br>– Customized probability distribution skewed towards selecting **lower** currents<br>– Ambient temperature set to 40°C |
| 7 | – Repeatedly charged to 4.2 V using a **randomly** selected duration<br>– Discharged to 3.2 V using a **randomized** sequence of discharging currents<br>– Randomized sequence of discharging currents between 0.5 A and 5 A<br>– Customized probability distribution skewed towards selecting **higher** currents<br>– Ambient temperature set to 4°C |

The dataset contained information about the start time of the experiment for each battery cell, as well as measurements of the current and voltage of the load pattern with time stamps. We used data from all 7 groups. The dataset was then divided into cycles, each of which ended with a check of the battery's remaining capacity. During the cycle, cells were charged or discharged according to the parameters based on the selected scenario, or the parameters were not used at all (details in Table 1, more information can be found in the dataset itself: Bole *et al*., 2014 [22]). From the whole dataset we extracted 3 features from each cycle, based on literature [20]:

- total time of battery life: $\boldsymbol{T}$,
- time of selected cycle: $\boldsymbol{t}$,
- charge throughput during cycle (which is the integral of current over time): $\boldsymbol{I}$.

Based on that information, our GP model is supposed to calculate an output $\boldsymbol{y}$ capacity difference $\boldsymbol{\Delta Q}$ during a single cycle.

### 3.2. Gaussian process model

In order to create the model, we took into account the principles of the Gaussian process from section 2 and tried to use them considering the state-of-health diagnostic based on the dataset from section 3.1. Our first approach consists of the ML-II method, which is one of the most widely used methods of GP implementation [23]. We optimize the parameters by using a sort of likelihood function. Its advantage is that it is easy to set up and has a cheaper computation process, in exchange for overall accuracy and a less informative model. For some applications, it can experience a problem with having multiple local maxima [23].

To create a model using the ML-II method, we implemented the Gaussian processes from the python scikit-learn library. This implementation allowed us to create a GP model, which can use predefined kernel functions such as: the RBF, Matern family and RQ. Considering the fact that these kernels (especially Matern one) seem succesfull in research of battery diagnostics [9], we had the full possibility of testing this approach. Additionally, we could use some other kernels as points of reference. In this approach we were only able to pass information about kernels and optionally set the starting point of hyperparameters, which will be optimized automatically during the process with the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LM-BFGS) optimizer. As a result of fitting, we receive only predicted values corresponding with the standard deviation.

In the second approach, we developed other methods which do not rely on marginal likelihood maximizing. We created an STAN model of the GP and used it with the Markov Chain Monte Carlo (MCMC) method of sampling distributions. The main drawback of such an approach is the requirement of additional and specific knowledge, as well as longer computation time, but in return, we gain access to full information about the distribution of our results [24]. This approach requires building a model from scratch, with all assumptions about prior distribution, kernel matrix operations, generating quantities, etc. It is also the best way to code expert knowledge about the problem into our model. The returned distributions should allow us to come up with more complex conclusions and future use, which compensate for the difficulties of use.

A detailed description of the implementation and use of both algorithms can be found in Fig. 1, in the form of a diagram. Details and theoretical background can be found in C.E. Rasmussen and C.K.I. Williams, Gaussian processes for Machine Learning [12].

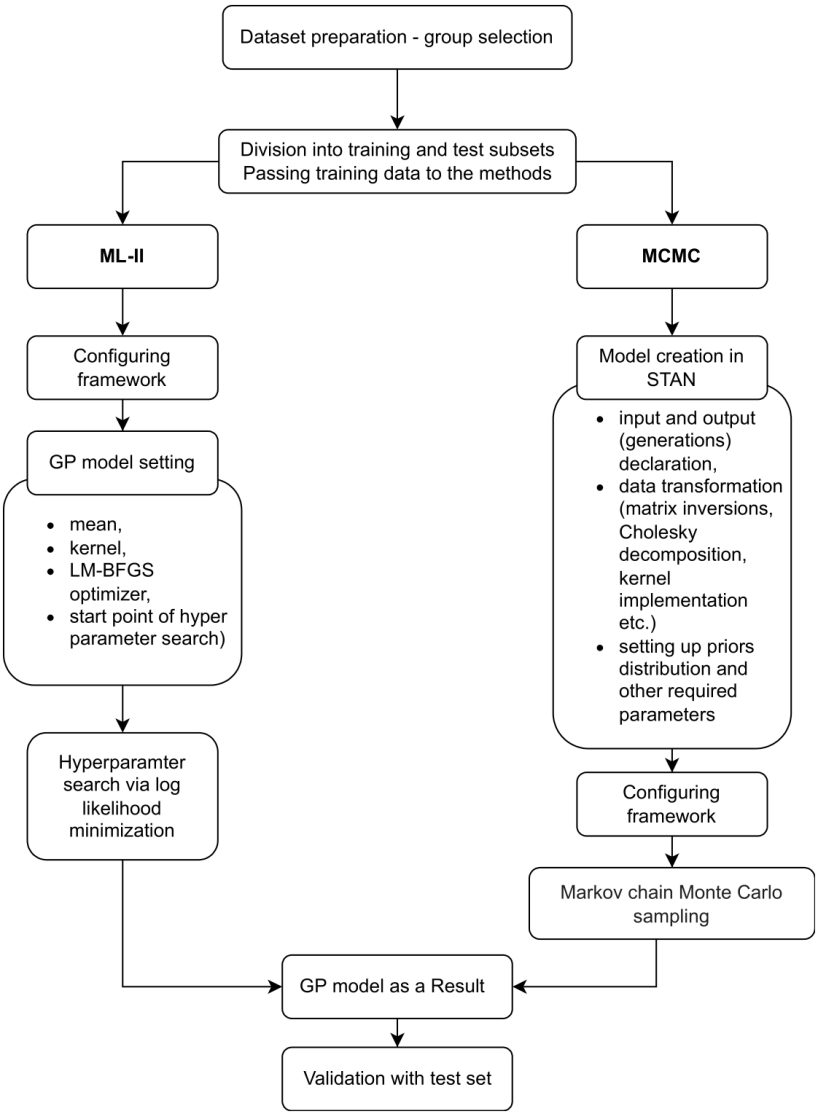



Fig. 1. Diagram showing the course of the algorithm divided into parts ML-II and MCMC

Setting the framework means setting all the necessary tools to work together in a specific structure, e.g. as a python script. ML-II, after providing the data, only requires setting the initial model values (average, hyperparameters) and optional settings for the optimizer. MCMC, on the other hand, requires creating the entire model manually with all the necessary data transitions (e.g. operations on the covariance matrix) and setting the initial distributions for the priorities. In addition, the necessary elements for samples generation. The sampling process is time-consuming but needs to be performed only once per model.

# 4. Results

## 4.1. ML-II approach

In order to evaluate the created model with this method we decided to use 4 different kernels: the RBF, Matern (3/2 and 5/2), RQ and Exp-Sine-Squared (ESS). The scikit-learn framework also allows one to easily combine different kernels into a multi-kernel approach, so we also considered that in our tests. The entire workflow consists of some principals. First, as the dataset relies on 7 different scenarios of load patterns, we created separate GP models for each scenario, which were then fitted using training data from each group. Validation of the models is done based on test data. Then we consider checking examples of batteries in terms of SoH diagnostic based on the learned model outcomes for a specific group. SoH prediction is based on knowledge of the battery capacity at the start and information about its life cycle necessary to select features and perform calculations. The model predicts changes in capacity for each cycle measurements (between moments in time when we collect data according to which we perform calculations with the GP model) and then the difference is subtracted from the previously known (estimated) capacity, which allows us to diagnose a current SoH.

The results of the model performance based on the test data set can be seen in Fig. 2. It presents several plots, with different kernels, where the relationship between the prediction of the capacity change and the measured value is shown. Red dots represent the mean value of prediction when blue bars show a 95% confidence interval. The method we chose to evaluate the models was to calculate the Root-Mean-Square Error (RMSE) value of prediction for each group and the whole test set, respectively. The concrete results of the RMSE for the total set are provided in Table 2.

Additionally, for the best-performing kernels, we decided to test the multi-kernel approach. The scikit-learn framework allows one to combine kernels by simply adding them to each other, with maintaining the ability to autotune hyperparameters. The results for the mentioned multi-kernels are shown in Fig. 3, as well as the RMSE values in Table 2.

The results show that the multi-kernel approach increases the accuracy of the overall model and the multi-kernel model themselves are at a similar level of accuracy. For the best-performing

Table 2. Total RMSE results of GP models with ML-II approach

| RMSE | Kernel |
|---|---|
| 0.096 | ESS |
| 0.057 | RQ |
| 0.185 | White |
| 0.096 | RBF |
| 0.058 | Ma 3/2 |
| 0.065 | Ma 5/2 |
| 0.055 | Ma 3/2 + RQ |
| 0.058 | Ma 3/2 + Ma 5/2 |
| 0.061 | Ma 3/2 + Ma 5/2 + RBF |
| **0.054** | Ma 3/2 + Ma 5/2 + RQ |

(in terms of RMSE) model, we created the SoH prediction, which is based on the knowledge of a basic battery capacity and then predicts its changes based on usage in the considered intervals. Example plots for the batteries from different groups can be seen in Fig. 4.

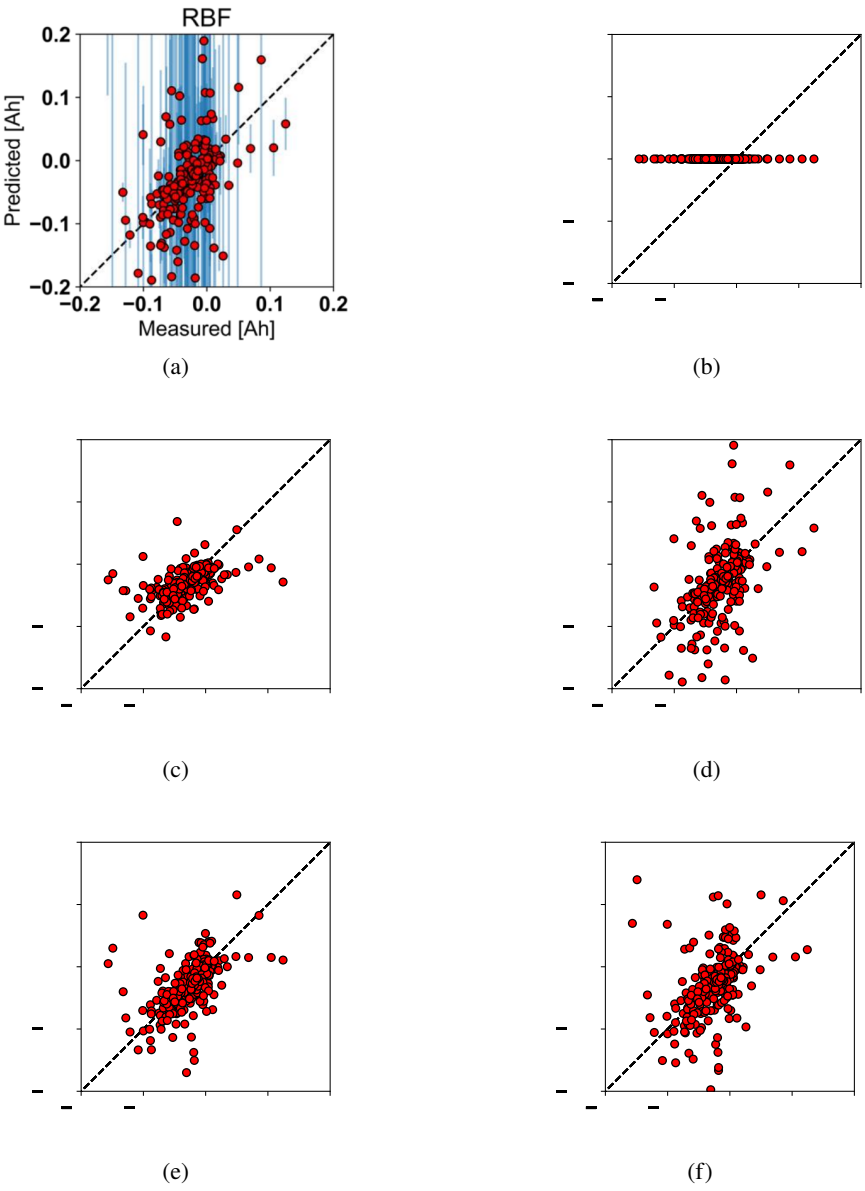
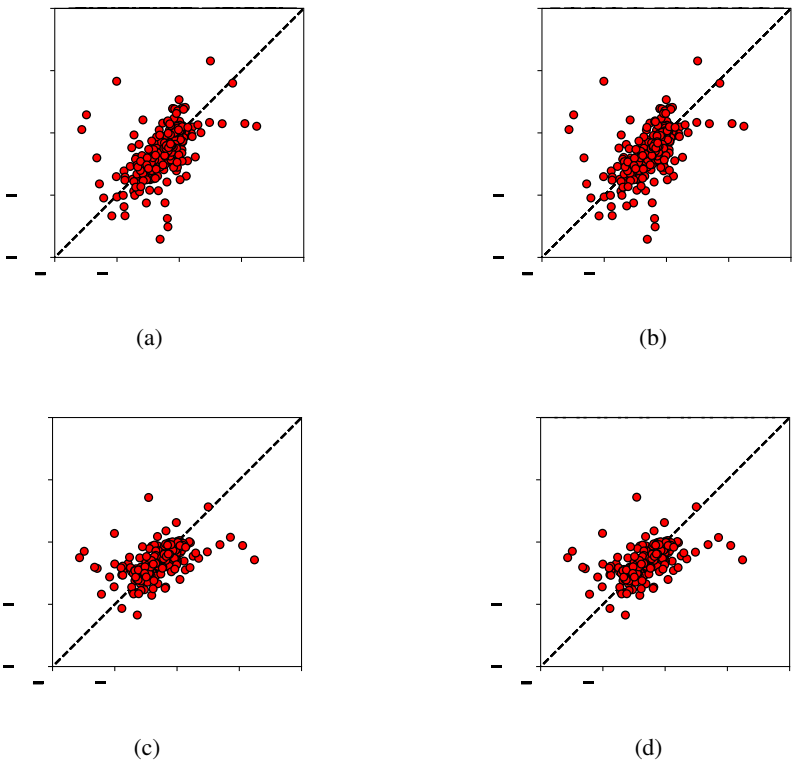


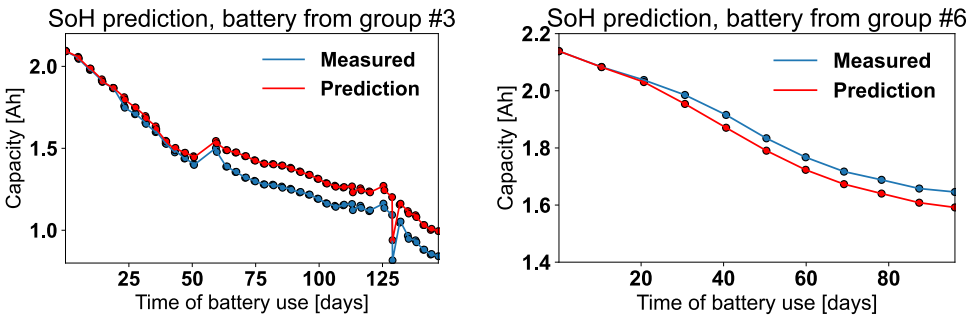

Fig. 2. Results of GP models with different kernels Radial-Basis Function (a); white kernel (b); Rational Quadratic (c); Exp-Sine-Squared (d); Matern 3/2 (e) and Matern 5/2 (f); plot represents prediction of model in relation to measured values; red dots are mean of prediction and the closer they are to the black dotted line, the smaller deviation from real values is; blue bars represent 95% confidence interval for each sample



(a) (b)

(c) (d)

Fig. 3. Results of GP models with different multikernes: sum of Matern family (a); sum of both Materns and Radial Basis Function (b); sum of Matern 3/2 and Rational Quadratic (c) and sum of both Materns and Rational Quadratic (d); plot represents prediction of model in relation to measured values; red dots are mean of prediction and the closer they are to the black dotted line, the smaller deviation from real values is; blue bars represent 95% confidence interval for each sample

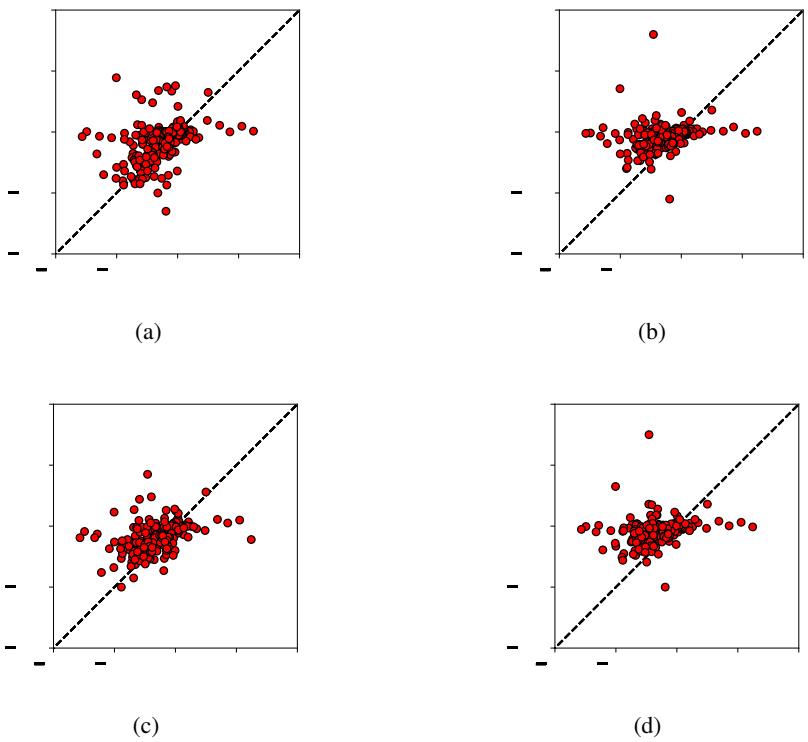



(a) (b)

Fig. 4. Results of SoH prediction with created GP model with ML-II approach: red dots show the predicted values, while blue ones are measured values; left plot (a) shows example battery from group 3 and right plot (b) shows example battery from group 6 (which has ambient temeperature set to 40°C)

### 4.2. MCMC sampling approach

Contrary to the ML-II approach, the MCMC sampling is characterized by the need to create a separate Stan model. The model is based on the hierarchical model approach, where each of the three levels consists of posteriors over parameters, hyperparameters and model structure. This requires setting up our priors for each of them. The overall accuracy depends on the correctness of their selection. A detailed description of the hierarchical model can be found in [12]. The model we created took data in the form of feature vectors and then created a covariance matrix corresponding to the selected kernel from them. Additionally, to speed up the calculations, we used the Cholesky decomposition mechanism. Initial settings for hyperparameters were set based on generally used distributions based on examples and applications of GP modeling in the Stan as well as our predictions. The output of our system was a generated distribution of predicted values for the test set. We created two sets of priors (one based on example usage and second set with our slight changes). The results for the RBF and Matern kernel with the test set can be seen in Fig. 5, as well as the RMSE results for expected values of posterior distribution in Table 3. Additionaly, we present the SoH prediction for example batteries in Fig. 6.

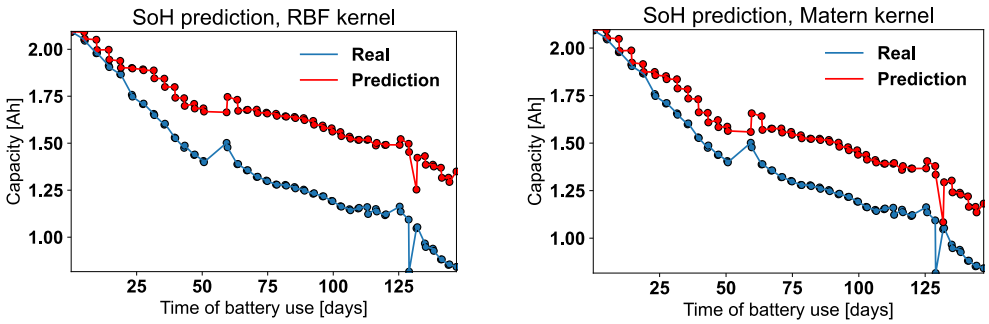


(a) (b) (c) (d)

Fig. 5. Results of GP model with MCMC sampling: RBF kernel with first set of hyperparameters prior (a); RBF kernel with second set of hyperparameters prior (b); Matern 3/2 kernel with first set of hyperparameters prior (c) and Matern 3/2 kernel with second set hyperparameters prior (d); plot represents prediction of model in relation to measured values; red dots are mean of prediction and the closer they are to the black dotted line, the smaller deviation from real values is; blue bars represent 95% confidence interval for each sample

Table 3. RMSE results for expected values of GP models with MCMC sampling

| RMSE | Kernel | Prior set |
|---|---|---|
| 0.153 | RBF | 1 |
| 0.079 | RBF | 2 |
| 0.183 | Ma 3/2 | 1 |
| 0.184 | Ma 3/2 | 2 |




(a) (b)

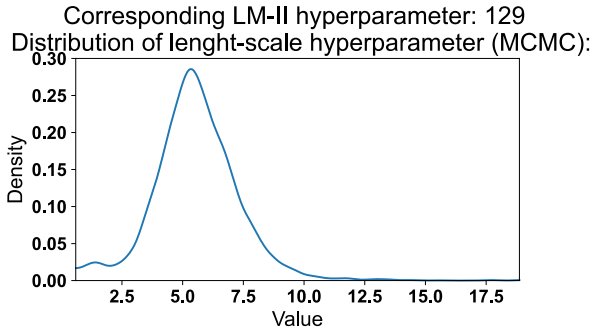
Fig. 6. Results of SoH prediction with created hierarchical GP model with MCMC sampling; red dots show the predicted values, while blue one are measured values; left plot (a) shows example battery predicted with RBF kernel and right plot (b) with Matern 3/2 kernel

First of all, it seems clear that the general prior assumption we used, does not fit SoH diagnostics very well. Secondly, the RMSE calculated over the expected values from posterior distribution is not the best way to measure model accuracy, because even though RBF kernels seem to get a better score, they still have worse results then the predicted example battery, as well as have higher uncertainty than Matern kernels, which can be seen in the plot in Fig. 4. The hierarchical model resulted in much more information about the problem, with slightly worse accuracy of prediction at the cost of a more complicated setup and higher computing time. As the setup of priors was not done perfectly, there is still room to improve it in future research.

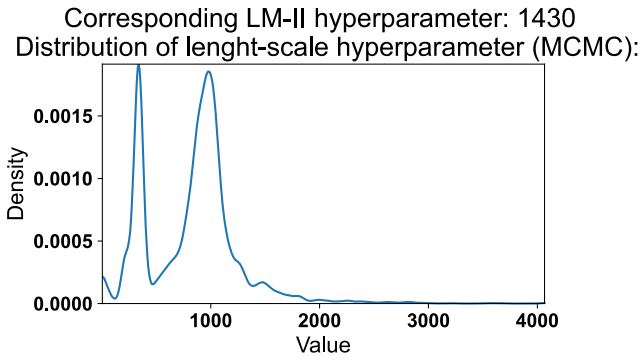
### 4.3. Comparison of presented approaches

First of all, it seems normal to compare the accuracy of both methods in terms of RMSE value, but it seems to be correct mainly in the ML-II solution, where firstly you can see solid differences between kernels, and secondly, this tool is mainly based on returning predictions, where the MCMC sampling returns to us more of the entire posterior of the model because this is its main advantage. The RMSE for the second case was calculated based on the expected values and not the whole distributions. In addition, the parameters in the case of ML-II were autotuned, where for the Stan model they were limited by the selection of distributions from the base GP-use cases, and not adapted to a given problem.

Because of it, hyperparameters resulting from autotuning in the case of the RBF and Matern differ from the mean value of the posterior distribution. Figure 7 shows the comparison of distributions of length-scale hyperparameters for the same example cases. The mean value to variant (a) is 5.5, while for variant (b) is 830. At the same time the ML-II method tuned the same parameter to 129 in variant (a) and 1 430 in variant (b). We also have to be aware that the Stan model uses an additional parameter called *magnitude* [25], which even further precludes a direct comparison of the two methods. Here, we will only focus on a comparison of length-scale hyperparameters, as these are common for both approaches.




(a)

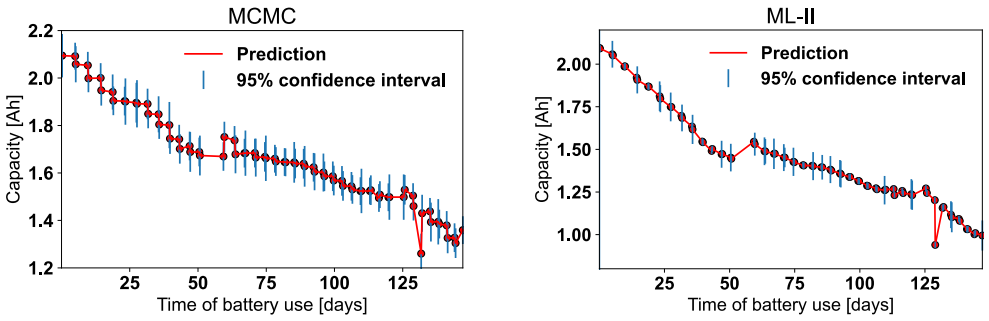



(b)

Fig. 7. Plots represent posterior distribution of length-scale hyperparameter aquired from model with MCMC sampling method; top plot (a) shows the distribution for case of Matern 3/2 kernel for third group from battery dataset (mean value: 5.5), correspoing value from ML-II method for this case is 129; bottom plot (b) shows the distribution for case of RBF kernel for fifth group from battery dataset (mean value: 830), correspoing value from ML-II method for this case is 180

Additionally, the MCMC sampling provides us with a more consistent confidence interval, while the scikit-learn framework sometimes has even some trouble catching the uncertainty. This can be noted in the plots in Figs. 2, 3 and 5, where we can see that the confidence interval is not consistent across all predictions, while in the case of the MCMC sampling, it remains constant for

the whole result (each sample). We can also better visualise that in the plot of the SoH diagnostic of the example battery when drawing uncertainty of prediction, as shown in Fig. 8.




(a) (b)

Fig. 8. Uncertainty of SoH prediction for example battery with hierarchical GP model with MCMC sampling (a) and ML-II approach (b); red dots show the predicted values, while blue lines are 95% confidence interval

ML-II in general is the fastest way to calculated GP's hyperparameters. It has a variety of implementations across different frameworks, but the principles are the same and based on maximum log-likelihood optimization. In the case of the framework which we used, it allowed us to set starting points for searching the parameters for kernel functions (as well as kernel functions) with the output of predicted (mean) value and its standard deviation. Additionaly, we can only add parameters to the diagonal of the covariance matrix. This makes the approach comprehensive and uncomplicated, and the use of the ML-II method for parameter autotuning is fast. The cost incurred in such an approach is the inaccuracy of the solution due to its tendency to overfit [23], having very little output information about the problem, and the frequent problems with catching the uncertainty [26].

MCMC requires much more in terms of preparation of the model (it has to be done by hand and with some expert knowledge) and the computation cost (sampling just a few chains can require a lot of time in certain problems). On the other hand, as the output we got not only the prediction but the whole distribution of it, as well as the distribution of parameters, from which we can calculate values that we need to improve research [24]. In our case we did not use the full potential of MCMC with a Stan model. There is still room to increase the accuracy and flexibility of our model (e.g. better prior selection), but we wanted to signal the existence of other approaches worth considering for their own properties.

## 5. Conclusion

Diagnosing the SoH with the use of the GP is something that is constantly being developed. This paper mainly focused on creating the multi-kernel approach with ML-II hyperparameter tuning as well as checking possible alternatives based on creating a hierarchical model with the MCMC sampling. The overall results of the ML-II approach are quite promising in terms of using the GP to battery SoH diagnostics. We compared a few known kernels when dealing with the

mentioned problem, where the Matern family with RQ showed the best results in terms of the RMSE. Then we tried to combine covariance functions with the best accuracy from the previous step in order to check if the multi-kernel approach can improve accuracy. The combination of Ma 3/2, Ma 5/2 and RQ resulted in the best RMSE score of 0.054 on the test set.

The used framework shows an uncomplicated way of GP modelling, with the possibility of coding previous knowledge about the problem with a choice of the kernel function and its start point of hyperparameter autotuning. Optimizing hyperparametres do the rest of the work within a reasonable time and the overall accuracy of prediction is satisfactory. Also, some limitations of this method are noticeable, e.g. sometimes it has some trouble with catching the uncertainty, results depend on the quality of optimizing and tends to overfit. A current state of implementation consists of models fitted into scenarios of battery usage, but in future development we probably could generalize the model even for situations not included in the NASA dataset.

Additionally, we tried an alternative approach creating a hierarchical model with the MCMC sampling method. Based on the theory from Rasmussen and Williams [12] and exemplary modeled problems, we implemented an STAN model for the GP, specifying priors and the kernel function. We used the two different kernels (RBF and Ma 3/2), as well as two different prior sets (basic one from STAN recommendation and the second one with our twists). The result showed that changes in prior assumptions can lead to significant differences in the posterior distributions. The best RMSE value was not a breakthrough in our research. The reason is that the best accuracy was not our first concern, but rather presenting a new possibility of GP modelling and its properties. The results of this approach cannot be directly compared to the presented ML-II method, because of not using its full potential. Furthermore, we should focus on the various pros and cons of both methods. The model with the MCMC sampling gives us the posterior distribution of each of the levels of a hierarchical model. This delivers more information about problems, which can lead to an additional test and meaningful conclusions. The main drawback is the need to accurately declare the prior distributions (based on the expert knowledge at best) and computational complexity.

In conclusion, both of the presented approaches show promise for the future development of the GP modeling of battery SoH diagnostics. The ML-II approach showed that autotuned parameters can lead to accurate models with valid kernel selection, while the method based on the MCMC sampling returns more information about problems but requires better knowledge of the problems and can outperform other approaches if set correctly. The paper showed a very early stage of development, especially in terms of the approach with a hierarchical model. It needs to be refined by adding expert knowledge in process creation. After such transformations, further comparative and testing stages can be specified, e.g. through measures other than the RMSE, like the mean absolute percentage error (MAPE). The current state shows the possibilities of an MCMC model and its meaningfulness for future research in this field, which will probably lead to a more general model. The improved MCMC model may not only be generalized but also become a competitor for other dominant solutions, which are used in the fields of capacity and SoH diagnostics, so our project will surely be developed.

**Acknowledgements**

This research was funded by AGH's Research University Excellence Initiative under the project "Interpretable methods of process diagnosis using statistics and machine learning" and by the Polish National Science Centre project "Process Fault Prediction and Detection" contract no. UMO-2021/41/B/ST7/03851.

**References**


[1] Tagade P., Hariharan K.S., Ramachandran S., Khandelwal A., Naha A., Kolake S.M., Han S.H., *Deep Gaussian process regression for lithium-ion battery health prognosis and degradation mode diagnosis*, Journal of Power Sources, vol. 445, 227281 (2020), DOI: 10.1016/j.jpowsour.2019.227281.

[2] Li X., Wang Z., Yan J., *Prognostic health condition for lithium battery using the partial incremental capacity and Gaussian process regression*, Journal of Power Sources, vol. 421, pp. 56–67 (2019), DOI: 10.1016/j.jpowsour.2019.03.008.

[3] Dillon S.J., Sun K., *Microstructural design considerations for Li-ion battery systems*, Current Opinion in Solid State and Materials Science, vol. 16, no. 4, pp. 153–162 (2012), DOI: 10.1016/j.cossms.2012.03.002.

[4] Li X., Yuan C., Li X., Wang Z., *State of health estimation for Li-Ion battery using incremental capacity analysis and Gaussian process regression*, Energy, vol. 190, 116467 (2020), DOI: 10.1016/j.energy.2019.116467.

[5] Garay F., Huaman W., Vargas-Machuca J., *State of health diagnostic and remain useful life prognostic for lithium-ion battery by combining multi-kernel in Gaussian process regression*, 2021 IEEE XXVIII International Conference on Electronics, Electrical Engineering and Computing, pp. 1–4 (2021), DOI: 10.1109/INTERCON52678.2021.9532733.

[6] Greenbank S., Howey D., *Automated Feature Extraction and Selection for Data-Driven Models of Rapid Battery Capacity Fade and End of Life*, in IEEE Transactions on Industrial Informatics, vol. 18, no. 5, pp. 2965–2973 (2022), DOI: 10.1109/TII.2021.3106593.

[7] Liu J., Saxena A., Goebel K., Saha B., Wang W., *An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries*, in Conference of the Prognostics and Health Management Society (2010), DOI: 10.36001/phmconf.2010.v2i1.1896.

[8] Fermín P., McTurk E., Allerhand M., Medina-Lopez E., Anjos M.F., Sylvester J., dos Reis G., *Identification and machine learning prediction of knee-point and knee-onset in capacity degradation curves of lithium-ion cells*, Energy and AI, vol. 1, in press (2020), DOI: 10.1016/j.egyai.2020.100006.

[9] Richardson R.R., Osborne M.A., Howey D.A., *Gaussian process regression for forecasting battery state of health*, Journal of Power Sources, vol. 357, pp. 209–219 (2017), DOI: 10.1016/j.jpowsour.2017.05.004.

[10] Tagade P., Hariharan K.S., Ramachandran S., Khandelwal A., Naha A., Mayya S., Kolake S., Han H., *Deep Gausian process regression for lithium-ion battery health prognosis and degradation mode diagnosis*, Journal of Power Sources, vol. 445, pp. 227–281 (2020), DOI: 10.1016/j.jpowsour.2019.227281.

[11] Zheng X., Deng X., *State-of-health prediction for lithium-ion bat- teries with multiple Gaussian process regression model*, IEEE Access, vol. 7, pp. 150383–150394 (2019), DOI: 10.1109/ACCESS.2019.2947294.

[12] Rasmussen C., Williams C.K.I., *Gaussian Processes in machine learning*, MIT Press (2006), DOI: 10.7551/mitpress/3206.003.0001.

[13] Davis R.A., *Encyclopedia of Environmetrics, Gaussian Process*, In Encyclopedia of Environmetrics; American Cancer Society (2006), DOI: 10.1002/9780470057339.vag002.

[14] Garnett R., *Bayesian Optimization*, Cambridge University Press (2022).

[15] Blum M., Riedmiller M., *Optimization of gaussian process hyperparameters using Rprop*, pp. 339–344 (2013).

[16] Raes W., Dhaene T., Stevens N., *On the Usage of Gaussian Processes for Visible Light Positioning with Real Radiation Patterns*, In Proceedings of the 2021 17th International Symposium on Wireless Communication Systems (ISWCS), pp. 1–6 (2021), DOI: 10.1109/ISWCS49558.2021.9562197.

[17] Chen T., Morris J., Martin E., *Gaussian process regression for multivariate spectroscopic calibration*, Chemometrics and Intelligent Laboratory Systems, vol. 87, no. 1, pp. 59–71 (2007), DOI: 10.1016/j.chemolab.2006.09.004.

[18] He Y.J., Shen J.N., Shen J.F., Ma Z.F., *State of health estimation of lithium-ion batteries: A multiscale Gaussian process regression modeling approach*, AIChE Journal, vol. 61, no. 5, pp. 1589–1600 (2015), DOI: 10.1002/aic.14760.

[19] Brevault L., Balesdent M., Hebbal A., *Overview of Gaussian process based multi-fidelity techniques with variable relationship between fidelities* (2020), DOI: 10.48550/arXiv.2006.16728.

[20] Richardson R.R., Osborne M.A., Howey D.A., *Battery health prediction under generalized conditions using a Gaussian process transition model*, Journal of Energy Storage, vol. 23, pp. 320–32 (2019), DOI: 10.1016/j.est.2019.03.022.

[21] Greenbank S., Howey D., *Automated feature selection for data-driven models of rapid battery capacity fade and end of life*, Transactions on Industrial Informatics, Institute of Electrical and Electronics Engineers (IEEE), vol. 18, no. 5, pp. 2965–2973 (2022), DOI: 10.1109%2Ftii.2021.3106593.

[22] Bole B., Kulkarni C., Daigle M., *Randomized battery usage data set, NASA AMES Prognostics Data Repository*, NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA (2014).

[23] Dudek A., Baranowski J., *Gaussian Processes for Signal Processing and Representation in Control Engineering*, Appl. Sci., vol. 12, no. 10, 4946 (2022), DOI: 10.3390/app12104946.

[24] Titsias M., Lawrence N., Rattray M., *Markov chain Monte Carlo algorithms for Gaussian processes. Inference and Estimation in Probabilistic Time-Series Models*, Bayesian Time Series Models, pp. 295–316 (2008), DOI: 10.1017/CBO9780511984679.015.

[25] Vanhatalo J., *Sparse Log Gaussian Process in Spatial Epidemiology*, Gaussian Processes in Practice in Proceedings of Machine Learning Research, vol. 1, pp. 73–89 (2022).

[26] Dudek A., Baranowski J., Mularczyk R., *Transient Anomaly Detection Using Gaussian Process Depth Analysis*, In Proceedings of the 2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 221–226 (2021), DOI: 10.1109/MMAR49549.2021.9528470.

[27] Cai L., Lin J., Liao X., *An estimation model for state of health of lithium-ion batteries using energy-based features*, J. Energy Storage, vol. 46 (2021), DOI: 10.1016/j.est.2021.103846.

[28] Burzyński D., *Useful energy prediction model of a Lithium-ion cell operating on various duty cycles*, Eksploatacja i Niezawodność, vol. 24, no. 2, pp. 317–329 (2022), DOI: 10.17531/ein.2022.2.13.

[29] Liu K., Tang X., Teodorescu R., Gao F., Meng J., *Future Ageing Trajectory Prediction for Lithium-ion Battery Considering the Knee Point Effect*, IEEE Trans. Energy Convers., vol. 8969, no. c, pp. 1–10 (2021), DOI: 10.1109/TEC.2021.3130600.

[30] Liu K., Shang Y., Ouyang Q., Widanage W.D., *A Data-Driven Approach with Uncertainty Quantification for Predicting Future Capacities and Remaining Useful Life of Lithium-ion Battery*, IEEE Trans. Ind. Electron., vol. 68, no. 4, pp. 3170–3180 (2021), DOI: 10.1109/TIE.2020.2973876.

[31] Burzyński D., Kasprzyk L., *A novel method for the modeling of the state of health of lithium-ion cells using machine learning for practical applications*, Knowledge-Based Syst., vol. 219, 106900 (2021), DOI: 10.1016/j.knosys.2021.106900.

[32] Su C., Chen H., Wen Z., *Prediction of remaining useful life for lithium-ion battery with multiple health indicators*, Eksploatacja i Niezawodność, vol. 23, no. 1, pp. 176–183 (2021), DOI: 10.17531/ein.2021.1.18.

*electronics*

MDPI

*Article*

# Spatial Modeling of Air Pollution Using Data Fusion

**Adrian Dudek and Jerzy Baranowski ***

Department of Automatic Control & Robotics, AGH University of Science & Technology, 30-059 Kraków, Poland; addudek@agh.edu.pl
* Correspondence: jb@agh.edu.pl

**Abstract:** Air pollution is a widespread issue. One approach to predicting air pollution levels in specific locations is through the development of mathematical models. Spatial models are one such category, and they can be optimized using calculation methods like the INLA (integrated nested Laplace approximation) package. It streamlines the complex computational process by combining the Laplace approximation and numerical integration to approximate the model and provides a computationally efficient alternative to traditional MCMC (Markov chain Monte Carlo) methods for Bayesian inference in complex hierarchical models. Another crucial aspect is obtaining data for this type of problem. Relying only on official or professional monitoring stations can pose challenges, so it is advisable to employ data fusion techniques and integrate data from various sensors, including amateur ones. Moreover, when modeling spatial air pollution, careful consideration should be given to factors such as the range of impact and potential obstacles that may affect a pollutant's dispersion. This study showcases the utilization of INLA spatial modeling and data fusion to address multiple problems, such as pollution in industrial facilities and urban areas. The results show promise for resolving such problems with the proposed algorithms.

**Keywords:** INLA; data fusion; air pollution; spatial modeling



**Citation:** Dudek, A.; Baranowski, J. Spatial Modeling of Air Pollution Using Data Fusion. *Electronics* **2023**, *12*, 3353. https://doi.org/10.3390/electronics12153353

Academic Editors: Esteban Tlelo-Cuautle and Manohar Das

Received: 22 May 2023
Revised: 2 August 2023
Accepted: 3 August 2023
Published: 5 August 2023



## 1. Introduction

Air pollution is a significant environmental and public health issue, affecting the well-being of millions of people worldwide. The increased industrialization and urbanization of modern society have led to a significant rise in air pollution levels, posing a severe threat to human health and the natural environment [1]. Exposure to air pollution has been linked to a wide range of health effects, including respiratory diseases, cardiovascular diseases, and even cancer.

Given the significant impacts of air pollution on human health and the environment, there is a critical need for data-driven solutions to address this problem. In recent years, there has been a surge of interest in data science approaches to air pollution research, including the use of machine-learning algorithms to model pollution levels and identify sources of pollution with gathered data [2–4]. Choosing the right models and collecting sufficient data are key to solving the problem.

One way to model air pollution data is through statistical models [5]. This allows for solving the problem of the air impact in the space in which it is located and measured. The distance between observations can be a source of correlation for such a model. For instance, housing values in a city typically show a gradual shift between adjacent neighborhoods [6]. Similarly, pollution levels display a spatially smooth pattern, such that measurements taken in close proximity are expected to yield comparable results [7].

These examples all share a common feature: their data are spatially dependent, with observations in close proximity likely to display analogous values and a high degree of spatial autocorrelation. Spatial models account for this autocorrelation to distinguish the general trend, typically conditioned on covariates, from the spatially random variation [8]. Problems that use spatial modeling fall into a variety of categories. One example could

be the problem of land surface temperature (LST). Its trends are measured using space sensors, specifically thermal infrared and passive microwave observations from satellite missions [9]. Satellites are useful in modeling air quality. In [10], we can see satellite-based atmospheric composition data collected by governments for air quality regulation and site-specific legislation compliance. The use of the TROPOspheric Monitoring Instrument showcases the potential of current instruments for detecting and isolating $NO_2$ emissions from regulated point sources. Moreover, satellites are used as a part of the data source for one of the datasets used in our research.

Such spatial modeling can be carried out with use of INLA (integrated nested Laplace approximation) [11,12]. INLA has gained recognition as an efficient approach to Bayesian inference in recent years. It offers a quicker and more user-friendly alternative to other methods, like MCMC, facilitated by the R-INLA package. INLA is designed to handle models expressed as latent Gaussian Markov random fields (GMRFs), but this broad category encompasses numerous practical models. INLA can be easily used for this type of problem, as evidenced by examples in the literature [5,13] in which it has been successfully applied.

Another problem that we often encounter during spatial modeling is the insufficient amount of information obtained from professional observation points. There are only a dozen such stations in the city of our operations. In this case, data fusion comes in handy. The amalgamation of multiple data sources to generate information that is more dependable, precise, and practical than what could be derived from any single source is referred to as data fusion [14]. It allows other data sources, for instance, automotive sensors [15], which include dozens of different sensors [16,17], which are still developing, or multi-sensor satellite images, which can be used to estimate surface soil moisture [18].

Sensors frequently vary in terms of their scale, measurement method, and accuracy. The information gathered through these sensors can be used, for example, to fill in gaps within the data.

In general, data fusion is a field of research that aims to combine information from multiple sources or sensors to produce a more accurate and reliable estimate of a phenomenon or event. With the increasing availability of diverse data sources and the need for improved decision making, data fusion has become an important area of study in many scientific and engineering disciplines [14]. The main challenge in data fusion is to integrate information from heterogeneous sources, which may have different measurement scales, uncertainties, and biases. To address this challenge, several theoretical frameworks and algorithms have been developed that enable the combination of information from multiple sources while accounting for their differences [14].

One such framework is the Bayesian theory of data fusion, which provides a probabilistic approach to combining information from multiple sources. The Bayesian theory assumes that each source of data provides a prior probability distribution over the phenomenon of interest, and the goal is to compute the posterior distribution that combines all the available information. The Bayesian theory also provides a natural way to incorporate expert knowledge and prior beliefs into the fusion process [19].

Another important aspect of data fusion is the selection of appropriate algorithms and techniques for integrating the data. There are several approaches to data fusion, including rule-based methods [20], mathematical modeling, and machine-learning techniques such as neural networks [21] and decision trees [22]. The choice of the fusion method depends on the nature of the data, the characteristics of the sources, and the application requirements.

Data fusion has many applications in fields such as remote sensing, image processing, robotics, health monitoring, environmental monitoring, and weight in motion (WIM). For example, in environmental monitoring, data fusion can combine information from different sensors to produce high-resolution images of the Earth's surface and let us calculate surface soil moisture [18]. In health monitoring, data fusion can be used to integrate data from wearable sensors to monitor the health of patients in real time [23]. In WIM, we can

combine data from different kinds of sensors [24], for instance, piezoelectric sensors [25] with polymer axle load sensors.

Data fusion is a powerful tool for combining information from multiple sources to produce more accurate and reliable estimates. The development of new theoretical frameworks and algorithms for data fusion is expected to play an important role in enabling data-driven decision making in many scientific and engineering domains.

In this paper, we investigate the possibility and effectiveness of using INLA within R-INLA framework in the spatial modeling of air pollution:

- Indoors, with the use of barrier modeling to implement information about obstacles;
- Outdoors, with the use of real data from air quality sensors in the large city;
- With the use of data fusion to enrich input data with datasets from different kinds of sensors and sources.

The key elements of our work are presented in Table 1. It also includes a comparison to other studies in a related field.

**Table 1.** Main contributions of this scientific paper in comparison to other studies in a related field.

| Article | Research Scope | Methodology | Results |
|---|---|---|---|
| Ours | Use of r-INLA package and data fusion for generation of spatial model of air pollution in different scenarios | Instances of modeled indoor and outdoor situations. Indoor case takes advantage of barrier modeling (information about obstacles), and outdoor (city) uses data fusion with different datasets. | Authors successfully created the air pollution model in all cases. Barrier models increase the reliability of indoor model. Data fusion fixes the imperfections of official measurement systems. |
| [5,13] | Use of INLA for spatial prediction of air pollutant concentrations | Creation of multi-pollutant geostatistical models with use of air quality data from monitoring measurements (observed monitoring measurements and sometimes correlations between pollutants). | Prediction accuracy compares favorably to previous efforts to map air pollution. Model properly estimates daily ambient pollution. |
| [6] | Analysis of the Spatial Diversity of Housing Prices | Introduction of composite external effect at a given point in space as the location-value signature (LVS) of that parcel, which is estimated in the model. It plays a role in the determination of the market value of a house residing on that parcel. Model based on fundamental assumptions: interact location variables with attributes like lot size, age, and dwelling size. | Authors use multiple hedonic specifications, including an interactive variables approach. Interactive model significantly improves explanatory power and removes spatial dependence in error terms compared to the noninteractive model within their sample. |
| [21,24,26] | Use of data fusion in different scenarios: vehicle weight, machine learning, classification | In the context of these articles, a similar framework is employed that involves the fusion of data from different sensors or datasets across different scenarios. | By integrating data from diverse sensors or datasets, these frameworks enhance the overall usability and accuracy of models. It allows researchers and practitioners to leverage the strengths of individual data sources and overcome the limitations of each, leading to more robust and reliable results. |

The rest of this paper is organized as follows. First, in the Materials and Methods section, we present the basic theory of INLA, the main points of our algorithm, and the methodology and datasets that we used. Then, we present the results of our experiments, with three different cases of use. The paper ends with a discussion and conclusions.

## 2. Materials and Methods

The main tool used in our modeling considerations is the R-INLA package, which contains detailed implementations of the possibilities of solving problems using sparse matrices, which are described in INLA. INLA, also known as the integrated nested Laplace approximation, was created by Rue, Martino, and Chopin in 2009 to serve as an alternative to traditional Markov chain Monte Carlo (MCMC) methods for approximate Bayesian inference [27]. INLA is specifically designed for models that can be expressed as latent GMRFs due to their computational properties [28]. In the INLA framework, the observed variables $\mathbf{y} = (y_1, \ldots, y_n)$ are modeled using an exponential family distribution, where the mean $\mu_i$ (for observation $y_i$) is linked to the linear predictor $\eta_i$ through an appropriate link function. The linear predictor includes terms on covariates and different types of random effects, and the distribution of the observed variables $y$ depends on some hyperparameters $\theta$. The distribution of the latent effects $x$ is assumed to be a GMRF with a zero mean and precision matrix $\mathbf{Q}(\theta)$, which depends on the hyperparameters $\theta$. The likelihood of the observations, given the vector of latent effects and the hyperparameters, can be expressed as a product of likelihoods for individual observations:

$$\pi(\mathbf{y} \mid \mathbf{x}, \theta) = \prod_{i \in \mathcal{I}} \pi(y_i \mid \eta_i, \theta)$$

In this context, the latent linear predictor is denoted by $\eta_i$ and is one of the components of the vector $x$ that includes all latent effects. The set $\mathcal{I}$ includes the indices of all the observed values of $y$, although some of these values may not have been observed. The primary objective of the INLA approach is to estimate the posterior marginal distributions of the model effects and hyperparameters. This is accomplished by taking advantage of the computational benefits of the GMRF and the Laplace approximation for multidimensional integration [29]. The joint posterior distribution of the effects and hyperparameters can be expressed as:

$$\begin{aligned} \pi(\mathbf{x}, \theta \mid \mathbf{y}) &\propto \pi(\theta)\pi(\mathbf{x} \mid \theta) \textstyle\prod_{i \in \mathcal{I}} \pi(y_i \mid x_i, \theta) \\ &\propto \pi(\theta)|\mathbf{Q}(\theta)|^{1/2} \exp\left\{ -\tfrac{1}{2}\mathbf{x}^\top \mathbf{Q}(\theta)\mathbf{x} + \textstyle\sum_{i \in \mathcal{I}} \log(\pi(y_i \mid x_i, \theta)) \right\} \end{aligned}$$

To simplify the notation, the precision matrix of the latent effects is denoted by $\mathbf{Q}(\theta)$, and its determinant is represented by $|\mathbf{Q}(\theta)|$. Additionally, $x_i = \eta_i$ for values of $i$ that are in the set $\mathcal{I}$ [29]. The computation of the marginal distributions for the latent effects and hyperparameters can be performed by taking into account that:

$$\begin{aligned} \pi(x_i \mid \mathbf{y}) &= \int \pi(x_i \mid \theta, \mathbf{y})\pi(\theta \mid \mathbf{y}) d\theta \\ \pi(\theta_j \mid \mathbf{y}) &= \int \pi(\theta \mid \mathbf{y}) d\theta_{-j} \end{aligned}$$

It is important to observe that both equations involve integrating over the hyperparameter space and require a reliable approximation of the joint posterior distribution of the hyperparameters. To achieve this, we approximate $\pi(\theta \mid \mathbf{y})$, denoted as $\tilde{\pi}(\theta \mid \mathbf{y})$ [27], and use it to approximate the posterior marginal distribution of the latent parameter $x_i$ as follows:

$$\tilde{\pi}(x_i \mid \mathbf{y}) = \sum_k \tilde{\pi}(x_i \mid \theta_k, \mathbf{y}) \times \tilde{\pi}(\theta_k \mid \mathbf{y}) \times \Delta_k$$

The weights $\Delta_k$ correspond to a vector of hyperparameter values $\theta_k$ in a grid. The calculation of the approximation $\tilde{\pi}(\theta_k \mid \mathbf{y})$ can vary depending on the method used, and it can take on different forms [27].

The main purpose was to create spatial models of air pollution with particulate matter. We divided our research into three separate steps (models):

- A model of an industrial room with a strong source of pollutant emissions (simulated data);
- A barrier model (including obstacles and walls) of an industrial room with a strong source of pollutant emissions (simulated data);
- A city air pollution model (real data).

The first two models, which are based on simulated data, are intended to show the difference between the use and non-use of barrier models that allow the implementation of information about the boundaries in the considered space. In this case, it allows us to model a hall with three rooms, where, in two of them, there are machines that are sources of pollution. The exact plan of the layout of the simulated rooms is shown in Figure 1. In the two larger rooms, there are machines—sources of air pollution. The room at the bottom of the diagram is a vestibule connected by narrow passages with rooms where devices work. This arrangement is to guarantee a lower spread of pollution.

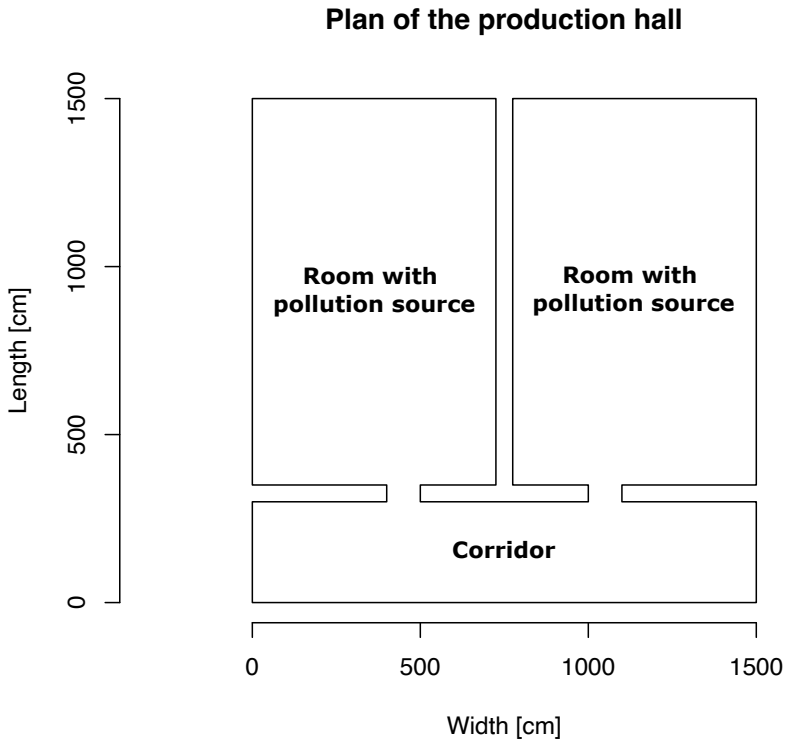



**Figure 1.** Plan of the production hall for the simulation process. The dimensions of the entire room are 15 m width and 15 m length. Inside the two larger rooms, there are machines that are sources of pollution. The room at the bottom of the graph is a vestibule between them, connected by narrow passages. This is to reduce the passage of pollutants between rooms.

We placed two sources of pollution in the defined rooms. Based on these assumptions, we created the distribution of pollutants using a bivariate normal distribution, the results of which can be seen in Figure 2. The obtained distribution simulates the random values of pollutants, where their peak is where the machines are located, i.e., points $(20, 100)$ and $(130, 100)$. Other settings (like the covariation function) have been randomized.

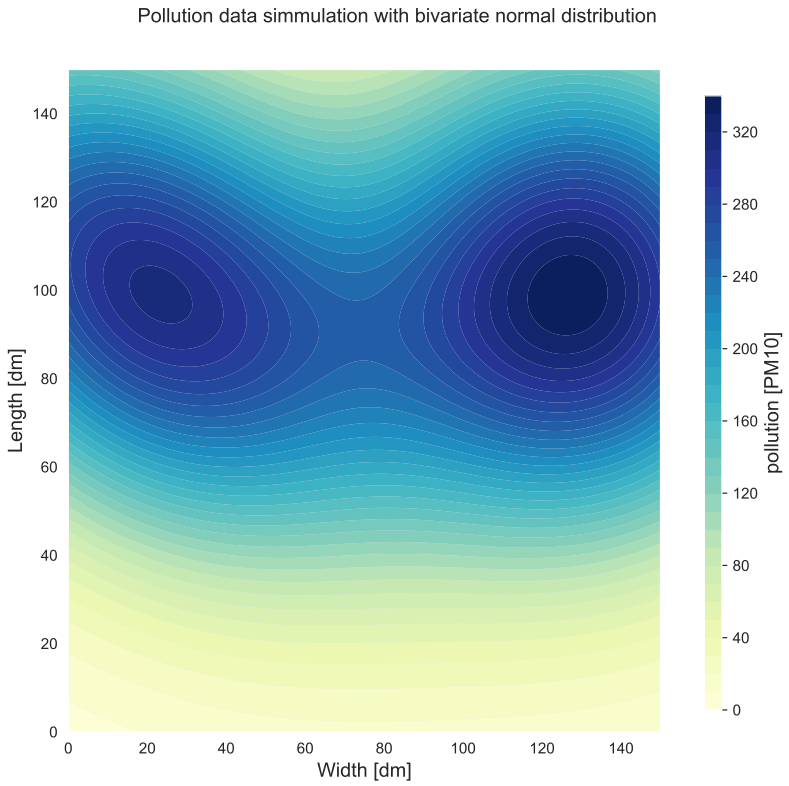



**Figure 2.** Randomly generated data on pollution in the defined production hall of 15 m width and 15 m length. Data were generated with bivariate normal distribution with randomized parameters, with the exception of maxima located at $(20, 100)$ and $(130, 100)$, where pollution sources are located.

The purpose of the third model is to use data fusion from different sources and sensors and create a spatial model of air pollution with INLA in the chosen city. In our case, it is Krakow in Poland. In order to fuse our data, we used two datasets, which provided us with differential sources of data. The first dataset comes from the database of a Polish government organization called the Main Inspectorate of Environmental Protection (MIEP) [30]. Their databases consist of a dozen public stations spread out in the city of Krakow and the surrounding area. Selected stations measured the level of PM10 in the air. These data are available in the MIEP dataset, and detailed information about stations near the city of Krakow can be found in Table 2. In the set of data shown in Table 2, we selected a subset of stations that actually measure PM10 values (bold names).

**Table 2.** All measuring stations near Krakow found in Main Inspectorate of Environmental Protection's dataset. The table shows the basic information to identify each station, such as the international station code and longitude and latitude. In addition, we have provided information on the values measured by the stations. We have highlighted in bold the stations that provide the data needed for the model (PM10).

| Station Name | International Code | Measured Values |
|---|---|---|
| **Krakow, Złoty Róg** | PL0643A | **PM10**, benzo(a)pyrene in PM10 |
| **Krakow, Aleja Krasińskiego** | PL0012A | **PM10**, PM2.5, carbon monoxide, nitric oxide, nitrogen dioxide, nitrogen oxides |
| **Krakow, Dietla** | PL0641A | **PM10**, nitric oxide nitrogen dioxide, nitrogen oxides |
| **Zabierzow, Wapienna** | PL0728A | **PM10**, benzo(a)pyrene in PM10 |

**Table 2.** *Cont.*

| Station Name | International Code | Measured Values |
|---|---|---|
| **Krakow, Bujaka** | PL0501A | **PM10**, PM2.5, benzo(a)anthracene in PM10, benzo(b)fluoranthene, benzo(k)fluoranthene, cadmium in PM10, dibenzo(a,h)anthracene in PM10, indene(1,2,3-cd)pyrene in PM10, nickel in PM10, nitric oxide, nitrogen dioxide, nitrogen oxides, ozone, benzo(j)fluoranthene in PM10 |
| **Krakow, Swoszowice** | PL0735A | **PM10**, benzo(a)pyrene in PM10 |
| **Skawina, Ogrody** | PL0273A | **PM10**, nitric oxide, nitrogen dioxide, nitrogen oxides, sulfur dioxide, benzene |
| **Kaszow** | PL0640A | Nitric oxide, nitrogen dioxide, nitrogen oxides, ozone |
| **Krakow, Piastow** | PL0642A | **PM10**, benzo(a)pyrene in PM10 |
| **Krakow, Bulwarowa** | PL0039A | **PM10**, PM2.5, arsenic in PM10, benzo(a)pyrene in PM10, benzene, cadmium in PM10, carbon monoxide, nickel in PM10, nitric oxide, nitrogen dioxide, nitrogen oxides, lead in PM10, sulphur dioxide |
| **Krakow, Wadow** | PL0670A | **PM10**, benzo(a)pyrene in PM10, arsenic in PM10, cadmium in PM10, nickel in PM10, lead in PM10 |
| **Niepolomice** | PL0125A | **PM10**, benzo(a)pyrene in PM10 |
| **Szarow** | PL0618A | Nitric oxide, nitrogen dioxide, nitrogen oxides, ozone |

The stations used by MIEP are public use stations and intended for the professional use of air quality measurements. The measurement methodology for particulate matter (PM10) is indicated in Directive 2008/50/EC of the European Parliament and of the Council of 21 May 2008 on ambient air quality and cleaner air for Europe (*Official Journal of the European Union*, L 152 of 11 June 2008, p. 1) and in the Regulation of the Minister of the Environment of 13 September 2012 on the assessment of the levels of substances in the air.

The Environmental Protection Inspection tests the content of PM10 in the air using two complementary methods:

- **The gravimetric (reference) method**, which is recognized and used around the world as the most precise measurement method;
- **An automatic method** with demonstrated equivalence to the reference method.

The gravimetric method, also called the manual (reference) method, uses the so-called dust collectors, special devices into which atmospheric air is sucked. Every two weeks, 14 disposable filters are placed in the collector, which the device changes automatically every 24 h. The advantage of this measurement method is its very high accuracy. Its only disadvantage is the time needed to obtain results, which is about 3 weeks. The automatic method has to be equivalent to the reference, so it must be demonstrated that the device meets the requirements for equivalence, and the results of such tests must be submitted to and approved by the European Commission. For measurements performed under MIEP, automatic meters with certificates confirming their equivalence to the reference method are used. These meters continuously measure dust concentrations and are used in the dataset used in our research [31].

In Figure 3, we can see an exemplary station (Dietla) from the set listed in Table 2, located close to the city center of Krakow. In contrast to the described set, the second one coming from the Airly source is a set of various types of more or less professional measuring stations. On the other hand, it has a much larger database that can be fused with the data provided by MIEP.

**Figure 3.** Example station used by the Main Inspectorate of Environmental Protection in their measurement system. The station name is Krakow, Dietla (code: PL0641A), and it is located near the city center and can measure the PM10 level, nitric oxide, nitrogen dioxide, and nitrogen oxides. Source: MIEP website [32].

The second set of air quality data, which was used for data fusion, come from the Airly dataset [33]. This dataset consists of a variety of sensors located throughout the city. The information that we collected was for a range of 15 km from the city center of Krakow. The sensors used in this set can be divided into three main types:

1. Air-quality-monitoring reference stations;
2. Low-cost sensors;
3. Satellites.

Reference stations are large enclosures that run on main power and house analyzers capable of generating reference-grade measurements. These structures are fixed at a specific point and provide air quality data. When data from several reference stations are combined, it becomes possible to model air quality for larger urban areas. Reference stations are often employed to ensure compliance with the legal limits of pollutants. There are various types of reference stations, the most common of which use filter-based gravimetric samplers to measure PM [34].

Low-cost sensors (LCSs) are used to collect real-time, high-resolution spatial and temporal air quality data. LCSs can measure PM, gas, or both. These sensors often consist of dense networks of multiple sensors, providing continuous monitoring that can identify areas of high air pollution within a town or city. Examples of LCSs are optical analyzers, which are popular PM sensors used in portable LCSs that use infrared or laser light to interact with particles and measure PM concentrations by number and size [34].

Satellite imagery can be utilized to analyze both PM and gaseous pollutants. By examining the amount of light that reaches the Earth's surface and the amount that is reflected off, satellites can measure the concentrations of different particles in the atmosphere. This method is called aerosol optical depth measuring and involves comparing these measurements with data on ozone concentrations and visibility [34].

In our research, we used an R-INLA implementation within R-studio. The spatial correlation between observations is computed using the SPDE technique, which discretizes the space by establishing a mesh that generates a fictitious set of neighbors across the research region. The inferences and predictions that we make will be significantly influenced by the mesh's design. In order to prevent outcomes from becoming sensitive to the mesh itself, it is crucial to build a decent mesh. Despite the fact that mesh construction varies

from case to case, there are certain recommendations for producing the best possible mesh. The size of the triangles in the inner zone must be appropriate, the outer bounds must be preserved, and the mesh must not create unexpected and unnatural patterns. The mesh, in certain cases, should also contain internal constraints, particularly when a barrier model is used. The barrier model is created by selecting appropriate polygons (triangles) and setting them as impassable barriers (e.g., walls). In the next step, the necessary elements for the model are defined, such as the covariance, the appropriate random effect, and the prediction function. At the end, according to the received model (supplied with data), we obtain a solution, which in, our case, due to transparency and data problems, we present as average prediction values in given areas. On this basis, we can draw conclusions about a given model for a given problem [29,35]. A diagram of the described algorithm is presented in Figure 4.

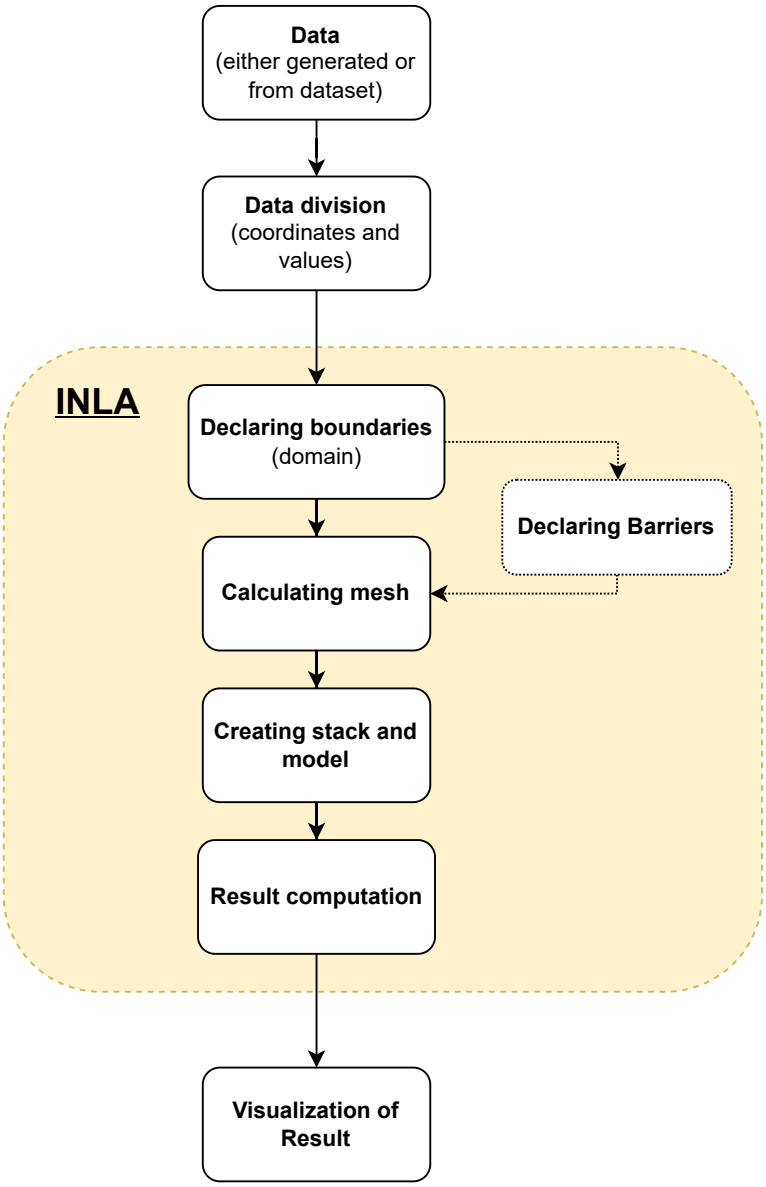



**Figure 4.** Diagrammatic representation of algorithm. First, we acquire data (which were generated or imported from a dataset) and then divide them into sets of coordinates and corresponding values. In INLA, we first declare the domain of our problem (boundaries), and if we are applying a barrier model, we also have to declare barriers. Next, we compute the mesh, which is used in the creation of the spatial model. Finally, the results are computed and visualized.

## 3. Results

The first two experiments were carried out in a three-room production hall defined for the model (Figure 1) on simulated data using the bivariate Gaussian distribution from Figure 2. In both cases, we determined the random positions of the PM10 measurement sensors in a defined space, where their values represent "measurements" from the data drawn using a bivariate Gaussian distribution. These sensor positions are shown in Figure 5. These 20 locations are inputs to the spatial model of air pollution in our defined factory. The locations of the sensors in the simulated space were based on a random selection of locations, with the condition that there must be sensors in each room.

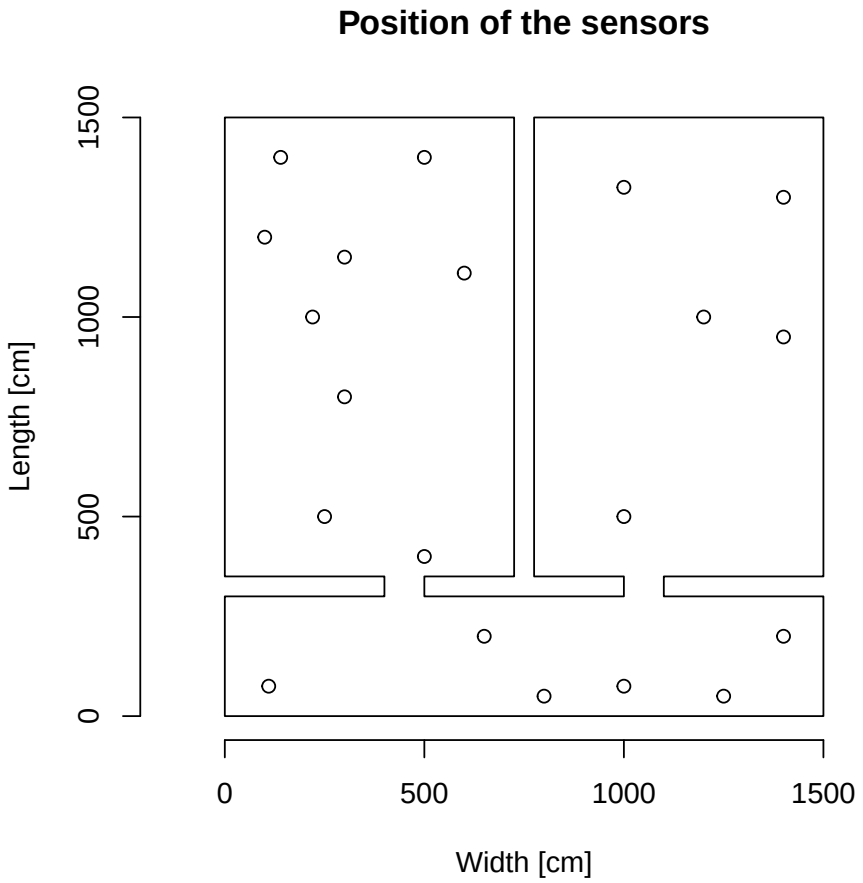



**Figure 5.** Selected positions of the sensors by randomized selection. Set of 20 points has values corresponding to the generated data using bivariate Gaussian distribution. Data from sensors are taken as inputs to both factory air pollution models.

The first step in creating a spatial model is to create the domain for which it will be calculated. In our case, it is, of course, a room with specific dimensions (15 m $\times$ 15 m). The mesh is the discretization of the domain (study area). The domain is divided up into small triangles. This is similar to creating a raster or a grid over space (reference). Such a mesh requires setting the appropriate values for the size of individual triangles constituting its interior (the place where all the data are located) as well as the appropriate buffer zone for the outer boundary. An equally important element is limiting the model only to the zone of considered data (in our case, the boundaries of the room). The mesh created in this way is shown in Figure 6, where the boundary separating the considered data area from the boundary is marked in blue, and the points with available data (simulated sensors) are in red. In the case of a general spatial model, we do not store information about internal wall barriers.

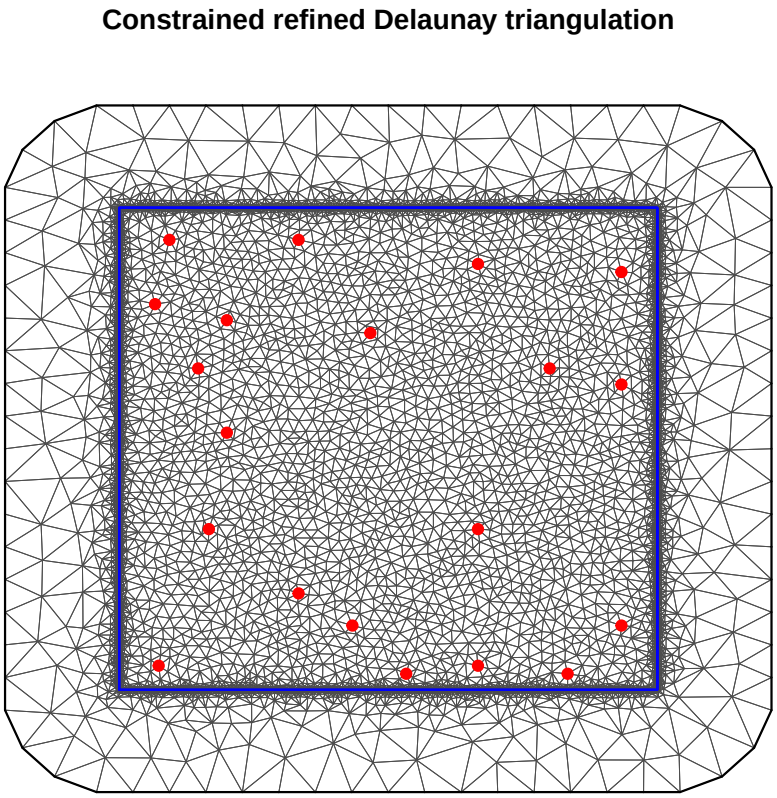



**Figure 6.** Mesh model for the first spatial modeling approach. The domain (inner section) is divided into small triangles, which allows them to connect to data from sensors. The outer boundary (buffer zone) is delimited by the size of the room (domain), marked with a blue line. In our case, we limit ourselves to the space of 15 m $\times$ 15 m only. Red dots represent positions of sensors.

The next stage of creating a spatial model in INLA is the creation of the so-called stack. Its first element is matrix A, mentioned earlier, whose main task is to connect information (data) at each point to its location in the previously created mesh. Moreover, it is a way of supplying covariates and random effects to INLA. For more complex spatial models, the stack is incredibly helpful, as the alternative is worse (we would have to construct the total model A matrix by hand). The last part is the definition of the spatial model component (random effect). In our case, the prior for the parameters/random effect is a Matérn prior with the hyperparameter range and marginal standard deviation. We start by defining the prior median for these hyperparameters. We also define the predictor for the model, which is a sum of model components and the observation likelihood.

The model created in this way allows the calculation of marginal distributions of hyperparameters, as well as marginal distributions of fixed effects. Plotting the posterior of the parameters in a random effect is much harder than plotting the hyperparameters because of the dimension of the parameters. So, as a result we do not plot the marginals, but instead we plot the summaries (like the mean). So, we present such a summary in the form of a spatial mean field (called a spatial estimate or spatial smoothing) as a solution to the problem. The graph in Figure 7 presents the average values obtained from the spatial model, with marked places representing simulated sensors. This spatial model is not a perfect solution to the problem of estimating the level of indoor air pollution, because it ignores obvious environmental obstacles, such as walls. This is important because isolated rooms at the bottom of the presented spatial plan of rooms should contain much lower average values in places separated by walls, and the only possible increases occur within the passages themselves.

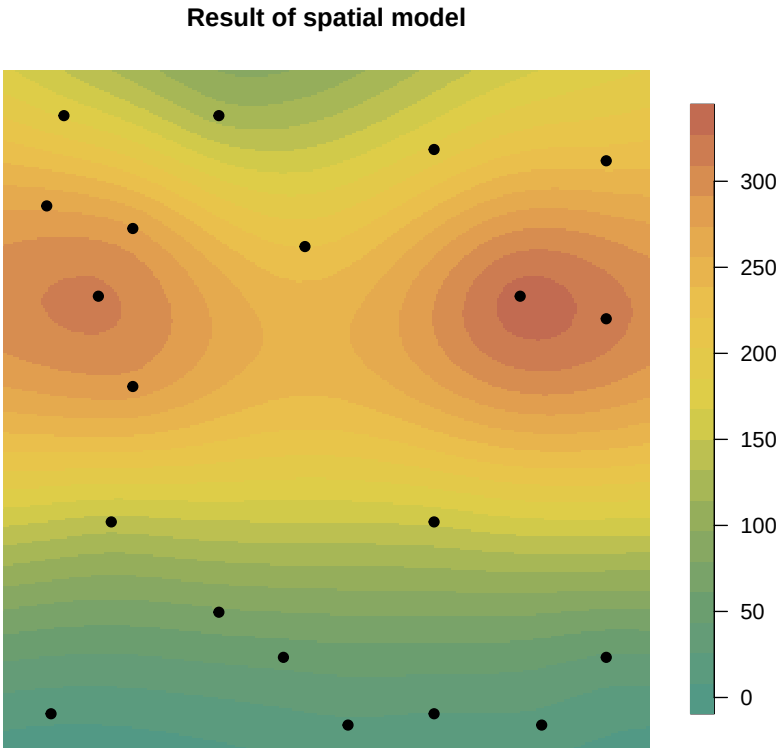



**Figure 7.** Result of spatial model without modeling space barriers. In the figure, the average pollution values for a given location in space are represented. Black dots represent positions of sensors. This model effectively reverses the randomly generated data from the bivariate normal distribution. It should be noted, however, that due to the lack of information about the walls of the building, it does not take into account, for example, the separation of the lower room, which makes it seem to have a much higher pollution factor than it would in practice.

In our second approach, we addressed the aforementioned imperfection of the spatial model for the production hall. The main difference is the implementation of an additional space barrier model, visible in Figure 8. It allows us to include information not only about the boundaries of the calculation domain but also about places that should be treated as impassable barriers. The created mesh model of space, of course, has the same limitations in regard to the domain of calculations indicated by the barrier model. The blue line in Figure 9 shows identical constraints to the barrier model. This approach allows us to include additional information for the model that should solve the problem of the previous approach.

The barrier model is subject to the same structure of settings as its previous version, i.e., the appropriate random effects, matrix A or predictor. Also, the result is presented in the same form as the average value of pollutants in space. However, in Figure 10, we see a difference in the form of the barriers involved. It is this factor that makes the results more reliable than the previous model. It is true that it does not perfectly represent the simulated data, but those data were only used to select a few values in space and not to calculate the quality of the model. In the case of the barrier approach, we clearly see that the model more reliably reflects the realistic spread of pollutants. The two rooms with sources of pollution have high values over the entire area. However, in the case of the isolated room, the inclusion of the walls allowed the model to change the prediction behavior and prevent air from spreading through the walls. This means that higher peaks of pollution values are only within the passages connecting individual rooms. Such results reflect reality better than the previous spatial model.

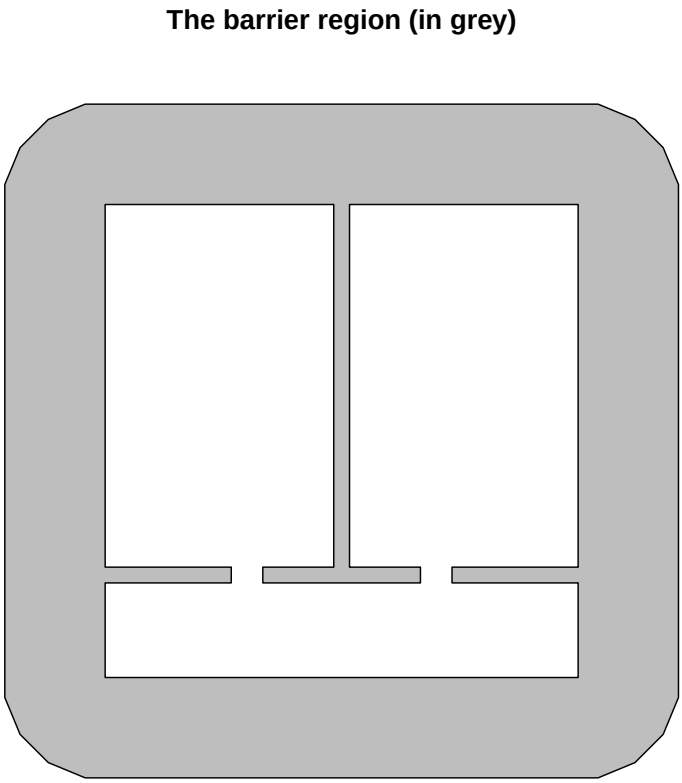



**Figure 8.** Barrier model for the considered production hall. It contains information about spaces inaccessible to the considered model (in our case, air pollution). It shows three rooms, two production rooms and one passage between them, as well as external and internal walls.

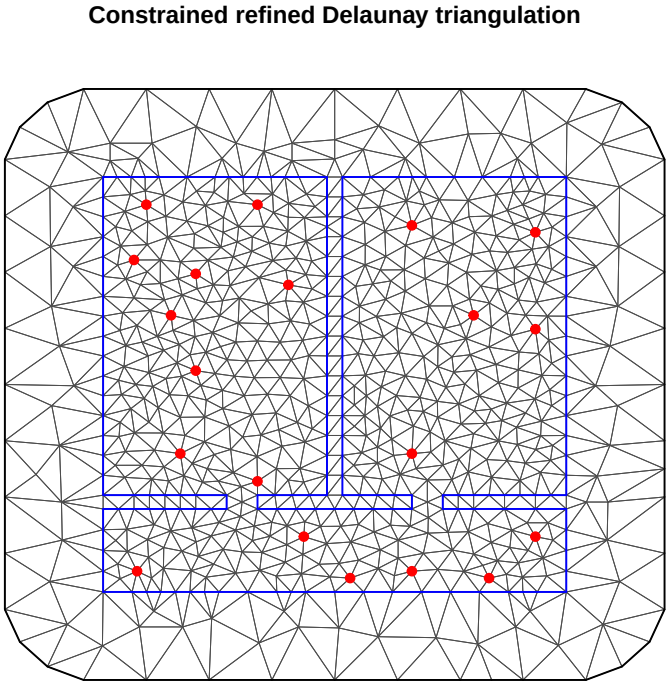



**Figure 9.** Mesh model for the second spatial modeling approach. The domain (inner section) is divided into small triangles, which allows them to connect to data from sensors. The inner zone (domain) is delimited by the location of barriers (walls) presented within barrier model from the outer boundary (buffer zone) and marked with a blue line. In our case, we limit ourselves to the space of 15 m $\times$ 15 m, only excluding walls. Red dots represent positions of sensors.

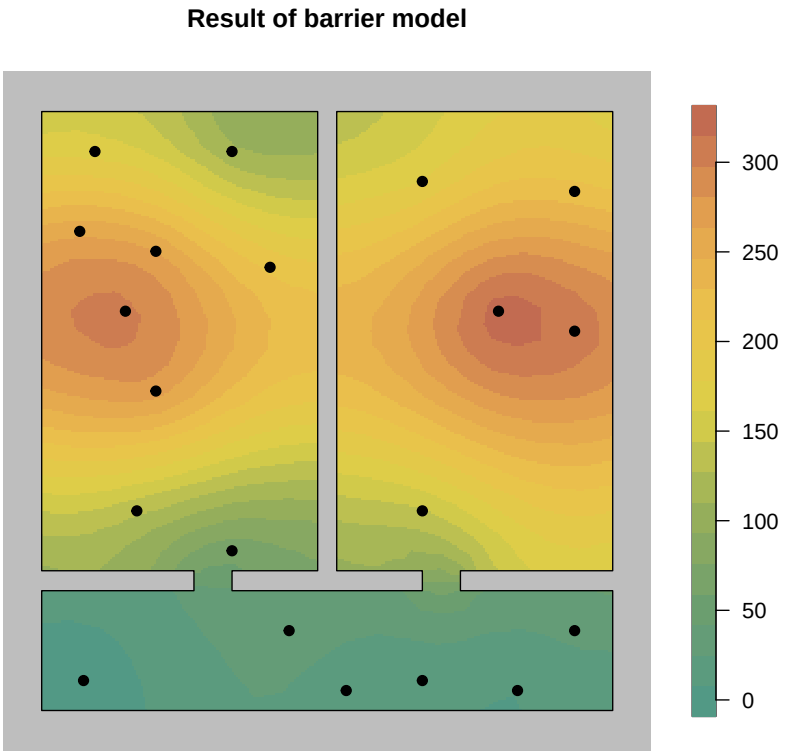



**Figure 10.** Result of spatial model with modeled space barriers. In the figure, the average pollution values for a given location in space are represented. Black dots represent positions of sensors. The result of this model is a good representation of the probable actual distribution of pollutants. Two production rooms, heavily polluted, do not differ significantly from those in the previous model. Separated rooms in the lower part have a better representation of average pollution values, taking into account higher values only in the area of passages and not in the whole space (where there are impassable walls).

The last model on which we conducted research was the spatial model of air pollution in the city of Krakow. As official professional sources have few research stations, we have merged the official data with the Airly database (which itself consists of many smaller databases and various sources and sensors). This allowed us to cover a sufficiently large area of Krakow (and its surroundings). In Figure 11, we have meshed our domain of consideration according to the previous procedure. Importantly, for better solutions at the edges of our final result, the domain area is larger than in the final considerations and has no clear separation from the buffer zone. As before, the red dots mark the locations of data from different sensors, from all databases together.

Based on information from the previously created spatial models representing air pollution, we set the priorities of our assumptions and predictors in a similar way. A noticeable difference is the lack of use of the barrier model (because it does not make sense for open airspaces) and not limiting the domain of considered data with the buffer zone. As mentioned earlier, the data are from midnight on 16 December 2022, and the average values calculated by the spatial model are included in Figure 12. In addition, characteristic places of the city (main streets, airport, etc.) have been marked in the background in order to improve orientation on the map. Judging from the results, the highest air pollution that day did not exceed 150 units, and its main accumulation was in the west and in the city center. Additionally, locations of stations that provided data are marked with dots, and the measuring stations of MIEP in particular are marked in blue.

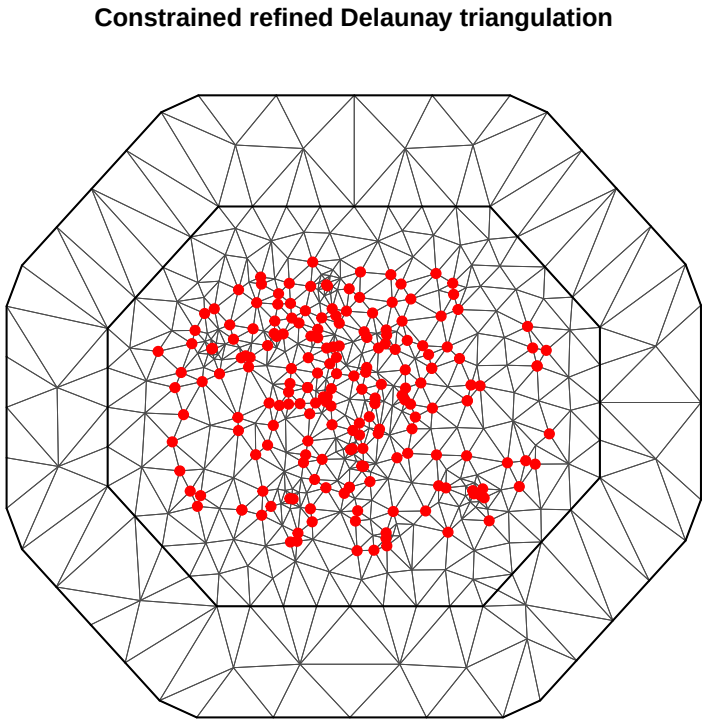



**Figure 11.** Mesh model for the spatial model of Krakow air pollution. The domain (inner section) is divided into small triangles, which allows them to connect to data from sensors. Outside, we can see the outer boundary (buffer zone). For this mesh, the domain of consideration is larger than the final result considered so as to be able to compute edge space information. Red dots represent positions of sensors from all datasets.

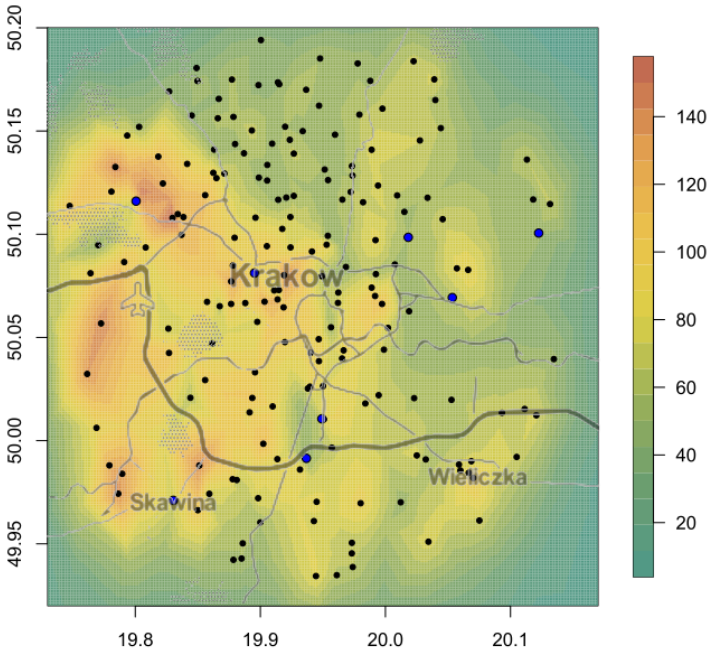



**Figure 12.** Result of spatial model of Krakow air pollution. In the figure, the average pollution values for a given location in space are represented. Black dots represent positions of sensors. Blue dots also represent positions of sensors, but only those from the MIEP dataset. In the background, there are elements that allow orientation in parts of the city (e.g., roads, airport). The maximum value of air pollution does not exceed 150, and its main concentration is in the western and central parts of the city. The $x$- and $y$-axes represent latitude and longitude values.

## 4. Discussion

The problem of air quality is a problem affecting many large and modern cities, as well as working spaces in factories. Its adverse effect on well-being and health is the key reason for solving it. However, before such measures are implemented, it is essential to understand the exact causes, sources, and behavior of polluted air in structures, whether in closed or open environments. Only then can appropriate countermeasures be taken. This is why it is so important to analyze the available data in advance and build appropriate mathematical models. The use of INLA in predicting the behavior of air pollution and detecting its sources is nothing new [5,13]. Some of the available studies even used a data fusion approach, as in our case. This is often necessary because even large cities, such as the examined Krakow, do not have a sufficiently dense infrastructure of official measuring stations. Another example of air pollution on a wide scale is presented in [36]. Research conducted during the COVID-19 pandemic examined the impact of carbon-based transport and industrial activity on air quality with the use of machine-learning techniques. The results provided by our algorithm, despite the structure of official stations not being sufficiently dense, may be relevant due to the use of data fusion from different collections and types of measurements. Using only the official measuring stations in the construction of the model could also be possible with the use of appropriate priors and INLA settings, but they would be burdened with high uncertainty. The use of data fusion allowed for the concentration of places where the information is reliable.

A different topic is pollution in workplaces, caused, for example, by the operation of machines. Detecting and preventing the causes is much simpler and less expensive compared to cities due to the scale of the problem. Separating the room with impurities is much easier. Detection, however, may prove to be a less trivial task due to, for example, a slight increase in contamination above the norm or contamination caused by a malfunction in the device. The problem of calculating air pollution exposure indoors can also be related to public spaces [37], taking into consideration human movement. It uses an activity-based travel demand model and low-cost air sensor network data, which leads to the use of different techniques than spatial modeling. A similar indoor problem can also be found in [38], where the research focuses on air quality in residential buildings. The topic is related to our problem, but it takes a different approach. It is based on a correlation with outdoor air pollution and takes into consideration the ventilation model, while ours provides results based on actual measurements, which can lead to better results, and the response within industrial structures. In the working environment of machines, our model can quickly recreate the current course of pollution in a two-dimensional space. It could be a very good addition to a system for warning about high pollution and detecting the place (source) of the failure, which may be difficult to locate using other sensors. Our model could discriminate between types of pollutants if other inputs were chosen. For this reason, it is worth analyzing information from sensors, especially with the use of barrier models, which can more accurately determine the cause and source of pollution, taking into account obstacles in the way.

## 5. Conclusions

The conclusion of this research in the field of spatial modeling is satisfactory. This is an early phase of research, the aim of which was to create a spatial model for indoor air quality and for open areas of the city. The first case was divided into two stages: a model without obstacles and a model with them (barrier model). Both models were based on the same source data, i.e., generated from a bivariate normal distribution. They were to simulate two rooms, each with a separate source of pollution (machine) and a connection (corridor).

In the first case, we can see that the model perfectly reflects the generated distribution, despite the small number of observation points (sensors). The maximum pollution values reach around 300, and the minimum values are in the vicinity, but greater than 0. Unfortunately, the aim of the model should be to create the best possible representation of indoor

air quality, taking into account, among other things, narrow passages, and not attempt to perfectly reproduce the pattern of generated data. For this reason, the second approach seems more appropriate. The barrier model allows us to take into account terrain obstacles, such as narrow spaces or walls. For this reason, it is much more suitable for modeling interior spaces. Thanks to these properties, it correctly detects, for example, a fenced-off lower corridor, in which the level of contamination is significantly reduced.

The case of the city of Krakow perfectly presents the operation of the model on a large scale of the city. On the day covered by the study, the level of pollution was concentrated around selected areas, which may be an appropriate place to consider a high source of pollution (e.g., an airport). It is equally important to use data fusion and to combine official information, including other measuring stations, not necessarily using identical measuring methods. This made it possible to precisely determine the level of pollution in space.

As a tool, INLA is perfect for solving these types of problems. The presented results clearly show the potential of this method, as well as areas for the further development of research. In the current, early stage, the algorithm does not have long-term history learning options, but adapting it to such a functionality may be a good step in improving our research. The same applies to extending the study of space from two to three dimensions. This would require the development of appropriate settings for the prediction model, prior, etc., but the INLA itself can accept data in three dimensions. So, the next step that we would like to take in the context of further research is to obtain more accurate models by, for example, taking into account heights for internal models and not using just two-dimensional coordinates. This will also allow us to consider information about spatial barriers that are not located at the entire height of room (e.g., furniture). In addition, with large-scale models such as cities, we could modify the model with additional information, such as a wind forecast, which will allow us to predict the level of pollution in the short and long term.

**Author Contributions:** Conceptualization, A.D. and J.B.; methodology, J.B.; resources, A.D. and J.B.; writing—original draft preparation, A.D.; writing—review and editing, A.D. and J.B.; visualization, A.D.; supervision, J.B.; project administration, J.B.; funding acquisition, J.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by AGH's Research University Excellence Initiative under project "Interpretable methods of process diagnosis using statistics and machine learning" and by the Polish National Science Centre project "Process Fault Prediction and Detection", contract no. UMO-2021/41/B/ST7/03851. Part of work was funded by AGH's Research University Excellence Initiative under the project "Interpretable methods of process diagnosis using statistics and machine learning".

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

| | |
|---|---|
| INLA | Integrated nested Laplace approximation |
| MCMC | Markov chain Monte Carlo |
| GMRF | Gaussian Markov random field |
| MIEP | Main Inspectorate of Environmental Protection |
| PM | Particulate matter |
| LCS | Low-cost sensor |

## References

1. Asghar, K.; Ali, A.; Tabassum, A.; Nadeem, S.; Hakim, S.; Amin, M.; Raza, G.; Bashir, S.; Afshan, N.; Usman, N.; et al. Assessment of particulate matter (PM) in ambient air of different settings and its associated health risk in Haripur city, Pakistan; [Avaliação de material particulado (MP) no ar ambiente de diferentes configurações e sua associação com o risco à saúde na cidade de Haripur, Paquistão]. *Braz. J. Biol.* **2024**, *84*, e256190. [CrossRef]
2. Fernández-Godino, M.G.; Lucas, D.; Kong, Q. Predicting wind-driven spatial deposition through simulated color images using deep autoencoders. *Sci. Rep.* **2023**, *13*, 1394. [CrossRef] [PubMed]

3. Duan, W.; Wang, X.; Cheng, S.; Wang, R. A new scheme of PM2.5 and $O_3$ control strategies with the integration of SOM, GA and WRF-CAMx. *J. Environ. Sci.* **2024**, *138*, 249–265. [CrossRef]
4. Davoodi, S.; Vo Thanh, H.; Wood, D.A.; Mehrad, M.; Rukavishnikov, V.S.; Dai, Z. Machine-learning predictions of solubility and residual trapping indexes of carbon dioxide from global geological storage sites. *Expert Syst. Appl.* **2023**, *222*, 119796. [CrossRef]
5. Saez, M.; Barceló, M.A. Spatial prediction of air pollution levels using a hierarchical Bayesian spatiotemporal model in Catalonia, Spain. *Environ. Model. Softw.* **2022**, *151*, 105369. [CrossRef]
6. Fik, T.; Ling, D.; Mulligan, G. Modeling Spatial Variation in Housing Prices: A Variable Interaction Approach. *Real Estate Econ.* **2003**, *31*, 623–646. [CrossRef]
7. Li, Y.; Zhu, Y.; Yin, W.; Liu, Y.; Shi, G.; Han, Z. Prediction of High Resolution Spatial-Temporal Air Pollutant Map from Big Data Sources. In *Big Data Computing and Communications: First International Conference, BigCom 2015, Taiyuan, China, 1–3 August 2015, Proceedings 1*; Springer International Publishing: Cham, Switzerland, 2015; pp. 273–282. [CrossRef]
8. Griffith, D. *Spatial Autocorrelation and Spatial Filtering: Gaining Understanding through Theory and Scientific Visualization*; Springer: Berlin/Heidelberg, Germany, 2003. [CrossRef]
9. Ahmed, M.R.; Ghaderpour, E.; Gupta, A.; Dewan, A.; Hassan, Q.K. Opportunities and Challenges of Spaceborne Sensors in Delineating Land Surface Temperature Trends: A Review. *IEEE Sens. J.* **2023**, *23*, 6460–6472. [CrossRef]
10. Potts, D.A.; Ferranti, E.J.S.; Timmis, R.; Brown, A.S.; Vande Hey, J.D. Satellite Data Applications for Site-Specific Air Quality Regulation in the UK: Pilot Study and Prospects. *Atmosphere* **2021**, *12*, 1659. [CrossRef]
11. Gómez Rubio, V.; Rue, H. Markov Chain Monte Carlo with the Integrated Nested Laplace Approximation. *Stat. Comput.* **2018**, *28*, 1033–1051. [CrossRef]
12. Lindgren, F.; Rue, H. Bayesian spatial modelling with R-INLA. *J. Stat. Softw.* **2015**, *63*, 1–25. [CrossRef]
13. Gong, W.; Reich, B.J.; Chang, H.H. Multivariate Spatial Prediction of Air Pollutant Concentrations with INLA. *Environ. Res. Commun.* **2021**, *3*, 101002. [CrossRef] [PubMed]
14. Hall, D.; Llinas, J. An introduction to multisensor data fusion. *Proc. IEEE* **1997**, *85*, 6–23. [CrossRef]
15. Senel, N.; Kefferpütz, K.; Doycheva, K.; Elger, G. Multi-Sensor Data Fusion for Real-Time Multi-Object Tracking. *Processes* **2023**, *11*, 501. [CrossRef]
16. Marszalek, Z.; Zeglen, T.; Sroka, R.; Gajda, J. Inductive Loop Axle Detector based on Resistance and Reactance Vehicle Magnetic Profiles. *Sensors* **2018**, *18*, 2376. [CrossRef]
17. Marszalek, Z.; Sroka, R.; Zeglen, T. Inductive Loop for Vehicle Axle Detection from First Concepts to the System Based on Changes in the Sensor Impedance Components. In Proceedings of the 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 24–27 August 2015. [CrossRef]
18. Singh, A.; Gaurav, K. Deep learning and data fusion to estimate surface soil moisture from multi-sensor satellite images. *Sci. Rep.* **2023**, *13*, 2251. [CrossRef]
19. Punska, O.; Doucet, A.; Wareham, R.; Walmsley, P.J. Bayesian Approaches to Multi-Sensor Data Fusion. Master's Thesis, Cambridge University, Cambridge, UK, 1999.
20. Safavi, S.; Mporas, I. Combination of Rule-Based and Data-Driven Fusion Methodologies for Different Speaker Verification Modes of Operation. In Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Exeter, UK, 21–23 June 2017; pp. 354–359.[CrossRef]
21. Lure, Y.M.F.; Grody, N.C.; Chiou, Y.S.P.; Yeh, H.Y.M. Data fusion with artificial neural networks for classification of earth surface from microwave satellite measurements. *Telemat. Inform.* **1993**, *10*, 199–208. [CrossRef]
22. Demirbas, K. Distributed sensor data fusion with binary decision trees. *IEEE Trans. Aerosp. Electron. Syst.* **1989**, *25*, 643–649. [CrossRef]
23. Singh, S.; Kumar, D. Energy-efficient secure data fusion scheme for IoT based healthcare system. *Future Gener. Comput. Syst.* **2023**, *143*, 15–29. [CrossRef]
24. Gajda, J.; Sroka, R.; Burnos, P. Sensor data fusion in multi-sensor weigh-in-motion systems. *Sensors* **2020**, *20*, 3357. [CrossRef]
25. Gajda, J.; Sroka, R.; Stencel, M.; Zeglen, T.; Piwowar, P.; Burnos, P. Analysis of the temperature influences on the metrological properties of polymer piezoelectric load sensors applied in Weigh-in-Motion systems. In Proceedings of the 2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings, Graz, Austria, 13–16 May 2012. [CrossRef]
26. Colosimo, B.M.; Pacella, M.; Senin, N. Multisensor data fusion via Gaussian process models for dimensional and geometric verification. *Precis. Eng.* **2015**, *40*, 199–213. [CrossRef]
27. Rue, H.; Martino, S.; Chopin, N. Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations. *J. R. Stat. Soc. Ser. B* **2009**, *71*, 319–392. [CrossRef]
28. Rue, H.; Held, L. Gaussian Markov Random Fields: Theory and Applications. In *Gaussian Markov Random Fields*; CRC Press: Boca Raton, FL, USA, 2005; Volume 104. [CrossRef]
29. Krainski, E.; Gómez Rubio, V.; Bakka, H.; Lenzi, A.; Castro-Camilo, D.; Simpson, D.; Lindgren, F.; Rue, H. *Advanced Spatial Modeling with Stochastic Partial Differential Equations Using R and INLA*; Chapman and Hall/CRC: New York, NY, USA, 2018. [CrossRef]
30. Portal Jakość Powietrza GIOŚ. Bank Danych Pomiarowych. Available online: https://powietrze.gios.gov.pl/pjp/archives (accessed on 20 April 2023).

31. Portal Jakość Powietrza GIOŚ. Pomiary Pyłu Zawieszonego w Powietrzu. Jak Się to Mierzy? Available online: https://powietrze.gios.gov.pl/pjp/content/show/1000919 (accessed on 20 April 2023).
32. Portal Jakość Powietrza GIOŚ. Stacja: Kraków Dietla. Available online: https://powietrze.gios.gov.pl/pjp/current/station_details/info/10121 (accessed on 20 April 2023).
33. Airly.org. Available online: https://airly.org/ (accessed on 20 April 2023).
34. Airly.org. Sources of Air Quality Data. Available online: https://airly.org/en/sources-of-air-quality-data-api-and-custom-reports/ (accessed on 20 April 2023).
35. Bakka, H. Online Course Topics for Bayesian Modeling. Available online: https://haakonbakkagit.github.io/index.html (accessed on 20 April 2023).
36. Wijnands, J.S.; Nice, K.A.; Seneviratne, S.; Thompson, J.; Stevenson, M. The impact of the COVID-19 pandemic on air pollution: A global assessment using machine learning techniques. *Atmos. Pollut. Res.* **2022**, *13*, 101438. [CrossRef] [PubMed]
37. Lu, Y. Beyond air pollution at home: Assessment of personal exposure to PM2.5 using activity-based travel demand model and low-cost air sensor network data. *Environ. Res.* **2021**, *201*, 111549. [CrossRef]
38. Gonzalo, F.d.A.; Griffin, M.; Laskosky, J.; Yost, P.; González-Lezcano, R.A. Assessment of Indoor Air Quality in Residential Buildings of New England through Actual Data. *Sustainability* **2022**, *14*, 739. [CrossRef]