SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Uncertainty Quantification for Gaussian Processes

Jana Huhne

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Uncertainty Quantification for Gaussian Processes

Unsicherheitsquantifizierung für Gauß-Prozesse

Author:	Jana Huhne
Supervisor:	Dr. Felix Dietrich
Advisor:	Vladyslav Fediukov, M.Sc.
Submission Date:	15.09.2023

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

J.Huhne

Munich, 15.09.2023 Jana Huhne

Acknowledgments

I would like to thank Vladyslav for his guidance, support and encouragement throughout the entire process of writing this thesis and for providing me the opportunity to work on such an engaging topic. I am also grateful to Felix for his valuable advice and feedback during my work. Moreover, I want to thank Jonathan Wenger for providing clarifications of his methodology and sharing the implementation he used in his paper, which greatly facilitated my comparative analysis. My appreciation extends to the Max-Weber Programm for the scholarship that supported my academic journey. Finally, I want to thank Moritz, Vianne and my family for their support and encouragement, which has been invaluable in completing this project. Special thanks to my dad for always inspiring me to tackle new challenges and being open to insightful discussions that have shaped my academic path.

Abstract

Machine learning is an essential tool for the predictive modeling of complex phenomena in a variety of fields, such as engineering, medicine, and finance. In addition to point predictions provided by the model, knowledge about the uncertainty of these predictions is crucial for informed decision-making. Due to their inherent ability to provide probability distributions alongside their predictions, Gaussian processes are especially well-suited for uncertainty quantification. This thesis explores various approaches to this task in the context of Gaussian process models. First, we apply sensitivity analysis to examine how the different input variables contribute to uncertainty in the prediction. Second, we quantify the additional uncertainty arising from numerical approximations used in place of the full model. Finally, we calibrate the predicted probability distribution to align more accurately with the empirical probability distribution. As an illustrative case study, these approaches are applied to a Gaussian process regression model built to predict the traction force on the wheels of a planetary exploration rover driving on soft soil. The findings presented in this thesis can benefit both researchers and practitioners aiming to enhance their knowledge of uncertainty in Gaussian process models.

Contents

1 Introduction

In recent years, machine learning has emerged as a promising complement to conventional data analysis and modeling techniques across various scientific disciplines. Machine learning models offer the benefit of directly capturing real-world phenomena from data without requiring an extensive modeling process. Additionally, they often require less computation time than complex, simulation-based approaches. However, their performance is intrinsically limited by the amount and quality of training data, as well as the ability of the model architecture to capture the problem at hand. Therefore, the predictions always entail uncertainty about the actual value.

Especially in high-stakes situations such as medical diagnosis, sociotechnical systems, or space exploration, it is essential for decision-makers to know how reliable their results are. For example, maneuvering a planetary rover in unknown terrain requires high caution since the rover cannot be repaired or moved out of a precarious situation with external help during the mission. The Mars rover Spirit, for instance, ended its operation by getting stuck in soft soil in May 2009 [1]. One of its successors, Curiosity, encountered multiple similarly critical situations in which its wheels were buried in sand up to 30% of their diameter. Due to an automatic onboard stopping mechanism and subsequent corrective maneuvers, immobilization was successfully averted on every occasion [2]. In such a scenario, relying on uncertain predictions could be detrimental to the mission's success. Uncertainty quantification therefore aims to assess the uncertainty associated with predictions and is as important as the predictions themselves for guiding decisions.

As a machine learning model,Gaussian processes are especially well-suited for handling uncertainty since they provide a normal probability distribution with each prediction. In this work, we explore various uncertainty quantification techniques for Gaussian processes that build on this probability distribution and provide additional insight. The first method is sensitivity analysis. The goal is to examine which input variables have the highest influence on the predictions and thus can have the most significant impact on its uncertainty if their values are uncertain as well. Additionally, identifying uninfluential variables enables to reduce the model's complexity by excluding them from the model. Moreover, Gaussian processes scale prohibitively for large amounts of data, necessitating numerical approximations. These introduce further uncertainty to the predictions, which is often overlooked but can be quantified by extending existing approximation methods. Lastly, we improve the uncertainty estimates of the model using calibration techniques. Since the predicted uncertainty is only an estimate, it may not accurately represent the actual uncertainty. For example, Gaussian processes tend to be overconfident.

We apply these uncertainty analysis methods to a practical case of modeling the locomotion of a wheeled planetary rover. Through this application, we evaluate their effectiveness in quantifying and understanding the inherent uncertainties of a Gaussian process model constructed to predict the traction force on the rover's wheels.

2 Background

2.1 Gaussian process regression

Gaussian processes are a non-parametric, probabilistic regression tool. In the following, we discuss what Gaussian processes are, how to use them for regression and why they are useful for uncertainty quantification.

2.1.1 Gaussian processes

Gaussian processes are highly related to multidimensional Gaussian distributions. In a multidimensional Gaussian distribution, each individual stochastic variable follows a normal distribution, and their joint distribution is normal as well. Formally, $X = \begin{bmatrix} X_1 & \cdots & X_D \end{bmatrix}^\top \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ is the mean vector with $\boldsymbol{\mu}_i = \mathbb{E}[X_i]$ and Σ the covariance matrix with entries $\Sigma_{ij} = \mathrm{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$. The variance of each dimension is thus captured in the diagonal entries Σ_{ii}, whereas the off-diagonal entries Σ_{ij} describe the correlation between dimensions i and j. Σ is always symmetric and positive semi-definite.

Two important properties of Gaussian distributions are closure under conditioning and marginalization, which means the resulting distributions from these two operations are Gaussian as well [3]. Conditioning can be used to determine the distribution of a subset of dimensions, given that the values of the other dimensions are already known (but arbitrary). For a Gaussian distribution

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_X \\ \boldsymbol{\mu}_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right) , \tag{2.1}$$

where X and Y represent subsets of the original random variable, the distribution for X given the values of Y is [3]

$$p_{X|Y}(\mathbf{x}) = \frac{p_{X,Y}(\mathbf{x}, \mathbf{y})}{p_Y(\mathbf{y})} = \mathcal{N}\left(\boldsymbol{\mu}_X + \Sigma_{XY}\Sigma_{YY}(Y - \boldsymbol{\mu}_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_Y Y^{-1} \Sigma Y X \right) . \tag{2.2}$$

Marginalization is another method for obtaining the distribution for a subset of the random variables. In this case, instead of fixing the other dimensions at specific values, their influence is eliminated by averaging over all possible values, weighted by their probabilities [3]

$$p_X(\mathbf{x}) = \int p_{X,Y}(\mathbf{x},\mathbf{y})d\mathbf{y} = \int p_{X|Y}(\mathbf{x})p(\mathbf{y})d\mathbf{y}\ . \tag{2.3}$$

Moving on to Gaussian processes, they can be interpreted as a generalization of Gaussian distributions. As stochastic processes, they represent probability distributions over functions. A multidimensional probability distribution assigns probabilities to vectors in a D-dimensional space. Each dimension represents a variable that can take on different values. A stochastic process, in contrast, is defined over functions. A function can be seen as a vector of uncountably infinite dimensions, each dimension representing a point in the input domain, such as the real numbers. The function is defined by the values assigned to each element in the input domain, similar to how a vector takes values in each dimension and a probability is assigned to each possible combination of values, in other words, to each function.

A Gaussian process

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x},\mathbf{x}')) \tag{2.4}$$

is fully specified by its mean function $m(\mathbf{x})$ and covariance function or kernel $k(\mathbf{x},\mathbf{x}')$. They are defined as

$$m(\mathbf{x}) \quad = \mathbb{E}[f(\mathbf{x})], \tag{2.5}$$

$$k(\mathbf{x},\mathbf{x}') = \mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{2.6}$$

and can be interpreted as a generalization of mean and covariance to functions. The defining property of a Gaussian process is that any finite collection of D of its random variables follows a D-dimensional multivariate Gaussian distribution [4]. This means the random variables representing function values at any finite set of points in the input domain are distributed according to a joint Gaussian distribution when marginalizing over all other input points. Formally, for the points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the resulting Gaussian distribution is given by

$$f(X) \sim \mathcal{N}(m(X), K(X,X))\ , \tag{2.7}$$

where $m(X)$ is the mean vector with entries $m(X)_i = m(\mathbf{x}_i)$ and $K(X,X)$ is the Gram Matrix of the kernel which is defined as $K(X,Y)_{ij} = K(\mathbf{x}_i, \mathbf{y}_j)$. Thus, even if not strictly correct in a mathematical sense, Gaussian processes can be thought of as Gaussian distributions over infinite-dimensional variables.

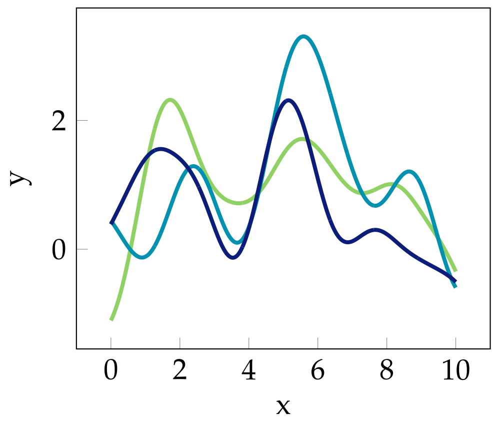
2.1.2 Application for regression

Regression is a form of supervised learning in which the training data comprises input-output pairs, with the objective of predicting the output based on new input values. Hence, the nature of this problem is inductive, as predictions for all possible input values are made based on a finite set of observations. The training data $\mathcal{D} := \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ is given by N observations, where $\mathbf{x}$ denotes the D-dimensional input and y the continuous, scalar output. The relationship between $\mathbf{x}$ and y is modeled by a function $y = f(\mathbf{x})$ in a deterministic setting or as a conditional probability distribution $p(y|\mathbf{x})$ if randomness is involved. The latter representation already contains information about the uncertainty of a prediction [5].

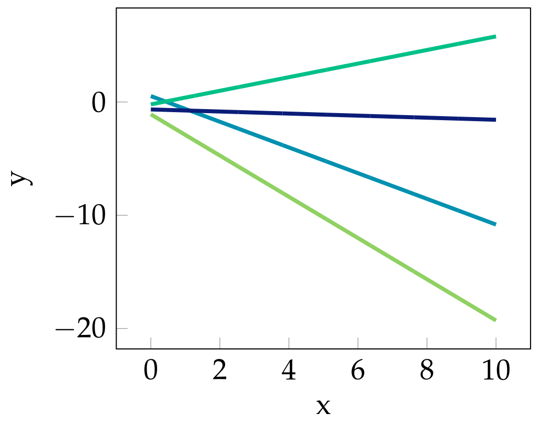
Generally, there exist infinitely many functions $f(\mathbf{x})$ consistent with the available training data. Therefore, searching the entire function space for the optimal fit is impractical, and the problem must be simplified to make it solvable. A common approach is parametric regression, where the set of possible functions is restricted to a specific parametric class, such as linear functions. The parameters are then optimized so that the resulting function best describes the training data. However, this method only works if the underlying input-output relationship is well modeled by the chosen class. For example, applying linear regression to highly nonlinear data will result in poor predictions. Choosing classes with higher approximation capability does not necessarily solve the problem since this increases the risk of overfitting, where the model represents the training data well but generalizes poorly to unseen data. Gaussian process regression uses a different approach, placing a probability distribution over all functions based on prior knowledge, such as smoothness and the training data. An advantage of this non-parametric approach is that matching the training data is always possible without the risk of overfitting [4].

Specification of the prior

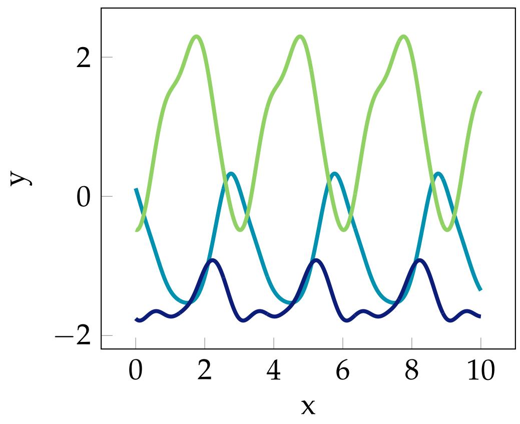
The first step in Gaussian process regression is choosing a Gaussian process that encodes prior knowledge independent of any data, assigning higher probabilities to functions considered more likely. This is similar to setting a prior for parameters in Bayesian regression. The kernel determines the characteristics of the predicted functions, such as smoothness or periodicity. As it provides a measure for similarity between the function values at two points in the input space, it influences the shape that functions from the distribution have. There are two general classes of kernels: stationary kernels, such as the squared exponential and periodic kernels, which only depend on the relative position of the input values and cannot model global trends, and non-stationary kernels, like the linear kernel, where the input location influences the result as well [3]. Figure 2.1 compares functions drawn from Gaussian processes with different kernels. In general, any function for which the Gram matrix K is positive semidefinite for all choices of input sets is a valid kernel [5]. In order to model more complex characteristics, different kernels can also be combined. Adding or multiplying



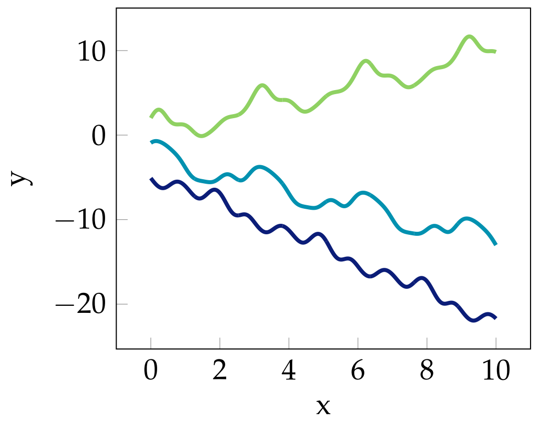
(a) RBF kernel



(b) Linear kernel



(c) Periodic kernel



(d) Periodic kernel + Linear kernel

Figure 2.1: Samples drawn from Gaussian processes with different kernels.

two kernels always yields another valid kernel, but there are also more complex options, like concatenation or composition with other functions. For example, adding a linear to a periodic kernel can incorporate a global rising or falling trend into a periodic model [3]. The last example in Figure 2.1 shows functions sampled from such a distribution. Kernels usually depend on hyperparameters that can be optimized to fit the data. For example, the length scale parameter of the squared exponential kernel determines the scale over which function values change, and the periodic kernel has a parameter for the period length of the oscillations [3]. For the optimization, a common approach is to use the logarithmic marginal likelihood

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}, X) p(\mathbf{f}|X) d\mathbf{f} \,, \tag{2.8}$$

which represents the probability of the training data given the hyperparameters $\boldsymbol{\theta}$ and can be computed in closed form [4]. $\mathbf{f}$ is the output of the Gaussian process model, which can be different from the observed data $\mathbf{y}$ if the data is assumed to be noisy. This will be discussed in more detail in the next part about Gaussian process inference.

In order to fully define the Gaussian process, a mean function has to be specified as well. It is typically chosen to be $m(\mathbf{x}) = 0$ for simplicity. Other common choices include the statistical mean of all values in the training data or another regression model such as linear regression [6].

Using this information, the Gaussian process representing all prior knowledge about the predictions without having seen any training data can be constructed. For a finite set of test locations $X_* = (\mathbf{x}_1^*, \dots, \mathbf{x}_m^*) \in \mathbb{R}^{m \times D}$, their distribution can be modeled by a multivariate Gaussian distribution obtained from the Gaussian process by marginalization

$$f(X_*) \sim \mathcal{N}(m(X_*), K(X_*, X_*)) \tag{2.9}$$

using the previously chosen mean function and kernel. Note that for a fixed Gaussian process, different queries for different sets of test locations will always be consistent [4].

Gaussian process inference

In the next step, knowledge about the training data $X = (\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathbb{R}^{n \times D}$ and corresponding outputs $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$ is incorporated by restricting the set of possible functions to those passing through the training points. Formally, the distribution of interest is $P(f(X_*)|f(X))$, where the values of $f(X) = \mathbf{y}$ are already known. In order to calculate this, we first need to construct the joint probability distribution

$$\begin{bmatrix} f(X) \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ m(X_*) \end{bmatrix}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix} \right) \,. \tag{2.10}$$

Inserting in Equation 2.2, the conditional probability distribution or posterior is given by [4]

$$\begin{aligned} f(X_*)|X, \mathbf{y} \sim \mathcal{N}(&m(X_*) + K(X_*,X)K(X,X)^{-1}(\mathbf{y} - m(X)), \\ &K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)) \,. \end{aligned} \tag{2.11}$$

In many real-world scenarios, there is uncertainty about the correctness of the training data. This could be caused by imprecise measurements, modeling errors, or the inherent probabilistic nature of the data-generating process. In these cases, it is helpful to allow predictions that deviate from the corresponding outputs in the training data. Assuming all uncertainty comes from independent, identically distributed Gaussian noise, the underlying input-output relationship can be modeled as $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Thus, the distribution of the training data is

$$f(X) \sim \mathcal{N}(m(X), K(X,X) + \sigma_n^2 I). \tag{2.12}$$

The posterior

$$\begin{aligned} f(X_*)|X, \mathbf{y} \sim \mathcal{N}(&m(X_*) + K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1}(\mathbf{y} - m(X)), \\ &K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1} K(X, X_*)) \end{aligned} \tag{2.13}$$

can be derived in the same way as above. Figure 2.2 illustrates the steps of Gaussian process regression.

The predictions of the Gaussian process are given by a joint Gaussian distribution. In many cases, point estimates for the predicted quantities are needed, and a probability distribution is not sufficient. In order to compare different predictions, we first need to define a loss function $\mathcal{L}(y_{\text{true}}, y_{\text{guess}})$, which represents a loss or penalty for guessing the value y_{guess} when the true value is y_{true}. The loss cannot be minimized directly since y_{true} is unknown, but the posterior describes a probability distribution for y_{true}. Therefore, a point estimate can be selected by minimizing the expected value of the loss

$$\tilde{R}_{\mathcal{L}}(y_{\text{guess}}|\mathbf{x}_*) = \int \mathcal{L}(y_*, y_{\text{guess}}) p(y_*|\mathbf{x}_*, X, \mathbf{y}) dy_* \,. \tag{2.14}$$

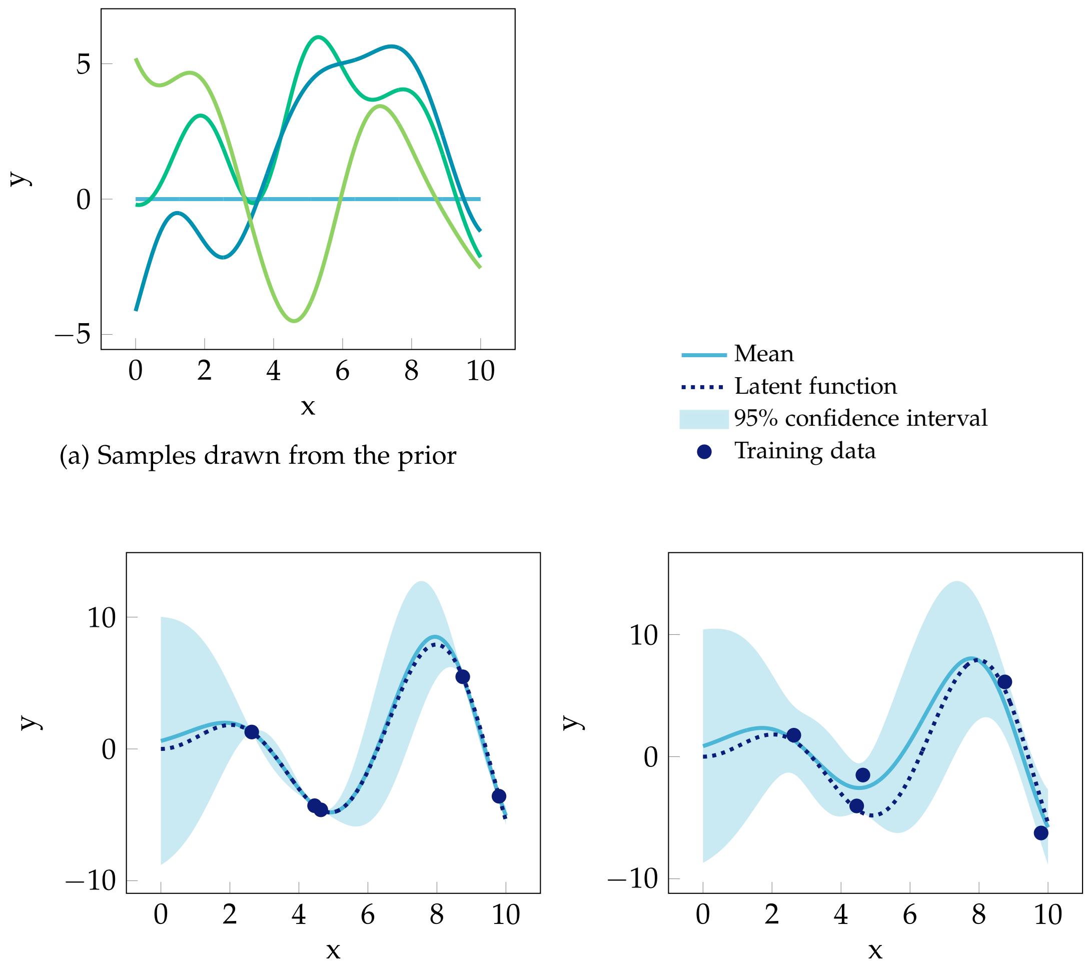
The point estimate is then given by

$$y_{\text{optimal}}|\mathbf{x}_* = \text{argmin } \tilde{R}_{\mathcal{L}}(y_{\text{guess}}|\mathbf{x}) \,. \tag{2.15}$$

For the absolute difference loss $|y_{\text{guess}} - y_*|$, the minimum is the median of $p(y_*|\mathbf{x}_*, X, \mathbf{y})$, for the squared loss $(y_{\text{guess}} - y_*)^2$ it is the mean. Those two coincide for Gaussian distributions; more generally, the mean is the optimum for any symmetric loss function. Therefore, the mean is usually used as point estimate for each prediction [4].

Predictive uncertainty

The predictive probability distribution of a Gaussian process serves as an estimation for the model's uncertainty about the predictions. In the case of no input noise, corresponding to Equation 2.11, this represents the epistemic uncertainty due to a lack of training data [4]. Therefore, the uncertainty is low in proximity to the training points, and it reaches zero exactly at those locations. In regions not well covered by training data, the uncertainty is higher. Figure 2.2b illustrates this behavior. Including a noise term as in Equation 2.13 allows to model aleatoric uncertainty about the training points **y**. Thus, the predictive uncertainty represents the combined epistemic and aleatoric uncertainty. As can be seen in Figure 2.2c, the uncertainty at the training locations is no longer zero in this case. Overall, Gaussian processes are particularly useful for quantifying uncertainty, as they provide specific uncertainty estimates for each prediction.



(a) Samples drawn from the prior

(b) Posterior

(c) Posterior with noise

Figure 2.2: The steps of Gaussian process regression. a) shows samples drawn from the prior distribution before conditioning on data. b) and c) both show the posterior mean and 95% confidence interval conditioned on the training data in dark blue for no uncertainty about the data points and an included noise term, respectively.

2.2 Modeling wheeled locomotion for a planetary rover

When exploring planetary bodies, wheeled rovers are a useful tool for examining not just the landing site but also the area around it. In such a scenario, the rover often needs to traverse areas with soft, sandy ground. This is especially challenging because of the risk of excessive slippage or sinkage of the wheels. If one or more wheels get embedded in the ground, the rover is stuck and cannot be recovered or repaired during the mission. Therefore, it is important to be able to analyze and simulate vehicle locomotion. In the design phase, engineers can use this information to test the wheel's behavior during many different possible scenarios. Moreover, the future behavior of the wheel can be predicted during operation and guide decisions in order to avoid critical scenarios. However, capturing the dynamics of wheel locomotion on soft soil poses difficulties, mainly due to the soil's lack of strong internal cohesion and the inconsistent non-proportional relation between the soil's deformation and the shear stress applied [7].

Therefore, simulation models that deliver results of high quality require a substantial amount of computation time and are not suitable for on-board predictions in real-time. A solution for this problem is building surrogate models based on machine learning approaches that can deliver results of similar quality in a shorter time [7, 8, 9, 10]. Uncertainty quantification for these models is an important part of assessing and preventing critical situations. Therefore, Gaussian processes with their built-in uncertainty estimates are well suited for this task [7].

2.3 Uncertainty in machine learning

In order to understand and effectively quantify the uncertainty inherent in machine learning model predictions, it is useful to differentiate between various sources and types. This section provides an overview of these different aspects of uncertainty.

2.3.1 Sources of uncertainty

There are many sources of uncertainty that can contribute to the overall predictive uncertainty of a machine learning model. In the following, we describe important sources discussed in recent academic literature.

- **Model uncertainty** describes the uncertainty about the choice of a model class. In machine learning, the set of possible models is usually restricted to a certain pre-selected class, such as linear models or Gaussian processes, before choosing one model from this class that fits the training data best. This class may not contain a model that is able to describe the true phenomenon well. Quantifying this kind of uncertainty is very difficult and often infeasible since all possible model classes would have to be considered and compared. Therefore, it is often left out of the uncertainty analysis. For models with high capacity, such as neural networks or Gaussian processes that can approximate many relationships in practical applications extremely well, it can safely be assumed that model uncertainty is low [11]. However, it is important to keep in mind that this type of uncertainty exists and is not included in most uncertainty estimates.

- **Approximation uncertainty** denotes the uncertainty associated with choosing the optimal model from the selected model class, usually by specifying the associated parameters and hyperparameters. Ideally, the model is chosen by minimizing a loss function over the whole space of input and output variables [11]. In practice, only a finite set of training data is available, and the model that minimizes the loss over this set does not necessarily coincide with the globally optimal model. Moreover, the optimization problem is often not convex, and thus, optimization algorithms may only find a local optimum even for the training data.

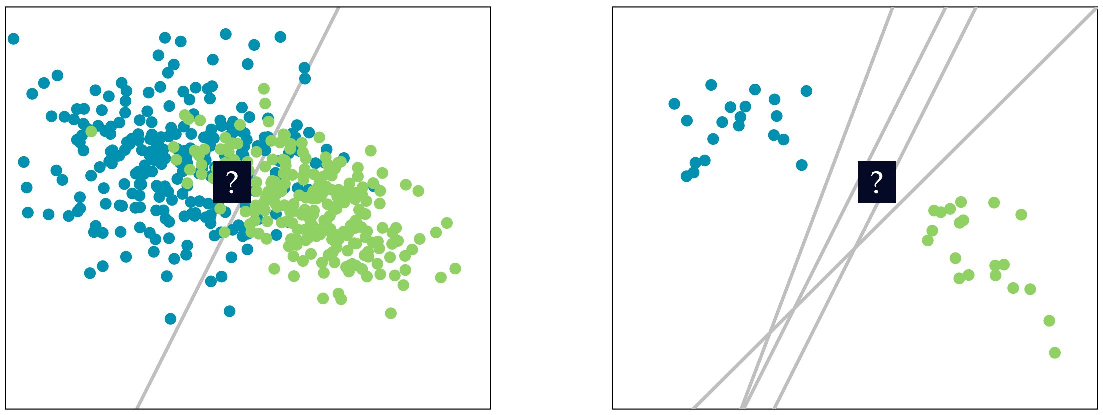
- The **amount of training data** determines how much knowledge the model has about the real-world phenomenon. In input regions not well covered by training data, the model relies on extrapolations, leading to higher uncertainty about the prediction. Generally, the higher dimensional the input space is, the more data is required to cover it well and reduce this type of uncertainty [11].

- **Noisy or imprecise data**, for example caused by measurements from inaccurate sensors, can affect the predictive uncertainty in two ways. Uncertain training data provide less accurate information for the model to learn from. Therefore, even in regions where training data are available, the model remains uncertain about predictions. Moreover, uncertainty in the test data is propagated through the model and causes uncertainty about predictions as well, especially if the output value is very sensitive with respect to the uncertain input variables [12].

- The **information content of the input variables** determines how much variability in the output quantity can be explained by the available data. If the input variables lack the predictive power to fully determine the output, the model remains uncertain about the prediction even with enough training data [12].

- **Numerical approximations** are often necessary during optimization or inference if the exact solutions are not analytically tractable or too computationally expensive for a specific application. This introduces uncertainty about the exact quantity of interest [11].

2.3.2 Aleatoric and epistemic uncertainty



(a) Aleatoric uncertainty (b) Epistemic uncertainty

Figure 2.3: Illustration of the difference between aleatoric and epistemic uncertainty in a classification task. The overlapping classes in a) make it impossible to determine the class of the test instance, marked by a question mark, with certainty even for the optimal model. Conversely, in b), it is not clear which model best represents the problem due to a lack of data, resulting in epistemic uncertainty about the class of the test instance. Adapted from [11].

For a comprehensive understanding of uncertainty, it is often useful to distinguish between two inherently different types of uncertainty. *Epistemic uncertainty* is the part of the total uncertainty that can be reduced given additional information. Conversely, *aleatoric uncertainty* is the remaining, irreducible part. What is reducible highly depends on the context since given enough effort, most sources of uncertainty can theoretically be resolved. Therefore, the definition must be interpreted relative to the problem at hand [13]. In practice, it is often necessary to view the setting consisting of the input space $\mathcal{X}$, output space $\mathcal{Y}$, the set of possible models $\mathcal{H}$ and a probability measure P on $\mathcal{X} \times \mathcal{Y}$ as fixed [11]. In such a setting, aleatoric uncertainty includes relationships between input and output variables that need to be viewed as stochastic, such as the outcome of a coin flip, noisy or imprecise data, and a lack of predictive power of the selected input variables. The remaining sources of uncertainty, i.e., uncertainty about the optimal parameters and hyperparameters, uncertainty from a lack of training data, and uncertainty from numerical approximations, are reducible by either additional data or more accurate computations, thus epistemic. Figure 2.3 illustrates the difference between aleatoric and epistemic uncertainty in a classification task. Figure 2.4 demon-

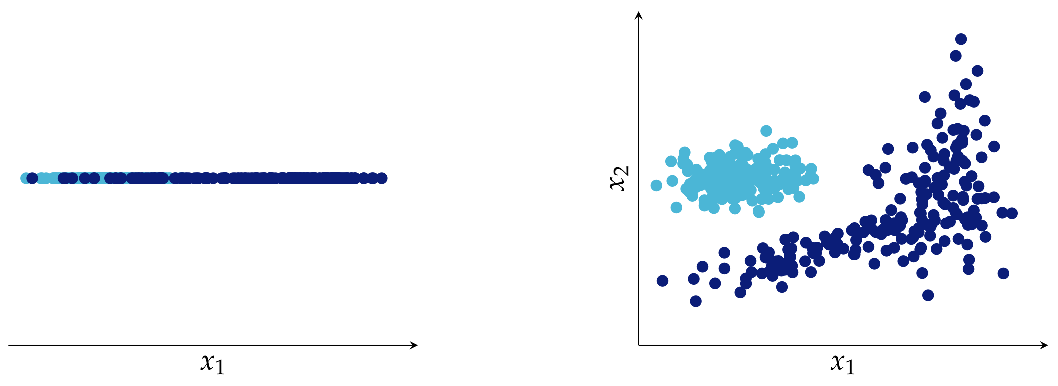


Figure 2.4: Depending on the problem setting, the classification of uncertainty as epistemic or aleatoric uncertainty changes. If the input variables are fixed to only x_1, predicting the class of a test instance entails aleatoric uncertainty due to the overlapping classes. If a second input x_2 can be added, the classes become separable, making the uncertainty reducible, thus epistemic. Adapted from [11].

strates how the classification of an uncertainty source as epistemic or aleatoric depends on the problem setting. The uncertainty in the region where the two classes overlap in one dimension x_1 is of aleatoric nature if the input space $\mathcal{X}$ cannot be modified. However, if the scenario allows for introducing the new input variable x_2, it is of epistemic nature since the classes become separable in two dimensions.

The distinction between aleatoric and epistemic uncertainty is especially useful in settings such as active learning, where the goal is to reduce the overall uncertainty. Since only epistemic uncertainty is of interest for this task, the performance can be improved by only considering this type of uncertainty [14]. However, separating the two is often difficult and may not be possible in cases where there are complicated interactions between sources of aleatoric and epistemic uncertainty [12].

2.3.3 Uncertainty quantification

In order to assess the predictive uncertainty of a model, it is desirable to have an uncertainty estimate in terms of a probability distribution or at least uncertainty bounds for each prediction. For this task, probabilistic machine learning models such as Bayesian linear regression, Bayesian deep learning, and Gaussian processes have been developed. Their predictions not only consist of point estimates but also include an estimate of the associated uncertainty. However, these uncertainty estimates usually only represent the combined aleatoric and epistemic uncertainty and do not give

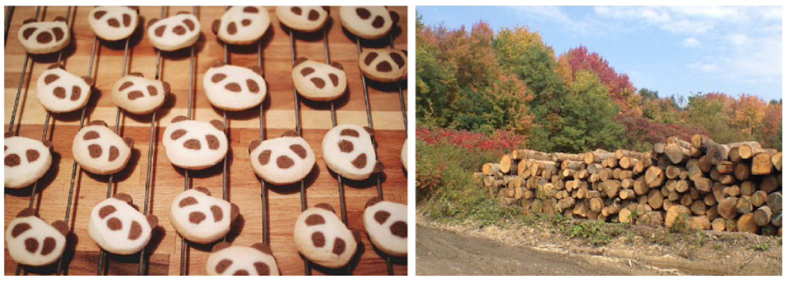


Figure 2.5: The neural network EfficientNet [15] confidently misclassifies two test images from ImageNet. The predictions are "typewriter keyboard" with certainty 83.14% for the left image and "stone wall" with certainty 87.63% for the image on the right. Source: [11].

information about its sources. Moreover, the predictions are only estimates and can be unreliable. For example, neural networks, though able to accomplish very high accuracy overall, tend to fail on specific instances while being confident in their prediction [11]. Two examples of this behavior are shown in Figure 2.5. Thus, additional methods for uncertainty quantification that go beyond the capabilities of the model are required in order to obtain more accurate representations of a model's predictive uncertainty and understand where this uncertainty is coming from [12].

2.4 Related work

In addition to the specific methods examined in depth in this thesis, the field of uncertainty quantification offers many more methodologies that are highly relevant to Gaussian process modeling. To give a broader picture of this field, we explore a selection of such methods that have found widespread use in recent scientific work in this section.

Heteroscedastic Gaussian processes

One basic assumption of Gaussian processes is that the data are generated by a deterministic function f with additive, statistically independent, and identically distributed Gaussian noise, that is,

$$y = f(\mathbf{x}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_n^2)\ . \tag{2.16}$$

This assumption about the noise is necessary to make inference analytically tractable. However, it is not always an accurate description of the data. Goldberg et al. [16] propose a heteroscedastic Gaussian process model which introduces an input-dependent noise variance modeled by another Gaussian process. Thus, the noise term becomes

$$\epsilon \sim \mathcal{N}\left(0, e^{g(\mathbf{x})}\right),\ g(\mathbf{x}) \sim \mathcal{GP}\left(\mu_0, k_g(\mathbf{x}, \mathbf{x}')\right)\ , \tag{2.17}$$

where μ_0 is a constant mean and $k_g(\mathbf{x}, \mathbf{x}')$ an arbitrary kernel function. The exponentiation is introduced in order to enforce the positivity constraint for the noise variance. Since no analytical solution for this model exists, Goldberg et al. [16] use a Markov chain Monte Carlo-based approach for inference. Improvements to this have later been developed using a most likely noise approach [17], expectation propagation [18] or variational approximation [19].

Non-stationary Gaussian processes

Similar to introducing input-dependent noise, Gaussian process models can also be adapted to use non-stationary kernel functions that vary in behavior across different points in the input space. This is desirable if the similarity of two function values depends on their location in the input space, as it is often the case for geostatistical data. While kernel families such as dot product kernels are inherently non-stationary [4], other kernels, such as the squared exponential kernel, can also be modified to exert this behavior. The dependence on the input values is introduced by modeling the kernel parameters as functions using a separate Gaussian process model, similar to modeling the noise in heteroscedastic Gaussian processes. Tolvanen et al. [20] model the variance parameter $\sigma(\mathbf{x})$ of a squared exponential kernel in this way using expectation propagation for inference. Heinonen et al. [21] additionally model the lengthscale $l(\mathbf{x})$ as a function of the input and use a Hamiltonian Monte Carlo method for inference. Patel et al. [22] follow a similar approach that includes both kernel variance and lengthscale and use the inducing point method to speed up inference for the latent Gaussian processes. Modeling the kernel variance $\sigma(\mathbf{x})$ as an input-dependent function allows for a more nuanced depiction of uncertainty, as it can represent the epistemic uncertainty at specific points in the input domain with greater flexibility than a single, fixed parameter [22].

Gaussian processes with input noise

Another assumption in Equation 2.16 is that the input data is noise free. However, this is not accurate for many use cases, for example, when the input values consist of sensor measurements. Gaussian processes thus cannot capture uncertainty about the input values propagated to the output, leading to overconfident predictions in scenarios where this plays a relevant role [23]. Several methods have been proposed that model the input noise as additive Gaussian noise, leading to the extended model [24]

$$y = f(\mathbf{x} + \boldsymbol{\epsilon}_x) + \epsilon,\ \boldsymbol{\epsilon}_x \sim \mathcal{N}(0, \Sigma_x),\ \epsilon \sim \mathcal{N}(0, \sigma_n^2)\ . \tag{2.18}$$

In this approach, Σ_x is diagonal, so the input noise is independent across dimensions.

McHutchon et al. [24] use a Taylor expansion of first order to propagate the input noise through f. This linear approximation offers the advantage of resulting in a Gaussian posterior for which inference can be carried out analytically. The noise variance for each input dimension can be inferred from the data by optimizing them as additional hyperparameters [24].

Other approaches consider the full propagation of input noise by marginalizing the output distribution over the uncertain inputs. Doing so generally leads to a non-Gaussian posterior [25]. Girard et al. [23] propose both a Monte Carlo-based method for obtaining the full posterior distribution as well as a method to analytically compute only its mean and variance. They only consider noise in the test data, which must be predetermined [23].

Dallaire et al. [26] avoid the problem of a non-Gaussian posterior by only incorporating the input noise into the kernel. Marginalizing the squared exponential kernel over the input probability distributions yields another valid, analytically tractable kernel that incorporates the input noise through hyperparameters. This approach can model uncertainty in both the test data and the training data and learn the noise parameters together with other kernel hyperparameters [26].

All these methods assume a constant noise across the input space. Wang et al. [25] introduce a method for modeling heteroscedastic input noise by assigning a separate variance to each point in the training set. Inference is again carried out using marginalization and a Monte Carlo estimate. Imposing uncertainty on the test data requires further estimations because the variance can differ throughout the input space [25].

Quantifying uncertainty about model selection

All methods for uncertainty quantification covered so far assume an optimal choice of the kernel and its hyperparameters. However, in practice, it is often not clear which kernel and what hyperparameters are best suited for generalization to unseen data.

Bajgiran et al. [27] include uncertainty about the kernel parameters in the predicted confidence intervals. They first restrict the set of possible parameters so that the probability of the true parameters falling in the set is at least $1 - \beta$ for some confidence level β with respect to the Gaussian process likelihood. The predictive confidence interval is then constructed from the pointwise worst-case values that can be produced by parameters within this set. Thus, this method balances accuracy, gained by making assumptions about the parameters, with robustness, achieved by considering worst-case scenarios [27].

Stephenson et al. [28] examine the sensitivity of Gaussian process predictions under changes of the kernel itself. Usually, there are uncountably many possible kernels that align with prior knowledge, such as smoothness or stationarity. A robust Gaussian process should not considerably change its predictions for different, qualitatively equivalent kernels consistent with the prior knowledge. This is especially important if critical decisions are based on the predictions, such as medical diagnosis. Stephenson et al. propose a method to evaluate this robustness by exploring a range of appropriate kernels around the original one and assessing the stability of the predictions [28].

Moreover, bounds on the squared predictive error can be derived under uncertainty about the kernel or its hyperparameters. Wågberg et al. [29] introduce such a lower bound for unknown kernel hyperparameters. An upper bound for the squared error is derived by Beckers et al. [30] considering uncertainty about both the kernel and its hyperparameters, given a set of possible kernels with corresponding hyperparameter sets.

Frequentist uncertainty bounds

The built-in uncertainty estimates of Gaussian processes are of Bayesian nature. This notion of uncertainty is not rigorous enough in applications such as learning based control with safety guarantees, for example autonomous driving, where theoretical guarantees in terms of frequentist bounds are required. These guarantees usually take the form

$$P\left(|f(\mathbf{x}) - \mu(\mathbf{x})| \leq \nu(\mathbf{x})\right) \geq 1 - \delta \tag{2.19}$$

for some $\delta \in (0, 1)$. Various methods for constructing such intervals have been developed [31, 32, 33, 34], depending on different assumptions about the true data generating function, the noise, and the kernel. There also exist error bounds that are robust to model misspecification. For example, the bounds introduced by Fiedler et al. [35] still hold if the chosen kernel does not perfectly describe the ground truth but is sufficiently close. Furthermore, Neiswanger et al. [36] construct frequentist confidence sequences without assuming correctness of the prior, including the choice of the kernel and hyperparameters.

3 Uncertainty quantification for Gaussian processes

3.1 Sensitivity analysis – methodical approach

Sensitivity analysis aims to identify how uncertainty in the individual model inputs contributes to the output uncertainty [37]. Even though it is not directly involved in the propagation and representation of uncertainty, sensitivity analysis should be a fundamental part of any such analysis, as emphasized in [38].

In practice, the values of input variables in the training data as well as in the data used for predictions are not known exactly. For instance, the accuracy of physical measurements is often limited by the precision of available sensors. This uncertainty about the input values is propagated to the model output, as different input values lead to different predictions. The more sensitive the output is to changes in a certain input variable, the stronger the uncertainty in this variable influences the output uncertainty. Therefore, sensitivity analysis is a valuable tool for identifying which input variables dominate the output uncertainty [38]. This is especially important for Gaussian process models since they assume their input values to be noise free. Thus, this type of uncertainty is not included in the predictive uncertainty unless it is explicitly accounted for through changes in the model, as for example in heteroscedastic Gaussian processes (see Section 2.4).

The results of sensitivity analysis not only provide insight into sources of the model uncertainty but can also be used to reduce it. Indicating which factors benefit most from further analysis or more accurate measurements, they can guide research prioritization [37]. Moreover, low sensitivity for an input variable indicates that it does not contribute much information and can potentially be excluded from the model. This reduction in the model's complexity leads to a lower cost of training and storage [39]. Moreover, fewer input dimensions generally reduce the epistemic uncertainty associated with the amount of training data since the input space is covered better by the same number of samples [11]. Simpler models can also be easier to optimize, leading to less uncertainty in the selection of optimal parameters or hyperparameters. For example, many common Gaussian process kernels have individual parameters for each input dimension. Thus, reducing the number of variables leads to fewer parameters that need to be optimized.

3.1.1 Methods for sensitivity analysis

There are two general approaches among sensitivity analysis methods.

Local sensitivity analysis examines how the output changes if the input variables are perturbed slightly with respect to a baseline value, usually in terms of partial derivatives. While this approach requires only few model evaluations and is therefore relatively cheap to compute, its expressiveness is limited. Derivative information in nonlinear models is only useful for uncertainty that entails sufficiently small deviations from the baseline value [37].

Therefore, the remaining part of this chapter focuses on the second approach, *global sensitivity analysis*, which considers variations of the input variables across their whole domain [40]. There are various methods for quantifying global sensitivity; this section briefly introduces those frequently discussed in the literature.

- **Regression analysis**: Regression analysis studies the coefficients and other statistics of a regression model fitted to an independent random sample of input-output pairs. For example, the statistical significance and magnitude of standardized regression coefficients, as well as the partial correlation coefficient (PCC), which measures the sensitivity of the output to an input variable when the effects of other variables have been canceled, can be used to characterize sensitivity [41, 42]. The method is only applicable if the conditions for linear regression are met, limiting its use to linear models. For nonlinear, monotonic models, rank regression can be used instead with the same sensitivity measures [42].
- **Elementary effects**: Similar to local sensitivity analysis, the elementary effects or Morris method uses partial derivatives as a sensitivity measure. Instead of only looking at derivatives at one baseline value, the method examines the distribution of derivatives over the whole domain. It can be efficiently calculated by placing a grid in the input domain and generating trajectories that change one variable at a time [43]. Multiple trajectories starting at different points allow for an even coverage of the input space. Being less computationally expensive than variance-based approaches, the elementary effects method is well-suited for problems with a large number of input variables. Consequently, it is often used in a preliminary screening step intended to narrow down the set of variables before carrying out a computationally more expensive analysis [44].
- **Variance-based analysis**: Variance-based methods express sensitivity as the reduction of output variance if one or more values of the input variables are known [45]. If the input variables are statistically independent, the output variance can be decomposed into contributions by single input variables and interaction effects between any combination of them [40]. Based on these variance

contributions, sensitivity indices known as Sobol′ indices offer insights into how the model is influenced by inputs and their interactions over the entire input domain. However, the computational cost can be substantial as the model has to be evaluated at a large number of points [37].

- **Metamodel based methods or response surfaces**: For models that are very expensive to evaluate, simpler metamodels can be built from a set of model evaluations and used in place of the original model in the subsequent analysis [42]. Popular choices of metamodels or emulators include linear regression models, polynomial chaos expansion, Gaussian processes, neural networks, and boosting regression trees [42, 46, 47]. Additional to being cheaper to evaluate, analytical solutions often exist for common sensitivity measures such as Sobol′ indices, eliminating the need to approximate them using sampling-based techniques. The quality of results highly depends on the ability of the metamodel to approximate the true model well [42].

3.1.2 Variance-based sensitivity analysis

In the following, the focus lies on variance-based sensitivity analysis since it provides an in-depth view of how the output variance can be attributed to different input variables and their interactions with only few assumptions about the model. For this approach, the model inputs $X_1, \dots, X_D$ are treated as random variables following some specified distribution and, as a consequence, the output $Y = f(X_1, \dots, X_D)$ becomes a random variable as well. In the case of Gaussian process regression, Y is the posterior mean. Therefore, only the variation of the posterior mean under variations of the input variables is analyzed, discarding the information provided by the predicted variance. However, variations of the method exist that use this additional information, as discussed later in this section.

Variance decomposition and Sobol′ indices

In order to separate the effects of single variables and their interactions, f can be represented as a sum of terms of increasing dimension

$$f(X_1, \dots, X_D) = f_0 + \sum_i f_i(X_i) + \sum_i \sum_{j>i} f_{ij}(X_i, X_j) + \dots + f_{1,2,\dots,D}(X_1, \dots, X_D) \,, \tag{3.1}$$

called high dimensional model representation (HDMR). Such a decomposition is always possible if f is square integrable over the D-dimensional unit hypercube [37]. In case the features have different ranges, they can be scaled to this domain prior to the analysis.

The individual terms in the decomposition are given by conditional expectations of Y

$$f_0 = \mathbb{E}(Y) \,, \tag{3.2}$$

$$f_i(x_i) = \mathbb{E}(Y|X_i = x_i) - f_0 \,, \tag{3.3}$$

$$f_{ij}(x_i, x_j) = \mathbb{E}(Y|X_i = x_i, X_j = x_j) - f_i - f_j - f_0 \,, \tag{3.4}$$

and so on [37]. Calculating the variance of each term yields a useful representation of the output variance in terms of contributions by individual variables and their interactions. For independent input variables, the partial variances add up to the total variance of the output

$$V_Y = \sum_i V_i + \sum_i \sum_{j>i} V_{ij} + \ldots + V_{1,2,\ldots,D} \,, \tag{3.5}$$

where

$$V_I = V(f_I) \tag{3.6}$$

for $I \subset \{1, \ldots, D\}$ [37]. Each term V_I represents the variance in the model output due to the interaction effect within the set of variables described by the indices I, or by the variable X_i alone for $I = \{i\}$ [37].

Variance-based sensitivity analysis is often used in scenarios where the input variables represent parameters with a fixed but unknown value. Hence, the variance of the output V_Y can be interpreted as a measure of the uncertainty about the correct output due to uncertainty about the true values of the parameters [37]. In the context of regression, where the input variables are the inputs of the modeled function and thus have no single correct value, the interpretation is different. Since there is no true value for Y but one prediction per combination of inputs, V_Y is a measure for the variability of Y over the whole input domain. The goal of sensitivity analysis is thus to find out which input variables or combinations lead to large variations in Y when their values change. If one of the components V_I has a high value, it means that the interaction effect of the corresponding variables explains a large amount of the total variability of Y. Thus, a change in these variables is expected to significantly influence the prediction.

Dividing the partial variances by V_Y yields a scale-invariant sensitivity measure known as Sobol' indices. The first order Sobol' index or main effect of input X_i

$$S_i = \frac{V_i}{V_Y} = \frac{V(\mathbb{E}(Y|X_i))}{V_Y} \tag{3.7}$$

describes the fraction of the overall output variance V_Y due to not knowing the value of X_i [37]. In other words, it is a measure of how much Y varies for different values of X_i. The effects of other variables are eliminated by calculating the expected value over all their possible values before evaluating the variance. Thus, this Sensitivity index can be used to identify highly influential inputs. However, a low main effect does not imply low relevance since there could be significant contributions through interactions with other variables [37].

Similarly, the second order Sobol' index

$$S_{ij} = \frac{V_{ij}}{V_Y} = \frac{V(\mathbb{E}(Y|X_i, X_j)) - V(\mathbb{E}(Y|X_i)) - V(\mathbb{E}(Y|X_j))}{V_Y} \tag{3.8}$$

represents the fraction of variance due to interaction effects between X_i and X_j that goes beyond the sum of contributions by X_i and X_j individually [37].

These sensitivity indices can in principle be calculated for all higher-order interactions as well. However, since there are $2^D - 1$ combinations in total, computing them becomes expensive, and the analysis is usually limited to first order and sometimes second order indices [37].

Another interesting sensitivity measure arising from the variance decomposition are total effect indices

$$S_{T_i} = 1 - \frac{V(\mathbb{E}(Y|\mathbf{X}_{-i}))}{V_Y} , \tag{3.9}$$

where $-i$ denotes the set of all indices except i. They describe the variance left unexplained in the output if all variables except X_i are known. If the input variables are independent, S_{T_i} is equivalent to the sum of all Sobol' indices that contain the index i, representing the fraction of output variance explained by the effect of X_i, including all its interactions with other variables. Therefore, a total effect index close to 0 is necessary and sufficient for X_i to be noninfluential [37].

Dependent variables

If the input variables are not independent, the partial variances in Equation 3.5 no longer add up to V_Y. The variance-based sensitivity can still be applied, but the results are more difficult to interpret.

Saltelli and Tarantola [48] suggest still using the first order Sobol' indices S_i as measures for the relative influence of the individual variables on the output uncertainty in this case. However, unlike for independent variables, these indices do not depend only on one variable X_i. The variables correlated with X_i can not only influence its probability distribution but also introduce interaction effects into the sensitivity index by interacting with X_i or other variables. For example, in a setting where $Y = x_1 + x_2 + x_3 + a_{23}x_2x_3$ and X_1 and X_2 are correlated, the sensitivity index S_1 depends not only on X_1 as would be the case for independent variables but also on the distributions of X_2 and X_3, as well as on the factor a_{23} [48]. Another disadvantage of this approach is that the computational shortcuts for calculating the indices are not applicable for dependent variables, resulting in a significantly higher computational cost for estimating the sensitivity indices [48].

Another approach is grouping correlated input variables so that there are no correlations between groups. The groups can thus be treated as independent, multidimensional variables in the subsequent analysis. The Sobol' indices for groups specified by subsets of indices $I, J \subset \{1, \ldots, D\}$ can be defined analogously to the versions for single variables [49]

$$S_I = \frac{V(\mathbb{E}(Y|\mathbf{X}_I))}{V_Y} \,, \tag{3.10}$$

$$S_{IJ} = \frac{V(\mathbb{E}(Y|\mathbf{X}_I, \mathbf{X}_J)) - V(\mathbb{E}(Y|\mathbf{X}_I)) - V(\mathbb{E}(Y|\mathbf{X}_J))}{V_Y} \,, \tag{3.11}$$

$$S_{T_I} = 1 - \frac{V(\mathbb{E}(Y|\mathbf{X}_{\bar{I}}))}{V_Y} \,. \tag{3.12}$$

The advantage of this approach is that correlation effects do not influence the sensitivity indices, so contributions between groups can be separated well. Additionally, more efficient computational methods for independent inputs are applicable. However, the downside is that the analysis offers no information about the effects of individual variables inside a group.

Computation of Sobol' indices

In order to compute the Sobol' indices, multidimensional integrals have to be evaluated for the expected values and variances. A common strategy is applying quasi-Monte Carlo methods that use low discrepancy quasi-random sequences such as those suggested by Faure, Niederreiter, Halton, Hammersley, Sobol, and others [50]. These sequences are used in place of uniform random samples as they cover the unit hypercube more uniformly and thus achieve faster convergence of the integral in $O(\frac{1}{n} log(n)^D)$ rather than the $O(n^{-\frac{1}{2}})$ in the Monte Carlo method with random samples [51, 40]. Another widely used method is Latin hypercube sampling, which divides the range for each variable in M equally sized intervals and selects one random value in each interval. In the second step, the values for different variables are combined randomly [52]. Both classes of sampling methods result in an independent uniform distribution for all input variables. In case a different distribution is more suitable, the samples for each variable can subsequently be transformed using the inverse transform theorem [40]: If μ is a measure in $\mathbb{R}$ and $F_\mu(x)$ its cumulative distribution function,

$$U \sim \text{Uniform}([0, 1]) \implies F_\mu^{-1}(U) \sim \mu \,. \tag{3.13}$$

Saltelli et al [53] propose an efficient method for computing first order and total Sobol′ indices, which avoids multiple stages of Monte Carlo samples in the nested integrals for the case of independent input variables by recombining samples. An extension requiring more samples additionally computes the second order Sobol′ indices [53].

For models that are expensive to evaluate, an alternative to Monte Carlo estimates is the Fourier amplitude sensitivity test (FAST) developed by Cukier et al. [54] and extended by Saltelli et al. [55], as it uses fewer model evaluations to calculate sensitivity indices. The idea behind the method is to select points in the input domain by varying each parameter at different frequencies ω_i. A high amplitude of the oscillations at the same frequency ω_i in the model output, as determined through its Fourier spectrum, indicates a strong influence of the corresponding input variable [37, 55].

When using Gaussian processes, the model output $Y = f(X_1, \ldots, X_D)$ is given analytically by the posterior mean. In certain cases, this can be used to simplify the integrals and use numerical integration techniques that are less computationally demanding than Monte Carlo simulations or even find analytical solutions. For example, if all variables are independent and the kernel function can be written as a product of one-dimensional covariances, the calculation of sensitivity indices can be split into one- and two-dimensional integrals [47, 45].

All approaches discussed so far treat Gaussian processes as a deterministic model by only considering the predictive mean for the sensitivity analysis. Viewing the output as the random variable it actually is, i.e.

$$Z(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \, , \tag{3.14}$$

where $m(\mathbf{x}$ and $k(\mathbf{x}, \mathbf{x}')$ denote the posterior mean and covariance, gives rise to sensitivity indices that are random variables themselves [56], such as

$$\tilde{S}_i = \frac{V(\mathbb{E}(Z(\mathbf{X})|X_i))}{V(Z(\mathbf{X}))} \, . \tag{3.15}$$

These probabilistic sensitivity indices contain information from the full probability distribution of $Z(\mathbf{X})$. Their variance thus represents the uncertainty in the sensitivity index due to the predictive (epistemic and aleatoric) uncertainty of the Gaussian process regression model. The distribution of the inner expected values

$$\mathbb{E}(Z(\mathbf{X})|\mathbf{X}_I) \sim \mathcal{GP}(\mathbb{E}(m(\mathbf{X})|\mathbf{X}_I), \mathbb{E}(\mathbb{E}(k(\mathbf{X}, \mathbf{X}')|\mathbf{X}_I)|\mathbf{X}_i')) \tag{3.16}$$

is again a Gaussian process since it is a linear transformation of another Gaussian process $Z(\mathbf{X})$ [56]. However, taking the conditional variance of this random variable is a nonlinear transformation and results in a distribution that cannot be established analytically. Therefore, approximation methods such as Monte Carlo simulations are necessary, drawing different samples from the Gaussian process [56]. Alternatively, under certain modeling restrictions, closed-form expressions for the expected value $\mathbb{E}(\tilde{S}_I)$ and variance $V(\tilde{S}_I)$ exist [45].

3.2 Computational uncertainty – methodical approach

The most computationally expensive step in Gaussian process regression is the inversion of the matrix $\hat{K} := K(X,X) + \sigma^2 I$ required for the calculation of the posterior in Equation 2.13, which we restate here for convenience:

$$\begin{aligned} f(X_*)|X,\mathbf{y} \sim \mathcal{N}(&m(X_*) + K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}(\mathbf{y}-m(X)), \\ &K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*)) \ . \end{aligned} \tag{3.17}$$

The matrix inversion with a runtime in $O(n^3)$ for n data points scales prohibitively for large datasets, so it is necessary to use suitable approximations [4].

Various methods for this task have been proposed. One approach is to reduce the number of training points with localized regression methods that train multiple Gaussian processes, each on a subset of the training data [57, 58]. Inducing point methods only consider one smaller set of $m < n$ inducing points for inference that do not necessarily coincide with the original data [59, 60, 61, 62]. Instead of reducing the number of training points, the problem can also be simplified by approximating the inverse of the full kernel matrix $\hat{K}$ using iterative methods such as conjugate gradients [63, 64, 65]. Alternatively, the inverse of a lower rank approximation of $\hat{K}$ can be used, for example with a partial, pivoted Cholesky decomposition [66, 67] or based on an approximation of the kernel function by a finite basis [68, 69, 70]. Moreover, other methods approximate the posterior based on stochastic variational optimization [71, 72, 73]. All of these methods usually compute an approximation to the mathematical posterior and then use it as a direct replacement. However, doing so ignores the additional uncertainty arising from the limited amount of computations expended [74].

Wenger et al. [74] propose a method to quantify this computational uncertainty. Due to its general formulation, their method is applicable to multiple of the most common iterative Gaussian process approximations, such as methods based on the incomplete Cholesky factorization, conjugate gradients, and inducing points. These methods estimate the computationally expensive part of the posterior mean, i.e., the representer weights

$$\mathbf{v}_* = \hat{K}^{-1}(\mathbf{y} - m(X)) \ , \tag{3.18}$$

by improving their approximation using a one-dimensional projection of the current residual in each iteration. The idea of Wenger et al. is to treat the representer weights

$\mathbf{v}_*$ as a latent random variable. Each iteration of the approximation algorithm performs a Bayesian update on the belief about the representer weights, thereby obtaining both a point estimate and the associated uncertainty. In the following, we describe this method in more detail.

The first step is to define a prior distribution for $\mathbf{v}_*$. Since no computations were performed yet, the prior represents the state of no knowledge about the data and is chosen so that the marginal distribution $\int p(f(X_*)|\mathbf{v}_*)p(\mathbf{v}_*)d\mathbf{v}_*$ is equal to the Gaussian process prior $\mathcal{N}(\mu(X_*), K(X_*, X_*))$. The distribution of the Gaussian process conditioned on $\mathbf{v}_*$ is

$$\begin{aligned} p(f(X_*)|\mathbf{v}_*) = \mathcal{N}(&mX_* + K(X_*, X)\mathbf{v}_*, \\ &K(X_*, X_*) - K(X_*, X)\hat{K}^{-1}K(X, X_*)) \;. \end{aligned} \tag{3.19}$$

The prior is then given by

$$\mathbf{v}_* \sim \mathcal{N}(\mathbf{0}, \hat{K}^{-1}) \;. \tag{3.20}$$

Note that $\hat{K}^{-1}$ remains unknown and does not need to be calculated explicitly at any stage in the approximation algorithm. It is only used here to set up the mathematical framework.

Each iteration then conditions $\mathbf{v}_*$ on a one-dimensional projection of the current residual

$$\mathbf{r}_{i-1} = \hat{K}(\mathbf{v}_* - \mathbf{v}_{i-1}) = \mathbf{y} - m(X) - \hat{K}\mathbf{v}_{i-1} \;. \tag{3.21}$$

The projections

$$\alpha_i := \mathbf{s}_i^\top \mathbf{r}_{i-1} \tag{3.22}$$

are defined by actions $\mathbf{s}_i$, arbitrary vectors given by the approximation algorithm used. For example, the partial Cholesky decomposition uses the unit vectors $\mathbf{e}_i$, targeting exactly one data point at a time, and conjugate gradients with preconditioning uses preconditioned conjugate gradients. Intuitively, the actions weight the approximation error of different data points, and only data points $\mathbf{x}_j$ with $(\mathbf{s}_i)_j \neq 0$ contribute to the current iteration.

Using the distribution from the previous iteration $p(\mathbf{v}_*) = \mathcal{N}(\mathbf{v}_*; v_{i-1}, \Sigma_{i-1})$ as prior, the new distribution conditioned on α_i can be obtained by performing a Bayesian update

$$p(\mathbf{v}_*|\{\alpha_j\}_{j=1}^{i}, \{\mathbf{s}_j\}_{j=1}^{i}) = \frac{p(\alpha_i|\mathbf{s}_i, \mathbf{v}_*)p(\mathbf{v}_*|\{\alpha_j\}_{j=1}^{i-1}, \{\mathbf{s}_j\}_{j=1}^{i-1})}{\int p(\alpha_i|\mathbf{s}_i, \mathbf{v}_*)p(\mathbf{v}_*|\{\alpha_j\}_{j=1}^{i-1}, \{\mathbf{s}_j\}_{j=1}^{i-1})d\mathbf{v}_*} \;. \tag{3.23}$$

Since both distributions in the equation are Gaussian, the updated distribution

$$p(\mathbf{v}_*|\{\alpha_j\}_{j=1}^{i}, \{\mathbf{s}_j\}_{j=1}^{i}) = \mathcal{N}(\mathbf{v}_*; \mathbf{v}_i, \Sigma_i) \tag{3.24}$$

is again Gaussian and defined by

$$\mathbf{v}_i = \mathbf{v}_{i-1} + \Sigma_{i-1}\hat{K}\mathbf{s}_i(\mathbf{s}_i^\top \hat{K}\Sigma_{i-1}\hat{K}\mathbf{s}_i)^{-1}\alpha_i = C_i(\mathbf{y} - m(X)) \,, \tag{3.25}$$

$$\Sigma_i = \Sigma_{i-1} - \Sigma_{i-1}\hat{K}\mathbf{s}_i(\mathbf{s}_i^\top \hat{K}\Sigma_{i-1}\hat{K}\mathbf{s}_i)^{-1}\mathbf{s}_i^\top \hat{K}\Sigma_{i-1} = \hat{K}^{-1} - C_i \tag{3.26}$$

where $C_i := S_i(S_i^\top \hat{K} S_i)^{-1} S_i^\top$ is a rank-i matrix. The most expensive computations in one iteration are matrix-vector products, so the algorithm has a runtime in $O(in^2)$ for a fixed number of i iterations and n training data points. If the actions $\mathbf{s}_i$ are linearly independent, the method converges in at most n iterations, the convergence rate depending on the choice of actions. This means after n iterations, $C_i = \hat{K}^{-1}$ and the computational uncertainty about $\mathbf{v}_*$ in terms of the variance Σ_n is zero.

In practice, only a smaller number of iterations are carried out, considering the limited computational budget. The resulting belief $p(\mathbf{v}_*) = \mathcal{N}(\mathbf{v}_*; \mathbf{v}_i, \Sigma_i)$ can be incorporated into the Gaussian process posterior by marginalization as follows:

$$\begin{aligned} p(f(X_*)) &= \int p(f(X_*|\mathbf{v}_*))p(\mathbf{v}_*)d\mathbf{v}_* \\ &= \mathcal{N}(f(X_*); m(X_*) + K(X_*, X)\mathbf{v}_i, K(X_*, X_*) - K(X_*, X)C_i K(X, X_*)) \,. \end{aligned} \tag{3.27}$$

The resulting combined uncertainty is then given by

$$\begin{aligned} &K(X_*, X_*) - K(X_*, X)C_i K(X, X_*) = \\ &\qquad \underbrace{K(X_*, X_*) - K(X_*, X)\hat{K}^{-1}K(X, X_*)}_{\text{mathematical uncertainty}} + \underbrace{K(X_*, X)\Sigma_i K(X, X_*)}_{\text{computational uncertainty}} \end{aligned} \tag{3.28}$$

and can be calculated without any knowledge of $\hat{K}^{-1}$, even if the individual components for the mathematical and computational uncertainty depend on it.

While the mathematical uncertainty stems from a limited amount of data and is high in regions of the input space not covered well by training data, computational uncertainty is high in regions covered by training data that have not been targeted by computations yet. This behavior is illustrated in Figure 3.1.

As stated above, the asymptotic runtime of the algorithm is $O(in^2)$ for all choices of actions. Therefore, the explicit modeling of computational uncertainty does not affect the asymptotic behavior of methods like Cholesky factorization and conjugate gradients. Moreover, it eliminates the need for additional computations to obtain the posterior covariance. For example, computing the posterior covariance with conjugate gradients requires m additional linear solves, where m is the size of the test data set, and this covariance does not contain information about computational uncertainty. In contrast, inducing point methods can usually be computed in linear time, so their asymptotic runtime increases using the algorithm discussed here. However, apart from offering information about the computational uncertainty, this algorithm can also significantly improve accuracy in terms of NLL and RMSE, as demonstrated by Wenger et al. [74] on various datasets. Thus, it can be the better choice if one can afford the increased computational cost.

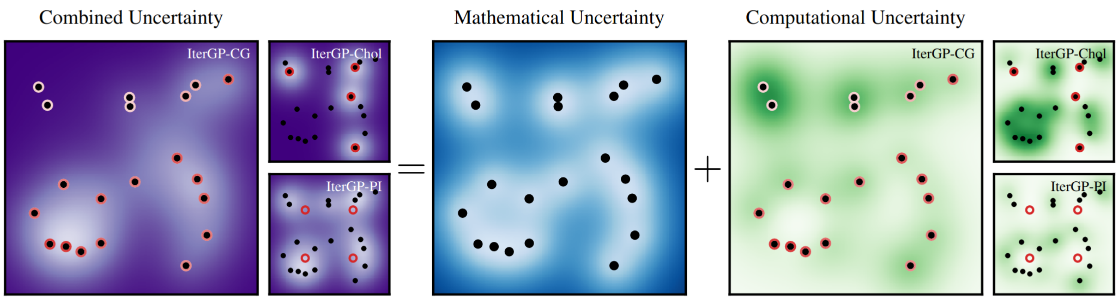


Figure 3.1: Distribution of mathematical and computational uncertainty over the input space after four iterations of the respective iterative algorithms. Black dots denote training points and red circles indicate where computation has been targeted, defined by the magnitude of the actions s_i at the component corresponding to the training point. While the variant for conjugate gradients (IterGP-CG) reduces uncertainty globally, the other two variants for the incomplete Cholesky factorization (IterGP-Chol) and for inducing point methods (IterGP-PI) target individual points at a time, thus reducing uncertainty locally. The mathematical uncertainty is high in regions with no training data, whereas computational uncertainty is high in regions with training data, where computations have not yet been targeted. (Source: [74]).

3.3 Calibration – methodical approach

A regression model is usually not able to predict the target variable with perfect accuracy. The same holds for the predicted uncertainty in the form of the output probability distribution for probabilistic regression models. During training, the objective function typically aims to minimize the prediction error but does not take into account the calibration of the model, i.e., its ability to accurately predict uncertainty.

For Gaussian processes, the predictive quality for both the target variable and uncertainty greatly depends on the choice of hyperparameters [75]. These are usually optimized using the marginal log likelihood $\log p(\mathbf{y}|X, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the hyperparameters. This approach consistently leads to a satisfactory mean square error on test data [4]. However, the posterior variance tends to be too low in practice, leading to overly confident predictions [76]. This problem is linked to the use of the log likelihood in the optimization objective. Since it only measures the predictive quality on the existing data but not on unseen data, it cannot assess how well the posterior variance explains the model error [75]. For example, if the predictive accuracy on the training data is already high, the marginal log likelihood can be further improved by decreasing the variance, a problem also observed in neural networks [77]. Therefore, to ensure reliable predictions of uncertainty, methods to improve calibration are required.

3.3.1 Definitions of calibration for regression

In a classification setting, a model is said to be calibrated if, for any group of instances receiving the same predicted probability p for some class, the fraction of instances actually belonging to that class is close to p [78]. While this definition is generally agreed upon in the literature [78, 79, 80, 81, 82], the same is not the case for calibration for regression models. As this field is relatively new, there is no general consensus about the definition and evaluation metrics [83]. This section summarizes three important definitions currently used in the literature.

Quantile calibration

Quantile calibration stems from the idea that for a predicted 95% confidence interval, the true value of y should fall within that interval for approximately 95% of all predictions [84]. Such a definition is especially useful for forecasting tasks such as energy usage or supply chain optimization [85]. Formally, a regressor is quantile calibrated if

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}\{y_n \leq F_n^{-1}(p)\} \to p \text{ for all } p \in [0,1] \text{ as } N \to \infty \tag{3.29}$$

for a set of data $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, where F_n denotes the predicted cumulative distribution at $\mathbf{x}_n$ and F_n^{-1} the corresponding quantile function [84]. This definition can also be extended to two-sided confidence intervals. If a regressor is quantile calibrated,

$$\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}\{F_n^{-1}(p_1) \leq y_n \leq F_n^{-1}(p_2)\} \to p_2 - p_1 \text{ for all } p_1, p_2 \in [0,1] \text{ as } N \to \infty \tag{3.30}$$

holds as well. A sufficient condition for quantile calibration is

$$\mathbb{P}\left(Y \leq F_{Y|X}^{-1}(p)\right) = p \text{ for all } p \in [0,1] \tag{3.31}$$

if the $(\mathbf{x}_n, y_n)$ are i.i.d. realizations of random variables $X, Y \sim \mathbb{P}$ [84]. $F_{Y|X}$ denotes the predicted cumulative distribution of Y conditioned on X.

Since this definition only considers marginal distributions, it represents a global view of calibration. Hence, it is useful for making statements about all predictions together, but the individual calibrated distributions at each location do not necessarily resemble the true distribution of Y there [85].

Distribution calibration

The definition of distribution calibration focuses on making locally calibrated predictions for each individual model output. This requirement is similar to calibration in classification, which groups instances by their predicted probability and requires each of these groups to match that probability. The idea of conditioning calibration requirements on model predictions can also be applied to regression problems.

Let $\mathcal{Y}$ denote the set of all possible target values y and $\mathcal{S}$ the set of possible probability distributions $S : \mathcal{Y} \to [0, 1]$ where $f_{Y|X}(y) \in \mathcal{S}$ are densities predicted by a regression model. Then, according to [85], the model is distribution calibrated if

$$p\left(Y = y | f_{Y|X} = s\right) = s(y) \text{ for all } y \in \mathcal{Y},\ s \in \mathcal{S}\ . \tag{3.32}$$

This definition requires that for any predicted distribution, on average over all cases receiving this prediction, the target value follows this distribution [85]. For regression models that predict normal distributions, such as Gaussian processes, this means that among all instances receiving the same predicted mean μ and variance σ^2, on average, the target mean is μ and the target variance is σ^2. Song et al. show that a distribution calibrated regressor is always quantile calibrated, but the inverse is not necessarily true, making it a stronger calibration requirement [85].

Variance calibration

Variance calibration is also closely related to the definition of calibration for classification. The difference to distribution calibration is the transfer of concepts from classification to regression. Distribution calibration views the predicted probability p of a classifier as the definition of a probability distribution over possible outcomes for the binary target variable, making it equivalent to the predictive distribution over continuous values for the target in the regression case. In contrast, variance calibration takes a different perspective and interprets the information conveyed by p as the probability of misclassification given by $1 - p$ [77]. Thus, the corresponding concept in regression is the expected predictive error $\mathbb{E}_{X,Y}[\mu(\mathbf{x}) - y]$. The goal of calibration is then to make the model predict its own error correctly so that the predicted variance σ^2 matches the expected (squared) error at each location. Formally, a regression model is variance calibrated [77] if

$$\mathbb{E}_{X,Y}\left[(\mu(\mathbf{x}) - y)^2 | \sigma(\mathbf{x})^2 = \sigma^2\right] = \sigma^2 \text{ for all } \sigma^2 \,. \tag{3.33}$$

Since the predictions are grouped by their variance σ^2, this definition is local in a similar sense as for distribution calibration. However, it only includes the variance and not the mean of the prediction. Therefore, the calibration is entirely decoupled from the predictive accuracy and changing the calibration of a model does not affect its point estimates but only the model's uncertainty about them [77].

3.3.2 Methods for regression calibration

There are two different approaches to obtaining a calibrated model. Post-hoc calibration operates on a model that has already been trained and transforms its output distribution so that it matches some calibration criteria. Conversely, loss-based methods are directly incorporated into the training procedure, penalizing poorly calibrated models as part of the loss function during the optimization of parameters or hyperparameters. In the following, we give an overview of different calibration methods from both of these categories.

Post-hoc calibration methods

Post-hoc calibration methods aim to transform the model output to improve its calibration. Formally, given the output cumulative distribution function $F_{Y|X}(y)$, the goal is to find a mapping $R : [0,1] \to [0,1]$ so that $R \circ F_{Y|X}(y)$ is calibrated. For quantile calibration, Kuleshov et al. suggest a method based on isotonic regression, which applies the same mapping R to all predictions [84]. In order to achieve distribution calibration, a more flexible, local approach is necessary, where R can vary across different predictions. Song et al. [85] use Gaussian processes to learn a mapping from regression output to parameters of their transformation $R_{\hat{y}}$. The Gaussian process allows to estimate the distribution of target variables at each regression output using just one sample per location by leveraging target values from nearby locations [85]. Both these methods make no assumptions about a parametric form of the calibrated distribution and represent it only through values of their cumulative distribution functions. However, in some cases, such as Kalman filtering in object detection, a closed form of the output distribution is required [83]. Methods based on variance scaling address this issue by not changing the type of the predictive distribution but only adjusting its variance. This has the additional benefit of decoupling the calibration of a model from its accuracy since the predicted values remain the same after calibration [77]. A simple approach for improving the variance calibration of models that systematically under-

or overestimate their uncertainty is temperature scaling. A global scale factor s for the predicted variance is introduced and optimized using the negative log likelihood on a calibration set separate from the training data [77]. If one global scale parameter does not suffice, an approach based on Gaussian processes analogous to the one introduced by Song et al. [85] can be used to learn individual scale factors for each location [83].

In order to avoid overfitting during the calibration procedure, it is advisable to use a set of *calibration data* separate from the *training data* for this task. Kuleshov et al. suggest an approach inspired by cross validation, where K models are trained on different splits of the data, and the hold-out serves as calibration set [84].

Loss-based calibration methods

A different approach for regression calibration is to integrate it directly into the training procedure. This integration can be achieved by introducing a term in the loss function that evaluates the calibration quality. For variance calibration, the calibration loss can be employed, which measures the difference between the predicted variance and the squared error [86]. The maximum mean discrepancy (MMD) loss targets quantile calibration. It measures the distance between the distributions over the targets y and predicted values $\hat{y}$ in a reproducing kernel Hilbert space (RKHS) [87]. The distributions can be embedded in this space using the mean embedding, which yields a unique representation for each probability distribution. The distance between two distributions in the RKHS norm becomes zero exactly if the two distributions are identical, making it a measure for similarity [88]. If distribution calibration is desired, the f-cal framework introduces a loss function that compares some statistic of standardized residuals to its expected distribution if the model were perfectly calibrated using an f-divergence. For example, for a Gaussian output distribution, the normalized residuals are given by $z_i = \frac{y_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)}$. If the true targets follow the individual output distributions at each location, the residuals are expected to follow a standard normal distribution and their sum of squares a Chi-squared distribution. Thus, the similarity of the empirical distribution to a Chi-squared distribution can indicate how well the model is calibrated. By standardizing the residuals, this method can evaluate the calibration at all locations simultaneously even if the predicted distributions are different [89].

In their recent work, Capone et al. [75] introduce another loss-based method specifically built for Gaussian process regressors. They argue that one set of hyperparameters does not suffice to represent both the predictions in terms of the posterior mean and the uncertainty of the model given by the posterior variance. Thus, they suggest using two different Gaussian process models, one for its mean and the other for its variance. The first Gaussian process is trained regularly, for example, using the marginal log likelihood on a set of training data. For the second model, a different objective function is introduced, with one term measuring how well the model is calibrated in terms of quantile calibration and another term penalizing large variances in order to obtain a sharp predictor. While this Gaussian process still uses the same set of training data for inference, another set of calibration data is used to evaluate the objective function in order to prevent overfitting [75].

Comparison of the two categories

Loss-based calibration methods can directly be integrated into the training procedure and modify the model only through its parameters and hyperparameters, thus retaining all its properties. Furthermore, they often do not require an additional calibration set. However, modifying the objective function can also increase the computational cost of the optimization [78]. Post-hoc methods offer greater flexibility since they introduce further degrees of freedom and can modify the predictive distribution beyond the capabilities of the original model. Due to this flexibility, the focus in the following will be on post-hoc methods. The following sections describe three of these methods in more detail, each targeting a different definition of calibration.

3.3.3 Quantile calibration based on isotonic regression

Kuleshov et al. [84] propose a simple method to achieve quantile calibration using isotonic regression, inspired by similar procedures in classification such as Platt scaling. The goal is to find a mapping $R : [0,1] \to [0,1]$ so that $R \circ F_{Y|X}$ is quantile calibrated. Therefore, R needs to be constructed so that, given a predicted quantile $q = F_{Y|X}^{-1}(p)$, the corresponding probability p is mapped to the true probability of Y falling below that quantile value q [84]. That is,

$$R(p) := \mathbb{P}\left(Y \leq F_{Y|X}^{-1}(p)\right) . \tag{3.34}$$

Since the distribution of Y can only be accessed through the available samples, it has to be approximated. Given a calibration data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, the values of R at these points can be estimated by

$$\hat{R}(p) := \frac{1}{N} \left|\{y_n | F_n(y_n) \leq p, n = 1, \dots, N\}\right| , \tag{3.35}$$

the fraction of the data for which y_n lies below its predicted p-quantile [84]. A continuous map can now be constructed by fitting any regression algorithm to the *recalibration set* $\{F_n(y_n), \hat{R}(F_n(y_n))\}_{n=1}^N$. Kuleshov et al. suggest using isotonic regression since it only predicts monotonically increasing functions and does not assume a parametric form, making it very flexible [84].

The objective of isotonic regression on a set of pairs $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$ is to find values $\hat{y}_i$ that minimize $\sum_i (y_i - \hat{y}_i)^2$ subject to $\hat{y}_i \leq \hat{y}_j$ whenever $x_i \leq x_j$ [90]. One commonly used algorithm for this task is the Pool-Adjacent-Violators algorithm (PAVA) [91]. It starts by assigning $\hat{y}_i = y_i$ for all i and ordering them by their x-values. Then, as long as there are pairs $\hat{y}_i$, $\hat{y}_{i+1}$ so that $\hat{y}_i > \hat{y}_{i+1}$, violating the monotonicity condition, both are replaced by their average $\frac{\hat{y}_i - \hat{y}_{i+1}}{2}$. In the final step, the algorithm interpolates between the existing values, usually by piecewise linear or constant interpolation [91].

While this calibration method has been shown to achieve good results in terms of quantile calibration [84, 75, 83], it is also capable of perfectly calibrating random uncertainty estimations that do not have any connection to the data, as demonstrated both analytically and empirically by Levi et al. [77]. Therefore, it is not suitable for safety-critical applications, where basing decisions on unreliable uncertainty estimates can pose a significant risk [77]. Moreover, it is important to use a sufficiently large set of calibration data since the method is prone to overfit for smaller datasets [85].

3.3.4 Local calibration based on Gaussian processes

Calibration according to the two local definitions, distribution and variance calibration, is more difficult than the global quantile calibration. In order to achieve local calibration, the calibration map R needs to be defined individually for each possible model output. Because there is usually not more than one target value per predicted distribution in the calibration set, it is not possible to estimate the empirical distribution based on samples [85]. Therefore, Song et al. introduce a method based on Gaussian processes, which utilizes the target values for predictions close by [85]. Their idea is a two-stage procedure where they first specify a parametric class of functions for the calibration map. In a second step, this calibration map is made dependent on the regression output (μ, σ) by modeling its parameters $\mathbf{w}$ as functions $\mathbf{w}(\mu, \sigma)$ using a Gaussian process. As mentioned above, the choice of Gaussian processes makes it possible to use target values y corresponding to similar regression output distributions for estimating their local distributions through the choice of a suitable kernel. Since the Gaussian process does not specify a single function but a distribution over functions, Song et al. choose to treat $\mathbf{w}(\mu, \sigma)$ as a latent random variable and average over it in a Bayesian manner.

With this approach, the marginal likelihood of target values $\mathbf{y}$ is given by [85]

$$p(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\sigma}) = \int_{\mathbf{w}} \underbrace{p(\mathbf{y}|\mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\sigma})}_{\text{calibration map}} \underbrace{p(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\sigma})}_{\text{GP model}} \, d\mathbf{w} = \int_{\mathbf{w}} (\prod_{i=1}^{n} p(y_i|\mathbf{w}_i), \mu_i, \sigma_i) p(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\sigma}) d\mathbf{w} \, . \tag{3.36}$$

$\mathbf{w} = [\mathbf{w}_1^\top, \ldots, \mathbf{w}_N^\top]$ denotes the parameters for the calibration map at each point in the calibration set, that may themselves be vectors if the calibration map has multiple parameters. $(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \{(\mu_1, \sigma_1), \ldots, (\mu_N, \sigma_N)\}$ are the outputs of the regression model at the points in the calibration set. In this context, we identify the output distributions by their mean μ and standard deviation σ, assuming that they are normal, as it is the case for Gaussian process models. However, it is worth noting that this method can also be extended to regressors with different output distributions, parametric or nonparametric. Moreover, the implied independence of the y_i by their factorization may not hold in general. For example, for a Gaussian process regression model, the output values can be correlated. However, the definition of distribution calibration does not take into account any dependencies between output values. Therefore, this method only operates on marginal distributions, and the additional information from the joint distribution is not used.

The calibrated density $\hat{f}_*(y)$ given a new prediction (μ_*, σ_*) can then be expressed as

$$\hat{f}_*(y) = \int_{\mathbf{w}_*} \int_{\mathbf{w}} p(y|\mathbf{w}_*) p(\mathbf{w}_*|\mathbf{w}) p(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\sigma}) d\mathbf{w} d\mathbf{w}_* \ , \tag{3.37}$$

where $p(y|\mathbf{w}_*)$ is defined by the calibration map, $p(\mathbf{w}_*|\mathbf{w})$ is the Gaussian process posterior using the current value of $\mathbf{w}$ as training data. The term $p(\mathbf{w}|\mathbf{y}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ is proportional to the product of individual likelihoods and the prior, i.e.,

$$p(\mathbf{w}|\mathbf{y}, \boldsymbol{\mu}, \boldsymbol{\sigma}) \propto (\prod_{i=1}^{n} p(y_i|\mathbf{w}_i)) p(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\sigma}) \ . \tag{3.38}$$

Note that this approach does not use classic Gaussian process regression for the parameters $\mathbf{w}$, since there are no available ground truth values corresponding to the training data $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ from the calibration set. Instead, all possible values $\mathbf{w}$ for the parameters in the training set are considered and marginalized according to their prior distribution.

This method can also be extended to regressors with multidimensional output. By jointly modeling the parameters $\mathbf{w}$ for each of the output dimensions with one extended Gaussian process, their covariance structure can be taken into account for the calibration [83]. In the following, we will focus on the one-dimensional case in accordance with the scope of this thesis.

Beta map for distribution calibration

For distribution calibration, Song et al. suggest using functions from the beta family as calibration map. They are defined by

$$c_\beta : [0,1] \to [0,1], \ c_\beta(p) = \phi\left(a \ \ln(p) - b \ \ln(1-p) + c\right) \ , \tag{3.39}$$

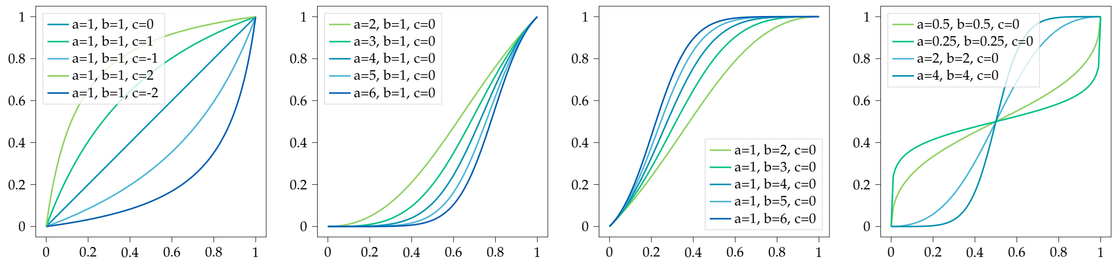


Figure 3.2: Instances of the beta map family for different values of the parameters.

where $\phi(z) = (1 + e^{-z})^{-1}$ is the logistic sigmoid function [85]. Beta functions are also used in calibration for classification [92] since they are defined on the correct interval $[0, 1]$ and provide flexible shapes that can be controlled with the parameters a, b and c [85]. Changing c pushes the distribution to the left for $c < 0$ or to the right for $c > 0$. If a and b are both greater than one, the function takes on a sigmoid shape, decreasing the variance, whereas $a, b < 1$ leads to an inverse sigmoid shape, which increases the variance. Changing the balance between a and b creates a left skew for $a < b$ or a right skew for $a > b$. The configuration $a = b = 1, c = 0$ results in the identity function, allowing to make no modifications in case the model is already calibrated [85]. Instances of functions from the beta family for different parameter values are shown in Figure 3.2. In order for the mapping to yield a valid cumulative distribution function, it must be monotonically increasing. For this requirement, $a, b \geq 0$ is necessary and sufficient [85].

The beta calibration map c_β is defined on the cumulative distribution function. However, in order to compute the likelihood in Equation 3.36, an expression for the transformed density is required. It is given by the derivative

$$\frac{\mathrm{d}\, c_\beta(F_{Y|X}(y))}{\mathrm{d}\, y} = \frac{\mathrm{d}\, c_\beta(F_{Y|X}(y))}{\mathrm{d}\, F_{Y|X}(y)} \frac{\mathrm{d}\, F_{Y|X}(y)}{\mathrm{d}\, y} = r_\beta \left(F_{Y|X}(y)\right) f_{Y|X}(y)\,, \tag{3.40}$$

where $f_{Y|X}(y)$ is the predicted density and r_β is defined in [85] as

$$r_\beta(p) = \frac{p^a(1-p)^b e^{-c}(a-(a-b)p)}{p(1-p)(p^a-(1-p)^b e^{-c})^2}\,. \tag{3.41}$$

The next step is to model the parameters a, b, c as functions of the model output using a Gaussian process. This Gaussian process needs to be defined over probability distributions. Thus, a sensible choice for the kernel is a kernel mean map, which uniquely represents any distribution in an infinite dimensional RKHS [93]. The feature map is defined as $p \to \mathbb{E}_{x\sim p}(\phi(x))$ for universal kernels of the form $k'(x, x') =$

$\langle \phi(x), \phi(x') \rangle$ [93]. Since the distance between two embeddings in the RKHS can be interpreted as a measure of similarity between the corresponding distributions [93], this choice represents the initial assumption that the calibration map should be similar for similar model outputs [85]. For normal distributions and the squared exponential kernel as k', an analytical solution for the embedding exists, and the resulting kernel is

$$k\left((\mu_1, \sigma_1), (\mu_2, \sigma_2)\right) = \frac{\theta}{|\sigma_1 + \sigma_2 + \theta^2|^{\frac{1}{2}}} e^{\left(-\frac{(\mu_1 - \mu_2)^2}{2(\sigma_1 + \sigma_2 + \theta^2}\right)} , \tag{3.42}$$

where θ is a hyperparameter [93, 85].

Moreover, since there are three potentially correlated parameters, Song et al. [85] use a multi output Gaussian process. The Gaussian process prior is then given by

$$\mathbf{w} = \begin{bmatrix} w_a \\ w_b \\ w_c \end{bmatrix} \sim \mathcal{GP}(\mathbf{0}, B \otimes K) , \tag{3.43}$$

where K is obtained by applying the previously chosen kernel to the calibration data $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and $B \in \mathbb{R}^{3\times 3}$ represents the correlation between a, b and c [85]. In order to enforce the constraints on a and b and introduce additional flexibility through further parameters, the output of the Gaussian process is transformed by

$$\begin{aligned} a &= e^{(\gamma_a^{-1} w_a + \delta_a)} , \\ b &= e^{(\gamma_b^{-1} w_b + \delta_b)} , \\ c &= \gamma_c^{-1} w_c + \delta_c , \end{aligned}$$

before inserting the parameters in the beta link function. The exponential functions ensure that a and b are always positive [85].

With these modeling choices, the likelihood in Equation 3.36 becomes

$$p(y_i | \mathbf{w}_i, \mu_i, \sigma_i) = f_{Y|\mathbf{x}_i}(y_i) r_\beta(F_{Y|\mathbf{x}_i}(y_i)) \tag{3.44}$$

where μ_i and σ_i define $f_{Y|\mathbf{x}_i}$ and $F_{Y|\mathbf{x}_i}$ and

$$p(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, B \otimes K) \tag{3.45}$$

is the likelihood of the Gaussian process [85].

Variance scaling for variance calibration

Küppers et al. suggest another calibration method building on this principle, which targets variance calibration [83]. Since only the variance needs to be modified, a simpler calibration map that acts directly on the density can be formulated by scaling the predicted variance with a scaling parameter s. It is defined as

$$r(p) = \mathcal{N}(\mu, s \cdot \sigma^2) \; , \tag{3.46}$$

where μ and σ are the model predictions, given that they define a Gaussian distribution [83]. The method can also be applied to other distributions, for example the Cauchy distribution, where the scale parameter is scaled instead of the variance [83].

Generally, this choice of distribution transformation interferes only little with the predicted distribution, conserving its parametric form and not affecting the mean and, thus, the point estimates. Since there is only one parameter, it can be modeled by a single output Gaussian process

$$w \sim \mathcal{GP}(0, K) \; , \tag{3.47}$$

where the same kernel as above is used for constructing K. In order to ensure positive variances, s has to be positive. Similar to the beta method, this is achieved through the mapping

$$s = e^w \; , \tag{3.48}$$

here without introducing further hyperparameters [83]. Using this method, the likelihood in Equation 3.36 is

$$p(y_i | \mathbf{w}_i, \mu_i, \sigma_i) = \mathcal{N}(\mu_i, s_i \cdot \sigma_i^2) \tag{3.49}$$

and

$$p(w | \boldsymbol{\mu}, \boldsymbol{\sigma}) = \mathcal{N}(w; 0, K) \; . \tag{3.50}$$

Parameter optimization and inference

Once a suitable model has been chosen, it remains to optimize the parameters and hyperparameters by maximizing the marginal likelihood in Equation 3.36. These include the kernel hyperparameter θ and for the beta calibration, additionally the correlation matrix B and the link parameters γ and δ for each of the three beta parameters. In order to reduce the computational cost of Gaussian process inference during optimization, Song et al. suggest using a sparse Gaussian process approximation based on a set of induced pseudo points [85, 59].

Moreover, the integrals for both the likelihood (Equation 3.36) and inference (Equation 3.37) are analytically intractable due to the nonlinearity introduced by the link function [85]. Therefore, Song et al. [85] use an approximation based on variational inference, optimizing the evidence lower bound (ELBO) with some further modifications instead of the likelihood directly. This technique stems from Gaussian process classification, where similar integrals arise from transforming the regression output to a class probability [94].

3.3.5 Measures for calibration

This section provides an overview of metrics for assessing model calibration. Although multiple metrics exist for each calibration definition, we will focus on one metric for each.

The **mean pinball loss ($\overline{\textbf{PBL}}$)** is a measure for quantile calibration. It is based on the pinball loss commonly used to train quantile regression models [95]. The asymmetric loss function is defined for each quantile individually and weights the errors of overestimation and underestimation based on the quantile [85]. It is defined as

$$\mathcal{L}_{\text{Pin}}(\tau, y) = \begin{cases} \tau(y - F_{Y|\mathbf{X}}^{-1}(\tau)) & y \geq F_{Y|\mathbf{X}}^{-1}(\tau) \\ (1-\tau)(F_{Y|\mathbf{X}}^{-1}(\tau) - y) & y \leq F_{Y|\mathbf{X}}^{-1}(\tau) \end{cases} \tag{3.51}$$

for a predicted inverse cumulative distribution function $F_{Y|\mathbf{X}}^{-1}$ and quantile $\tau \in [0, 1]$ [85]. The average loss over all instances in the test set and a set of quantiles

$$\overline{PBL} = \frac{1}{N_\tau} \frac{1}{N_y} \sum_{\tau} \sum_{y} \mathcal{L}_{\text{Pin}}(\tau, y) \ , \tag{3.52}$$

where N_τ denotes the number of quantiles and N_y the size of the test set, can then be used as a measure for the overall quantile calibration [83]. In the following, we use quantiles ranging from 0.05 to 0.95 in increments of 0.05 in order to capture the quality of quantile calibration over a wide range of values.

The **negative log likelihood (NLL)** can be used to measure distribution calibration as suggested by Song et al. [85], since it measures how likely the observed test data was generated by the model taking into account the predictive distribution at each individual point. It is defined by

$$NLL = -\sum_{y} \log p\left(f_{Y|\mathbf{X}}(y)\right) \ , \tag{3.53}$$

where y are instances in the test set and $f_{Y|\mathbf{X}}$ denotes the predicted probability density function.

The **expected normalized calibration error (ENCE)**, proposed by Levi et al. [77], measures calibration in terms of the scaled difference between the squared error and the predicted variance. Hence, it serves as an indicator for variance calibration. Inspired by similar methods in classification, the ENCE is based on a binning scheme. It divides the test data into N bins $\{B_j\}_{j=1}^N$ for the predicted variance $\sigma^2(\mathbf{x})$. The instances are ordered by variance and assigned consecutive subsets of the same size to each bin. The root mean variance

$$RMV(B_j) = \sqrt{\frac{1}{|B_j| \sum_{y_t \in B_j} \sigma^2(\mathbf{x}_t)}} \tag{3.54}$$

and the root mean square error

$$RMSE(B_j) = \sqrt{\frac{1}{|B_j| \sum_{y_t \in B_j} (y_t - \hat{y}_t)^2}} \tag{3.55}$$

are then compared for each bin to assess the variance calibration. $\hat{y}_t$ is the model prediction at $\mathbf{x}_t$, for a Gaussian process, it is equal to the predicted mean $\mu(\mathbf{x}_t)$. Levi et al. [77] suggest the metric

$$ENCE = \frac{1}{N} \sum_{j=1}^{N} \frac{RMV(B_j) - RMSE(B_j)}{RMV(B_j)} \tag{3.56}$$

as a summary of all bins. The calibration error in each bin is normalized by the corresponding mean predicted variance since the expected error increases with greater variances [77].

Apart from the quality of the calibration, another important property of a probabilistic regression model is sharpness [84]. Calibration does not necessarily guarantee that the model output is informative. For example, a model that always predicts $\mathbb{P}(y \leq Y)$ is perfectly quantile calibrated but cannot be accurate since it does not depend on the input value X [84]. The confidence intervals of a sharp model should be as narrow as the aleatoric uncertainty allows [96, 75]. For a deterministic relationship between input and output values, that means that the predicted probabilities should be close to zero or one [84]. We examine the effect of each calibration procedure on the model's sharpness using the **mean prediction interval width (MPIW)**. For a confidence level α, it is defined as the width of the corresponding confidence interval [97],

$$\frac{1}{N} \sum_{n=1}^{N} \left(F_{Y|\mathbf{X}}^{-1} \left(1 - \frac{\alpha}{2} \right) - F_{Y|\mathbf{X}}^{-1} \left(\frac{\alpha}{2} \right) \right) . \tag{3.57}$$

We use the common choice of $\alpha = 0.05$ denoting a 95% confidence interval.

3.4 Data provision for wheeled locomotion

In the following sections, the previously discussed methods for uncertainty quantification are applied to a practical scenario. Specifically, we focus on modeling the locomotion of a wheeled planetary rover. This section describes the data used for building and evaluating the base Gaussian process model, on which we then perform sensitivity analysis, quantify computational uncertainty, and apply calibration techniques. Simulating the wheel's movement involves forecasting the forces and torques that will impact it along its intended path [7]. In the following, we focus on predicting the traction force acting on the wheel in the direction of primary movement. The traction force is the force that the driven wheel momentarily exerts against all resistances such as friction, inertial forces, and the downhill force if locomotion is uphill [98]. The input variables for the Gaussian process model are chosen so that they can be provided by the onboard sensors of a rover. Table 3.1 summarizes these variables and Figure 3.3 provides a physical representation.

Adopting the framework of [7], all quantities are represented in the wheel coordinate system denoted by w, which is positioned at the wheel center. The y-axis is aligned with the rotational axis of the wheel, and the x-axis is parallel to the wheel reference plane in the direction of motion. This plane is constructed as the best-fit plane to the ground for a circular area with a radius equal to the wheel radius.

The input variables are the wheel's velocity $\vec{v}_w$, angular velocity $\vec{\omega}_w$ and position $\vec{p}_w$. The position is defined relative to the intersection between the wheel's z-axis and the reference plane. Additionally, we use the downhill force F_d in the driving direction as input, which is proportional to $\sin\left(\arctan\left(F_{g,x}, F_{g,z}\right)\right)$. $F_{g,x}$ and $F_{g,z}$ denote the gravity

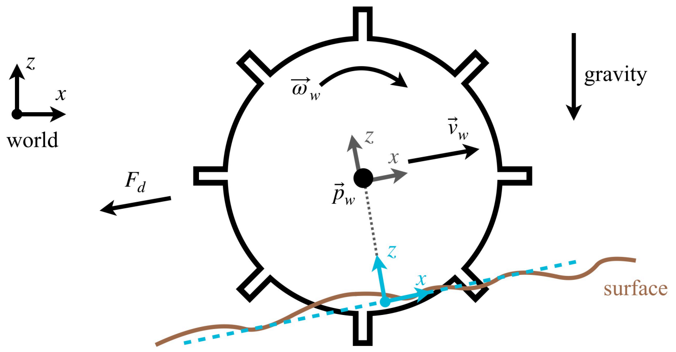


Figure 3.3: Representation of the data available as input to the Gaussian process model. The coordinate system in grey is the wheel coordinate system. The reference plane and coordinate system are depicted in light blue. Adapted from [7].

components in x and in z-direction relative to the wheel coordinate system. This choice led to a lower generalization error on the validation set compared to directly including the gravity components in the model.

The training data is generated using the soil contact model (SCM) [99]. As a semi empirical model, it implements a trade-off between cheap to evaluate but less accurate empirical models and computationally expensive, theoretical models that produce higher quality results. SCM explicitly models the soil deformation by representing the soil with an equidistant set of nodes and modifying their height throughout the course of a simulation. Additionally, the soil has internal states to reflect the soil conditions [99]. Figure 3.4 shows the path created by a rover in a SCM simulation.

The training data consist of independent runs, each representing movement in a straight path over randomly generated terrain with slopes of up to 12 degrees. The input variables and traction force are sampled ten times per second over the span of 15.1 seconds per run.

Prior to training the model, we apply some preprocessing steps to the data. Since Gaussian process models do not perform well for input dimensions with different scales, we first scale all input variables and the output to zero mean and unit variance. Furthermore, frequencies higher than 2 Hz in the traction force are considered to be noise and filtered out using frequency domain filtering.

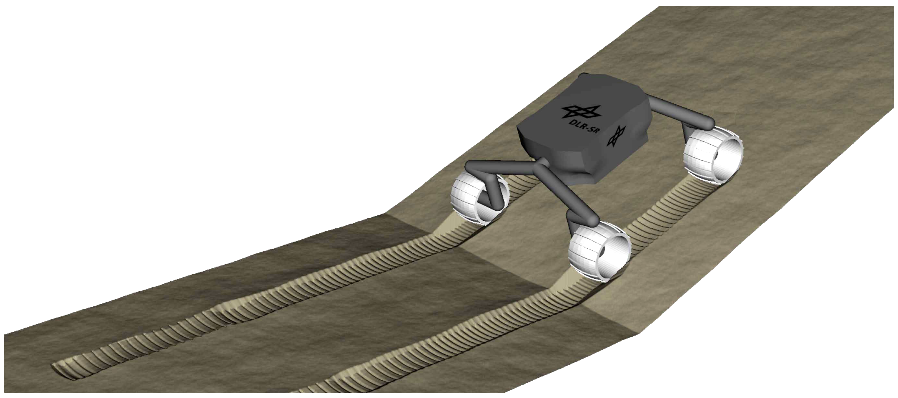


Figure 3.4: Soil deformation by the rover in a SCM simulation. (Source: [100]).

Table 3.1: Overview of the input variables.

Variable	**Description**
$\vec{p}_w = [p_x, p_y, p_z]$	Relative position of the wheel w.r.t. the surface
$\vec{v}_w = [v_x, v_y, v_z]$	Velocity of the wheel
$\vec{\omega}_w = [\omega_x, \omega_y, \omega_z]$	Angular velocity of the wheel
F_d	Downhill force acting on the wheel against the direction of motion

3.5 Building the base Gaussian process model

Based on the data detailed in the previous section, we build a Gaussian process model as the foundation for performing uncertainty quantification techniques. The sets of training data, validation data, and test data each consist of 67 independent runs, resulting in 10,117 data points each. Moreover, we use additional sets of the same size for calibration and model selection. Though more data is available for training the model, increasing the number of training points would make exact inference computationally infeasible and require approximation methods that introduce further uncertainty. An exception to this is the section about computational uncertainty, where we use a larger dataset for training the model and approximation algorithms for inference.

Before training the Gaussian process model, we need to select a prior mean and kernel. For simplicity, we choose a mean of 0. This choice is reasonable since a constant term in the output is removed through standardization, and the prior mean plays no significant role for predictions in regions where training data are available [6].

For the kernel selection, several commonly used kernels are compared. We consider the squared exponential (SE), rational quadratic (RQ), and Matérn kernels. Due to their universal property, they can be used to approximate any function given enough training data [101]. All of these kernels have lengthscale parameters that determine the scale over which the output value varies for changes in the input values and a global factor for adjusting the overall variance, additional to other parameters specific to the kernel. For the Matérn kernel, we restrict the parameter ν to $\nu = \frac{1}{2}$, $\nu = \frac{3}{2}$ and $\nu = \frac{5}{2}$, leading to once, twice and three times differentiable predicted functions, respectively [4]. The predictions for both the SE and RQ kernel are infinitely differentiable [4]. Moreover, for each kernel, we test the isotropic variant with one global lengthscale as well as the anisotropic variant with one separate lengthscale per input dimension [4].

In order to determine which kernel is the best choice, we train three Gaussian process models on three independent datasets, each of size 10,117. We then compare their performance on the validation set in terms of RMSE. Throughout all experiments, the Gaussian process implementation provided by GPflow (Version 2.8.1) [102] is used, together with the L-BFGS-B optimizer by scipy (Version 1.7.3) [103] for hyperparameter optimization, unless stated otherwise.

The anisotropic Matérn Kernel with $\nu = \frac{3}{2}$, in the following called Matérn$\frac{3}{2}$ kernel, performs best for all three training sets. Therefore, we adopt this kernel for all subsequent experiments. It is defined by

$$k(\mathbf{x}_*, \mathbf{x}) = \sigma^2 \left(1 + \sqrt{3} \left(\sum_{m=1}^{D} \frac{(x_{*,m} - x_m)^2}{l_m^2} \right)\right) \exp\left(-\sqrt{3} \left(\sum_{m=1}^{D} \frac{(x_{*,m} - x_m)^2}{l_m^2} \right)\right), \tag{3.58}$$

where σ is the global variance parameter and l_m for $m = 1, \ldots, D$ are the lengthscales for each dimension [4]. The stability of the optimized hyperparameters across the three training datasets further supports our choice.

3.6 Sensitivity analysis – experimental setup and results

In this section, we apply variance-based sensitivity analysis as discussed in Subsection 3.1.2 to the Gaussian process model. We first perform an analysis assuming independent input variables and compare the results to another variant, where monotonic dependencies between the input variables are considered.

3.6.1 Testing method assumptions

In order to apply variance-based sensitivity analysis, two assumptions about the model and the data have to be tested. The first assumption is that the function $Y = f(X_1, \ldots, X_D)$ is square integrable over the D-dimensional unit hypercube Ω^D [37]. If f does not have this property, the expected value or variance might not exist. However, numerical estimates of the integrals are still finite in this case and can give plausible but wrong results [40]. For Gaussian Processes, f is the posterior mean given by

$$f(x_*) = K(\mathbf{x}_*, X)[K(X, X) + \sigma_n^2 I]^{-1}(\mathbf{y} - m(X)) \tag{3.59}$$

for a fixed set of training data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{y} = [y_1, \ldots, y_n]^\top$, kernel $k(.,.)$ and a zero-mean prior, evaluated at a test point $\mathbf{x}_* \in \mathbb{R}^D$. Since only the first part depends on $\mathbf{x}_*$, the equation can be rewritten as

$$\begin{aligned} f(x_*) &= K(\mathbf{x}_*, X)\mathbf{a} \\ &= \sum_{i=1}^{n} a_i k(\mathbf{x}_*, \mathbf{x}_i) \end{aligned}$$

where $\mathbf{a} = [K(X,X) + \sigma_n^2 I]^{-1}(\mathbf{y} - m(X))$ is some constant depending on the training data. Therefore, the integral of $|f|^2$ can be written as

$$\begin{aligned}
\int_{\Omega^D} |f(\mathbf{x}_*)|^2 d\mathbf{x}_* &= \int_{\Omega^D} (\sum_{i=1}^{n} a_i k(\mathbf{x}_*, \mathbf{x}_i))^2 d\mathbf{x}_* \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \int_{\Omega^D} k(\mathbf{x}_*, \mathbf{x}_i) k(\mathbf{x}_*, \mathbf{x}_j) d\mathbf{x}_* \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \frac{1}{2} \left(\int_{\Omega^D} \left(k(\mathbf{x}_*, \mathbf{x}_i) + k(\mathbf{x}_*, \mathbf{x}_j)\right)^2 d\mathbf{x}_* \right. \\
&\left. - \int_{\Omega^D} k(\mathbf{x}_*, \mathbf{x}_i)^2 d\mathbf{x}_* - \int_{\Omega^D} k(\mathbf{x}_*, \mathbf{x}_j)^2 d\mathbf{x}_* \right) .
\end{aligned}$$

If $k(.,\mathbf{x})$ is square integrable for any choice of $\mathbf{x}$ in the training set, each integral in the sum is finite, as square integrable functions are closed under addition. Hence, the sum itself is finite as well. It remains to show that the chosen kernel, the Matérn$\frac{3}{2}$ kernel defined in Equation 3.58, is square integrable. This kernel is continuous over $\mathbf{x}_*$ for any choice of $\mathbf{x}$ since it consists of a product of a polynomial function and a chained exponential and polynomial function. Its square k^2 is thus also continuous, and since continuous functions are bounded over a compact domain such as Ω^D [104], also has a finite integral. It follows that for the chosen kernel, the function of interest $f(\mathbf{x}_*)$ is square integrable over Ω^D.

The second assumption, independence of the input variables, is more difficult to verify without extensive knowledge of the application domain. In the case where the variables represent different physical properties of one object, as examined here, it is implausible that they are completely decoupled. This is also reflected in the empirical Spearman correlation coefficients of the training data shown in Figure 3.7, which indicate several significant interactions. Since the sensitivity analysis is considerably more difficult in the case of dependent variables, as discussed in Subsubsection 3.1.2, and the experiments presented here primarily serve demonstration purposes, results for independent sampling will be discussed first, followed by a simple method for the analysis of dependent variables.

3.6.2 Independent sampling

As a first step in the sensitivity analysis, distributions for the input variables must be determined. The input variables are treated as independent in this section, so it suffices to specify one marginal distribution per variable without considering any interactions. If the actual distributions are unknown, one common approach is to choose a theoretical distribution and fit its parameters to the available data [105]. In order to

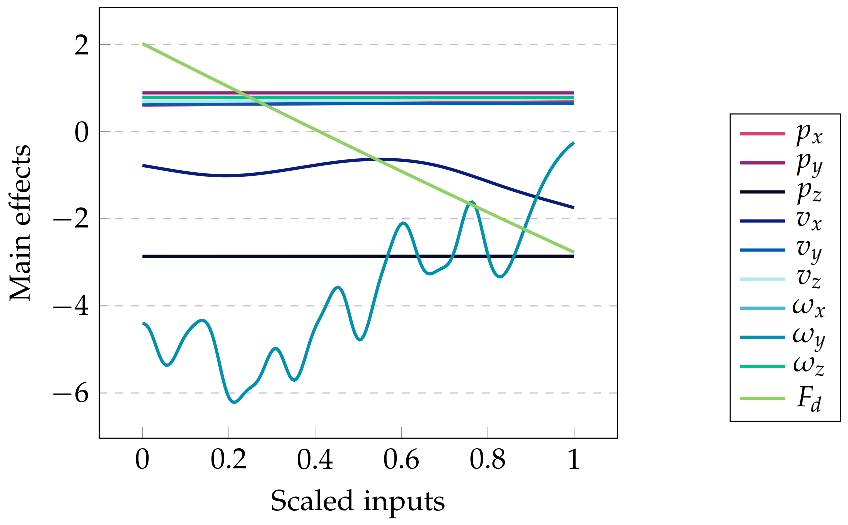
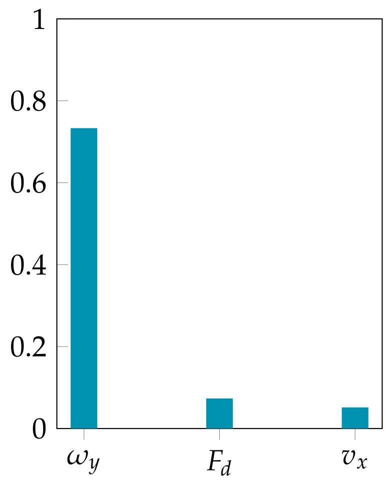


Figure 3.5: Main effects $\mathbb{E}(Y|X_i)$ for each input variable X_i. The values represent 200 points over the full range of each input and are rescaled to the interval $[0,1]$ for comparability. Estimates for the expected values are calculated using Monte Carlo integration with a Sobol' sequence of size 2^{12} for each point.

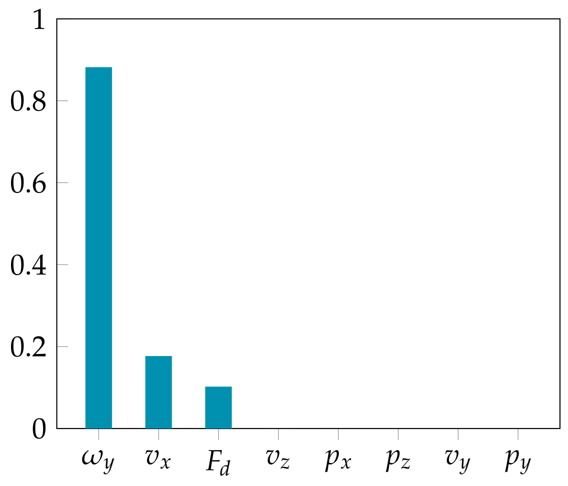
find optimal distributions, different distributions were fitted to each variable on the test dataset and compared using the Kolmogorow-Smirnow test. However, this approach did not yield useful results as all p-values were orders of magnitude lower than any reasonable significance level. In this case, Helton and Davis [105] suggest specifying the percentile function directly without assuming some closed form for the distributions as an alternative approach. In this experiment, we use the empirical percentiles of all samples in the test set and interpolate them with cubic splines, assuming the percentile function is differentiable. This produces samples close to the training data distribution but is also likely to overfit. Therefore, a more robust approach would be to consult experts to specify the percentiles based on additional knowledge, as advised by Helton and Davis [105].

In the following analysis, the sensitivity indices are calculated using the implementation provided by SALib (Version 1.4.7) [106, 107], adapted to transform the uniform samples to any probability distribution using the method described in Subsubsection 3.1.2. The integrals are estimated using a Sobol' sequence with 2^{12} samples, the number limited by the available RAM. In order to construct 95% confidence intervals, we use the bootstrapping technique described in [108] with 2000 resamples per sensitivity index.

A useful tool for a preliminary, visual analysis is plotting the main effects $\mathbb{E}(Y|X_i)$ for each input variable X_i [37, 45]. The main effect is the measure used in Sobol′ indices before taking the variance over the expected values, resulting in a function of a single feature X_i. Therefore, they give a more detailed picture of how the model output responds to the different values for each input variable. As can be seen in Figure 3.5, v_x, ω_y, and F_d appear to have the most significant influence on the traction force when the effects of other variables have been averaged out by integration. However, since these plots only show first order effects, the other variables could still significantly influence the model output through their interactions.



(a) First-order Sobol′ indices



(b) Total Sobol′ indices

Figure 3.6: Sobol′ indices for independently sampled data, including only significant indices where 0 was not inside the 95% confidence interval.

The results for the first order and total Sobol′ indices are presented in Figure 3.6. Table 3.2 additionally contains the values for all total indices since they span too many orders of magnitude to be well represented in the plot. About 86% of the total variance of the traction force is explained by the first order effects of ω_y, F_d and v_x. The other variables do not show any significant first order effect. These results are consistent with the main effects in Figure 3.5, where all lines except those for ω_y, F_d, and v_x appear to be constant. The only significant second order effect arises from the interaction between ω_y and v_x, accounting for another 11.77% of the total variance. Hence, close to 98% of the total variance can be attributed to the first and second order effects of the three most important variables. While the first order effect dominates the contributions from

Table 3.2: Total Sobol′ indices with their 95% confidence intervals for all input variables.

Variable	Total Sobol′ Index	95% CI half width
ω_1	8.8198×10^{-1}	$\pm 3.6124 \times 10^{-2}$
v_x	1.7694×10^{-1}	$\pm 1.6755 \times 10^{-2}$
F_d	1.0225×10^{-1}	$\pm 6.5163 \times 10^{-3}$
v_z	8.3371×10^{-5}	$\pm 1.8020 \times 10^{-5}$
p_x	1.5179×10^{-5}	$\pm 1.0138 \times 10^{-6}$
p_z	6.5795×10^{-8}	$\pm 9.9538 \times 10^{-9}$
v_y	1.7464×10^{-8}	$\pm 1.8678 \times 10^{-9}$
p_y	6.2265×10^{-12}	$\pm 8.7946 \times 10^{-13}$
ω_0	0.0000	± 0.0000
ω_2	0.0000	± 0.0000

ω_y and F_d, v_x is more influential through its second and higher-order interactions.

The remaining variables only contribute through higher-order interactions. ω_x and ω_z have total indices of zero and thus have no effect on the traction force. Thus, they can be excluded from the model without affecting the quality of predictions. Since the total effects of the remaining variables are very close to zero as well, they can be considered for exclusion as well.

To determine the variables to keep in the model, we train different versions with varying numbers of variables according to their importance indicated by the total Sobol′ indices and compare their RMSE on the validation set. Only including the three most important variables increases the RMSE from 1.7855 to 2.2010. However, further including v_z and p_x with total effects in the order of magnitude of 10^{-5} slightly improves accuracy compared to the full model, resulting in an RMSE of 1.7602. Including more features beyond these leads to worse accuracy again. Therefore, for the following experiments only ω_y, F_d, v_x, v_z, and p_x are used as input variables. The sensitivity analysis thus enables us to cut the number of input variables in half.

3.6.3 Correlated sampling

In order to examine the effect of dependencies in the input variables, a suitable probability distribution has to be chosen so that their relationship is reflected in the samples. However, such a joint distribution quickly becomes complex and is difficult to model if the input variables are not distributed close to some theoretical distribution, such as a Gaussian distribution. Furthermore, most sampling schemes like low discrepancy sequences or Latin hypercube sampling only generate independent variables, so their convergence benefits in the Monte Carlo estimates cannot be utilized for distributions with arbitrary dependence structures. Therefore, in the following we use the Iman-Conover transformation [109]. By only re-pairing the variables but not changing their values, the method introduces correlation into a sample without interfering with the properties of stratified sampling schemes like Latin hypercube sampling. Moreover, it makes no assumptions about the marginal distributions for the variables and does not alter them in the process. The limitation of the approach is that it can only introduce dependencies in the form of Spearman rank correlations, which correspond to monotonic relationships between variables [109].

Formally, given a $N \times D$ matrix of N D-dimensional samples, the goal is to reorder the values in each column so that the resulting Spearman rank correlation matrix is as close as possible to the desired matrix C. The theoretical basis of the method is the Cholesky transformation, which can be used to transform an independent random row vector X with correlation matrix I into a random vector X' with any correlation matrix C. Because C is positive definite and symmetric, it can be written as the product of a lower triangular matrix P and its transpose $C = PP^\top$. Using this decomposition, X' is given by $X' = XP^\top$ [109]. Because the matrix multiplication alters the values in X, the method cannot be applied directly to the sample without changing the above-mentioned properties. Therefore, the Cholesky transformation is performed on a score matrix R first, and the correlation is subsequently transferred to the sample through reordering. The columns of R are formed by distinct random permutations of the van der Waerden scores $\{\Phi^{-1}(\frac{i}{N+1})\}$, $i = 1, \dots, N$, where Φ is the cumulative standard normal distribution function. Since R is generated randomly and may have a different rank correlation than I, the inverse direction of the Cholesky transformation $RQ^{-\top}$ is used as a variance-reduction step first. $QQ^\top$ is the Cholesky decomposition of the Spearman rank correlation matrix of R. In the next step, R is multiplied by $P^\top$, where $C = PP^\top$ is the Cholesky decomposition of C. Hence, the resulting matrix $R^* = RQ^{-\top}P^\top$ has a rank correlation close to C. In order to transfer this correlation structure to the sample, the columns of the sample are arranged so that they have the same order as the corresponding column in R [109] in the final step.

The same empirical percentile functions as in the independent analysis are used for the marginal probability distributions. Correlations are then imposed on samples from these distributions using the Spearman rank correlation matrix of the training data. In order to make the group-based sensitivity analysis feasible, features with a correlation ≤ 0.4 are considered sufficiently small to treat the corresponding features as independent. Figure 3.7 illustrates the resulting correlation structure. The connected components in the graph give rise to six independent groups, to which the group-based sensitivity analysis, as described in Subsubsection 3.1.2, can be applied. The sensitivity indices presented in Figure 3.8 are obtained the same way as in Subsection 3.6.2, except

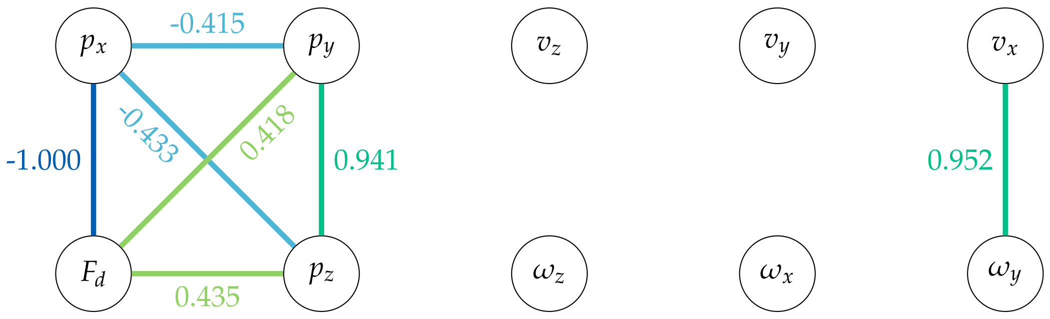


Figure 3.7: Spearman rank correlation structure in the training data, including only values with a p-value below 0.05 and absolute value greater than 0.4.

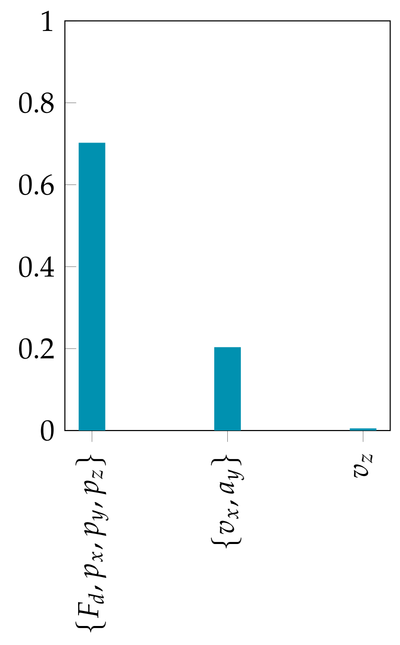
Table 3.3: Total Sobol' indices with their 95% confidence intervals for all independent groups of input variables.

Group	Total Sobol' Index	95% CI half width
$\{F_d, p_x, p_y, p_z\}$	7.7039×10^{-1}	$\pm 1.6330 \times 10^{-2}$
$\{v_x, a_y\}$	2.8271×10^{-1}	$\pm 8.3220 \times 10^{-3}$
v_z	6.0914×10^{-2}	$\pm 3.9104 \times 10^{-3}$
v_y	6.6842×10^{-4}	$\pm 4.7168 \times 10^{-5}$
ω_z	9.1967×10^{-5}	$\pm 9.1537 \times 10^{-6}$
ω_x	5.9665×10^{-5}	$\pm 5.2849 \times 10^{-6}$

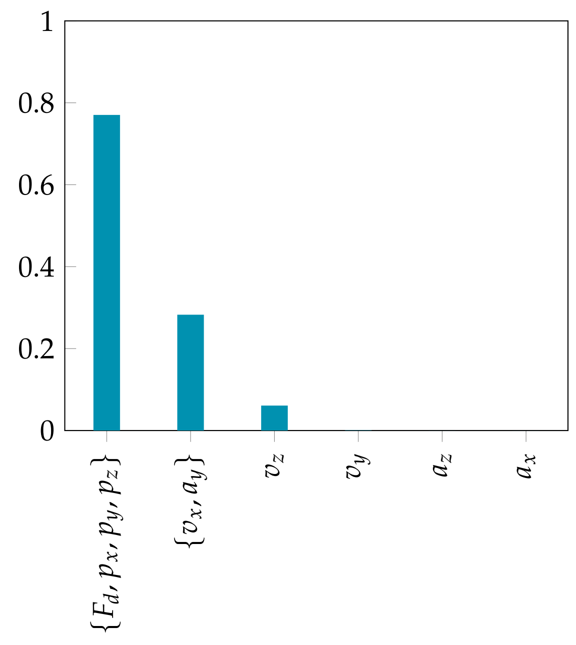
for the sampling scheme. Because reordering the columns would destroy the structure of a Sobol' sequence but not that of a Latin hypercube sample, the latter is used instead. Furthermore, the reduced number of variables allows a greater sample size of 2^{14}.

Figure 3.8 shows the first and total Sobol' indices obtained from the analysis with dependent input variables. More details about the total indices are provided in Table 3.3. The results are roughly consistent with those in the independent case with some deviations. F_d, ω_y, and v_x remain the most important variables, though their contributions cannot be separated from those of the three position variables p_x, p_y and p_z anymore due to the grouping. Their first order effects and interactions within groups account for about 91% of the total variance. In contrast to the analysis for independent inputs, F_d together with the positions takes the first place for both first order and total indices and significantly surpasses the effect of v_x and ω_y.

While the order of the remaining variables remains the same compared to the independent analysis, their total effects are all nonzero and make up a larger part of



(a) First-order Sobol' indices



(b) Total Sobol' indices

Figure 3.8: Sobol' indices for correlated and grouped variables, including only significant indices where 0 was not inside the 95% confidence interval.

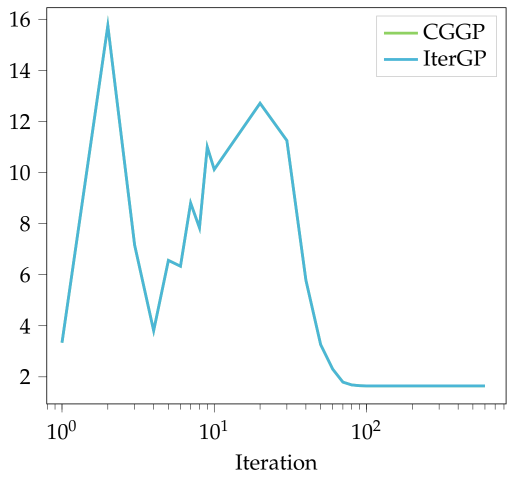
the overall variance, with no total index falling below 10^{-5}. Most notably, v_z has a nonzero first order index and the only significant second order index of 0.0164 for the interaction with v_x and ω_y. Its relatively high total index of about 0.0609 is two orders of magnitude larger compared to all remaining variables. Therefore, these results can better explain the benefit of including v_z in the model. However, the method does not offer specific insights into the contribution of p_x, since its effect cannot be separated from the group.

3.7 Computational uncertainty – experimental setup and results

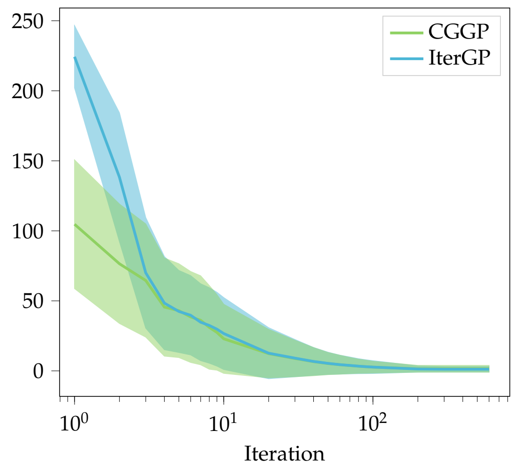
In this section, we apply the method for quantifying the computational uncertainty arising from the matrix inverse approximation discussed in Section 3.2 and compare the results to a similar state-of-the-art method, which does not take computational uncertainty into account. Both methods are based on conjugate gradients and result in the same prediction for the mean. They only differ in their predictive variance. For the first method, in the following called IterGP, we use the implementation provided by the IterGP library (Version 0.1.dev42) [74] with the conjugate gradients approximation method. It calculates the computational uncertainty according to Section 3.2 without considerable additional computational cost. The second method, in the following called CGGP, is for example used in the GPyTorch library [65]. It estimates the posterior variance with the Lanczos Variance Estimate technique [110], which uses information from the mean approximation to speed up the additional required linear solves. The implementation was kindly provided by Jonathan Wenger [111].

The Gaussian process models for both IterGP and CGGP are trained on an extended set of training data with 20,234 data points in order to more accurately represent a scenario in which approximation methods are necessary. To ensure comparability, the same kernel parameters optimized on the original training data with 10,117 instances and an exact Gaussian process model are used for both methods. Moreover, the isotropic variant of the Matérn$\frac{3}{2}$ kernel is used because IterGP does not support separate lengthscales for each dimension.

The generalization error in terms of RMSE across iterations is shown in Figure 3.9a. As expected, it is identical for both methods everywhere due to their equivalent computations for the mean approximation. After only 100 iterations, the RMSE of 1.6459 is comparable to the RMSE of 1.6264 of the model using the full conventional matrix inversion and trained on half the data points (see Table 3.4, first column). Figure 3.9b compares the mean values of the predicted variance over the test set. Additionally, the standard deviation is depicted as shaded area in order to better represent the distribution of variances at different points. While IterGP predicts a significantly higher variance in the first few iterations, both its mean and standard deviation quickly become similar between the two methods and converge to the same value. This can also be observed in Figure 3.10, where the respective confidence intervals are depicted after 3, 10, and 100 iterations. These plots show that the variances become increasingly similar not only in their mean but also locally at every prediction in the displayed portion of the test set.



(a) RMSE



(b) Mean and standard deviation of the predicted variance

Figure 3.9: Comparison of CG approximation methods. CGGP is the method currently used by GPytorch and does not estimate computational uncertainty. IterGP is the equivalent instance from the IterGP package, explicitly modeling computational uncertainty.

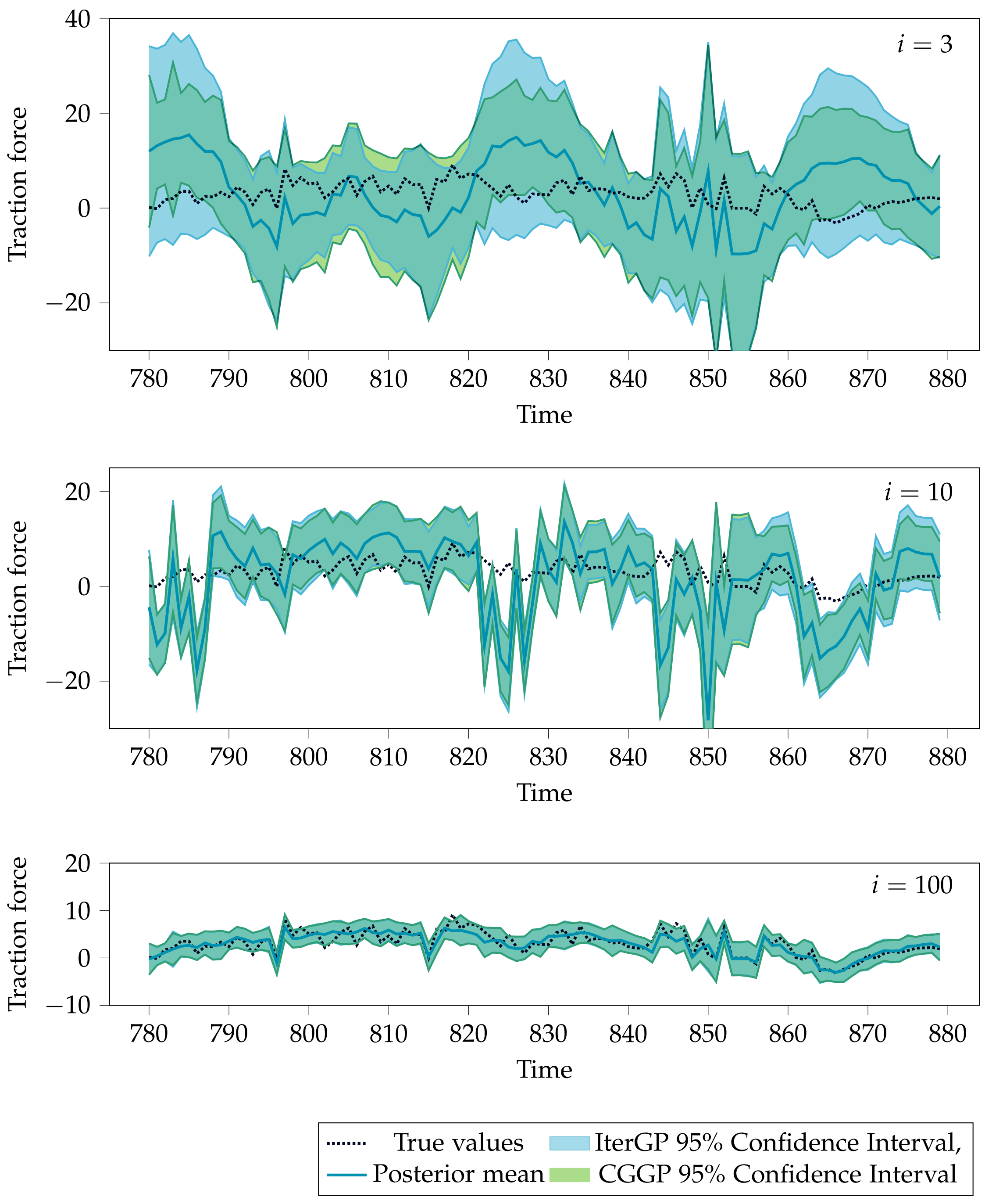


Figure 3.10: Comparison of the predictions for IterGP, which explicitly models computational uncertainty, and CGGP, which only approximates mathematical uncertainty. Both methods are based on the conjugate gradient method and always result in the same mean.

3.8 Calibration – experimental setup and results

In this section, we apply three different calibration methods, one for each definition of calibration, and compare the results. Isotonic calibration (Subsection 3.3.3) is used for quantile calibration, beta calibration (Subsubsection 3.3.4) for distribution calibration, and normal calibration (Subsubsection 3.3.4) for variance calibration. We only examine post-hoc methods since they offer more flexibility than other methods and do not require any modification of the Gaussian process model itself.

The following results use the implementation of calibration methods provided by the net:cal package (Version 1.3.5) [83]. The training, validation, and test sets are the same as in previous experiments. Additionally, another set of calibration data of the same size as the training set is used for calibration. The kernel is a Matérn$\frac{3}{2}$ kernel with separate lengthscales for each input dimension. For both of the Gaussian process-based methods, we use 1024 inducing points, a batch size of 256 and 256 Monte Carlo samples per batch for computing the objective function and gradient. The parameters are optimized using ADAM with a learning rate of 0.001.

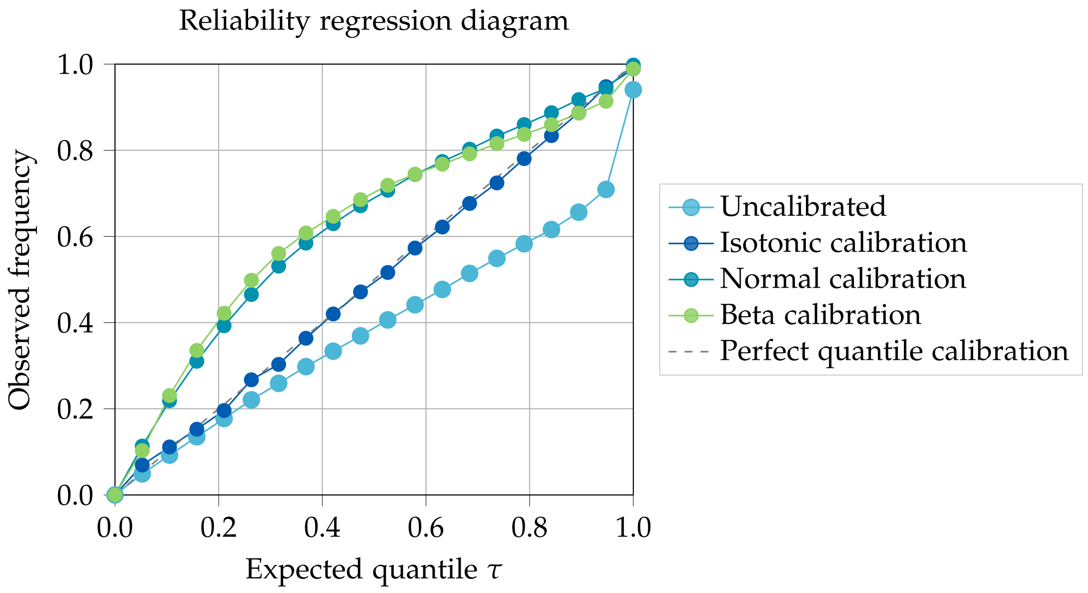


Figure 3.11: The reliability diagram illustrates the proportion of actual samples in the test set covered by the predicted quantile for a specific value of τ. If the model is perfectly quantile calibrated, all predicted quantiles match the empirical quantiles, resulting in a straight line.

The reliability plot in Figure 3.11 compares the predicted quantiles with the empirical

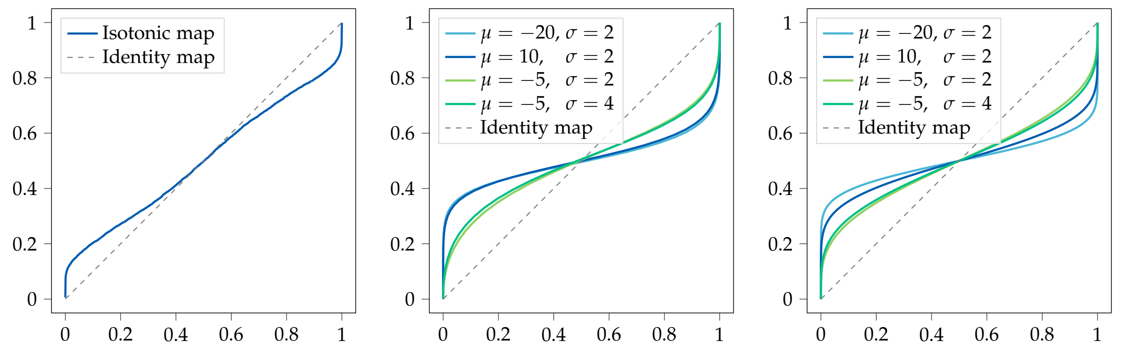
Table 3.4: Comparison of error metrics for different calibration methods. The metrics measure accuracy, quantile calibration, distribution calibration, variance calibration, and sharpness, respectively. The best values are shown in bold.

	Uncalibrated	**Isotonic calibration**	**Beta calibration**	**Normal calibration**
RMSE	**1.6264**	1.6269	1.6272	**1.6264**
$\overline{\textbf{PBL}}$	0.3059	**0.2812**	0.3232	0.3260
NLL	4.1017	**1.5645**	1.8423	1.6738
ENCE	0.6479	**0.6360**	0.6544	0.7010
MPIW	**0.0766**	0.0840	0.2579	0.2207

quantiles of the traction force in the test set. If a model is perfectly quantile calibrated, these quantiles match exactly. The line corresponding to the uncalibrated model is consistently below the diagonal. Hence, the model is overconfident in its predictions on average over the whole test data. This observation is consistent with the plot of the predictive distribution on a portion of the test data in Figure 3.13a, where the confidence bounds are too narrow to reflect the model's uncertainty almost everywhere. For example, the true values of the traction force lie outside the 98% in significantly more than 2% of the cases. While the isotonic regression is able to almost perfectly calibrate the quantiles, as can be seen by the line close to the diagonal in Figure 3.11, the other two calibration methods lead to a worse quantile calibration by making the model underconfident instead of overconfident. This is also reflected in the $\overline{PBL}$ values of each method in Table 3.4. While isotonic calibration improves the quantile calibration, the other two methods lead to worse quantile calibration. The same holds for variance calibration in terms of ENCE. All three methods significantly improve the distribution calibration measured by the NLL. However, isotonic calibration still outperforms the other two.

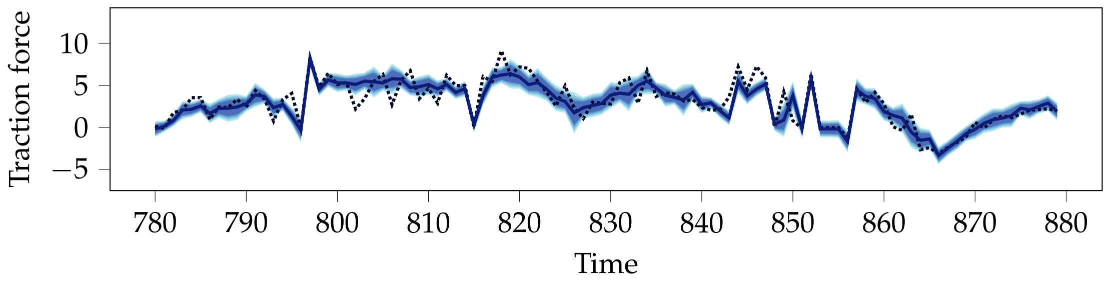
The accuracy of the model is not significantly different after calibration for any method, as the RMSE remains nearly constant. While this is always the case for normal calibration, which only scales the variance but does not affect the mean, the other two methods could lead to worse accuracy. In this case, we observe no trade-off between calibration and accuracy.

Furthermore, isotonic calibration only decreases sharpness in terms of MPIW by a small amount. In contrast, for beta calibration and normal calibration, the mean width of the 95% confidence interval becomes almost three times as large compared to the uncalibrated case. An increase in the confidence interval width is generally necessary for calibration in this case since the uncalibrated model is consistently overconfident. It should also be noted that the value of this measure largely depends on the confidence interval under consideration. As can be seen in Figure 3.13, isotonic calibration places more probability mass in the tails while keeping the center narrow, whereas normal and beta calibration widen the center region but have shorter tails. Therefore, if for instance, the width of the 98% confidence interval is considered instead, isotonic calibration may even lead to less sharp predictions than the other two methods according to the MPIW.

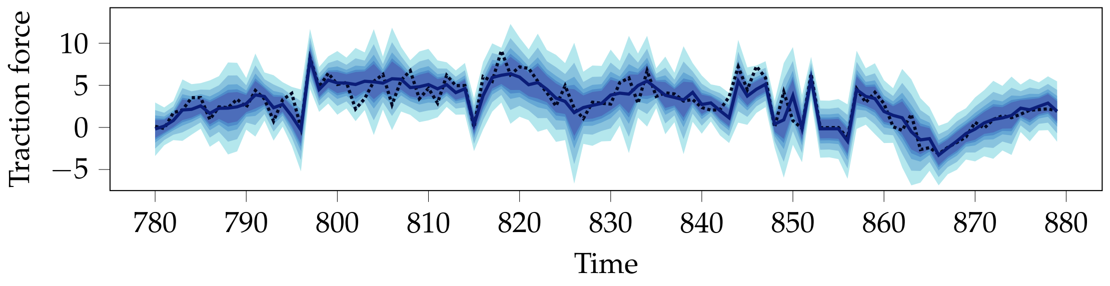


(a) Isotonic calibration (b) Beta calibration (c) Normal calibration

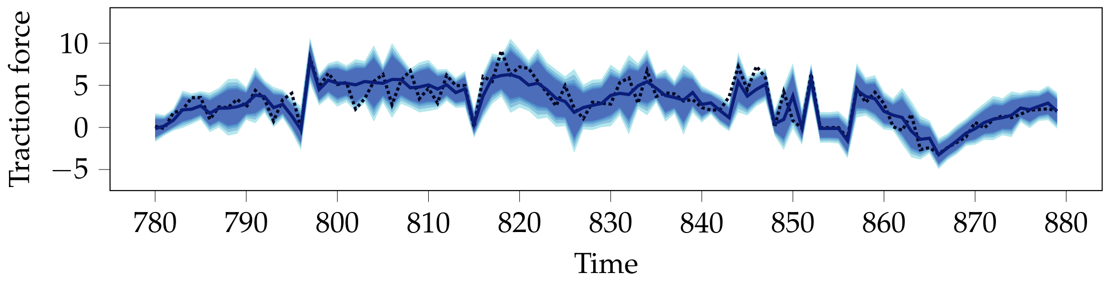
Figure 3.12: Calibration maps for the cumulative distribution of different calibration methods. For isotonic calibration, the map is globally the same. For beta and normal calibration, the maps are different for each output. Therefore, three samples are displayed.



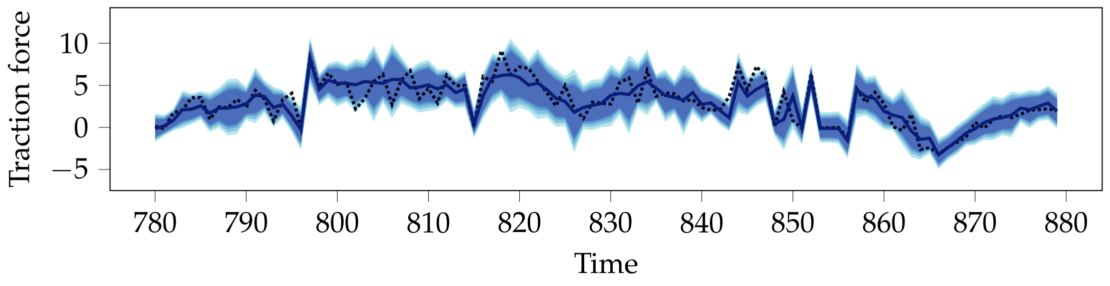
(a) Uncalibrated



(b) Isotonic calibration



(c) Beta calibration



(d) Normal calibration

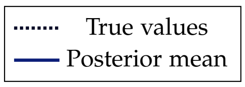


Figure 3.13: Comparison of the calibrated probability distributions for a part of the test data. The shaded regions show the predicted 98%, 95%, 90%, 85%, and 80% confidence intervals, respectively.

3.9 Discussion

Sensitivity analysis

The results from the analysis of independent variables and the analysis including correlations are consistent with respect to the identification of the most important input variables. These findings are confirmed by the changes in model accuracy for different subsets of inputs and reflect the physical properties of the modeled wheel. The downhill force F_d directly influences the traction force while the velocity v_x and angular velocity ω_y in the driving direction determine how the wheel interacts with the soil.

However, there are some significant differences in the relative contributions of each feature or group between the analysis of independent and correlated variables. Therefore, the independent case provides enough information for the selection of features but does not allow accurate insights into the sensitivity of the traction force with respect to the individual input variables. Furthermore, it cannot be ruled out that the rank correlation structure under consideration is not sufficient to represent the true dependencies between input variables. In that case, a more sophisticated model for the joint distribution of input variables would be necessary. However, this comes at the expense of requiring larger sample sizes for the involved Monte Carlo integration, as common sampling schemes to speed up the convergence can only be used to produce independent samples for each variable. Increasing the number of Monte Carlo samples could further improve the accuracy of the results.

Moreover, the sensitivity analysis only considers variations in the predictive mean but not in the whole predictive distribution. In order to include this information, adaptions such as the one discussed in Subsubsection 3.1.2 could be used.

Lastly, we only examine the model's sensitivity with respect to the input features. However, the same methods could also be used to investigate the impact of different kernel parameters on the model output. Such an analysis would be considerably more complicated due to the necessity of specifying a distribution for each parameter. The computational cost would be substantially higher as well since each model evaluation in the Monte Carlo integration requires a matrix inversion. The sensitivity analysis of input variables allows to reuse the same model for all evaluations, requiring only one matrix inversion in total.

Computational uncertainty

The results comparing IterGP to CGGP show that modeling computational uncertainty only makes a difference for the first few iterations in this scenario. The predicted variances of IterGP and CGGP produce increasingly similar predictions even before both converge to the exact solution. As expected, the variance predicted by IterGP decreases for an increasing number of iterations as the part making up the computational uncertainty decreases. Interestingly, the variance predicted by CGGP exhibits the same behavior without explicitly taking into account the computational uncertainty. Thus, especially for a larger number of iterations, the choice of the method does not significantly affect the result. However, IterGP is able to calculate the variance without a significant computational overhead, making it more efficient. Moreover, the method guarantees to accurately represent the uncertainty due to approximations, whereas CGGP has no such theoretical interpretation.

Calibration

Comparing all tested calibration methods, isotonic calibration performs better than beta and normal calibration for all calibration metrics. This result is unexpected since the method only targets the weakest calibration definition and is, unlike the other two methods, restricted to one global calibration map. One possible explanation is that the Gaussian process based methods perform worse for large amounts of training data, as also observed by Song et al. [85] in their experiments. They suggest that using a larger number of inducing points may help improve the results in such a scenario [85]. However, changing the number of inducing points from 64 to 256 and 1024 did not change the results significantly. Therefore, it is unlikely the cause of the worse performance of these methods.

Another explanation of the better performance of isotonic calibration is the greater flexibility of its calibration map. While beta and normal calibration are bound to certain parametric functions, isotonic regression can approximate arbitrary functions. Figure 3.12 shows a comparison of the calibration maps learned by the different methods. The shape of the isotonic calibration map, deviating from the identity map at the edges but being close to it in the middle section, is impossible to represent with a function from the beta family or simple scaling of the variance. For instance, functions from the beta family that show the same deviations in regions close to zero or one will always have a significantly lower slope in the middle section than the identity function. As a result, both beta and normal calibration broaden the distribution everywhere, whereas isotonic calibration only widens the tails, and is consequently better able to capture the actual uncertainty of the model. This can also be observed in Figure 3.13, where the transformed distributions for normal and beta calibration have a wider 80% but narrower 98% confidence interval than the one for isotonic calibration. Interestingly, the results for beta calibration and normal calibration are very similar, evident from both the metrics in Table 3.4 and the transformed distributions in Figure 3.13. The calibration maps in Figure 3.12 look very similar as well for every combination of mean and standard deviation. This shows that the increased flexibility of the beta map does not lead to an advantage in this scenario, and this model is not better suited to capture the true distribution of target values compared to simply scaling the variance.

The global nature of isotonic calibration does not appear to be a problem in this case as the local distributions of values for the traction force can be approximated well by the marginal distribution. For instance, since the uncalibrated model seems to be underconfident almost everywhere, the calibration map can improve calibration locally everywhere by increasing the variance. This would not be possible if the model was also underconfident in a significant portion of predictions. Therefore, for a scenario such as the one examined here, isotonic regression is the best choice for all calibration objectives. These findings are consistent with the results of Küppers et al. [83] on object detection tasks.

4 Conclusions

In this work, we explored three different techniques for uncertainty quantification in the context of Gaussian processes. In order to evaluate how these methods can improve the understanding of predictive uncertainty in a real-world problem, we applied them to a Gaussian process model trained to predict the traction force on the wheel of a planetary rover.

First, we used variance-based sensitivity analysis to assess how different input variables contribute to the output and the associated uncertainty. We computed Sobol′ indices both under the assumption that the input variables are independent and including dependencies in the form of rank correlations. While the latter method represents the structure of the input variables more accurately, the analysis is less specific as it compares groups of variables instead of all individual variables. Both methods identified the same variables as the most influential, enabling us to cut the number of input variables in half without losing accuracy. However, the relative importance of these variables differed for the two approaches. Thus, if the relative importance is of interest, for example, to prioritize the quality of sensors installed on the wheel for measuring the input quantities, assuming the variables are independent is not sufficient in this case. Moreover, more detailed modeling of the joint probability distribution could be beneficial.

We also compared two approximate methods for calculating the matrix inverse needed for Gaussian process inference and their uncertainty estimates. Both methods are based on conjugate gradients and result in the same approximation for the posterior mean but differ in their variance estimate. The first, recently developed method explicitly models computational uncertainty. In contrast, the second method uses the current state-of-the-art Lanczos variance estimate, treating the approximate mean as the exact solution. While there were no significant differences in the predicted variances after the first few iterations, the method including computational uncertainty offers the advantages of being more computationally efficient and providing a theoretical interpretation of the predicted uncertainty.

Finally, we examined the quality of the predicted uncertainty estimates of the Gaussian process model. We found it to be consistently overconfident and compared three different post-hoc methods to improve calibration. The simplest method, isotonic regression, performed best for all our calibration metrics. This is likely due to the high flexibility of its calibration map and the model's consistent overconfidence, which can be effectively addressed with a single global calibration map to achieve good local results.

These results provide insights into different areas of uncertainty. In addition to the three techniques employed for uncertainty quantification in this study, further areas remain for future research. In the problem at hand, including the uncertainty about input values directly in the Gaussian process model would represent reality more accurately since they come from possibly noisy sensor measurements. Moreover, including the uncertainty about the choice of kernel and hyperparameters could make the analysis more robust to model misspecification. In order to obtain theoretical guarantees for uncertainty bounds, the predictive uncertainty estimates from the Bayesian variance prediction the Gaussian process delivers could be extended to a frequentist framework. By incorporating these elements into future work, we can obtain a more complete picture of the uncertainties associated with our model.

List of Figures

List of Tables

Bibliography

[1] R. E. Arvidson, J. F. Bell III, P. Bellutta, N. A. Cabrol, J. G. Catalano, J. Cohen, L. S. Crumpler, D. J. Des Marais, T. A. Estlin, W. H. Farrand, R. Gellert, J. A. Grant, R. N. Greenberger, E. A. Guinness, K. E. Herkenhoff, J. A. Herman, K. D. Iagnemma, J. R. Johnson, G. Klingelhöfer, R. Li, K. A. Lichtenberg, S. A. Maxwell, D. W. Ming, R. V. Morris, M. S. Rice, S. W. Ruff, A. Shaw, K. L. Siebach, P. A. de Souza, A. W. Stroupe, S. W. Squyres, R. J. Sullivan, K. P. Talley, J. A. Townsend, A. Wang, J. R. Wright, and A. S. Yen. "Spirit Mars Rover Mission: Overview and selected results from the northern Home Plate Winter Haven to the side of Scamander crater." In: *Journal of Geophysical Research: Planets* 115.E7 (2010). DOI: `https://doi.org/10.1029/2010JE003633`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2010JE003633`.

[2] R. E. Arvidson, K. D. Iagnemma, M. Maimone, A. A. Fraeman, F. Zhou, M. C. Heverly, P. Bellutta, D. Rubin, N. T. Stein, J. P. Grotzinger, and A. R. Vasavada. "Mars Science Laboratory Curiosity Rover Megaripple Crossings up to Sol 710 in Gale Crater." In: *Journal of Field Robotics* 34.3 (2017), pp. 495–518. DOI: `https://doi.org/10.1002/rob.21647`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21647`.

[3] J. Görtler, R. Kehlbeck, and O. Deussen. "A Visual Exploration of Gaussian Processes." In: *Distill* (2019). https://distill.pub/2019/visual-exploration-gaussian-processes. DOI: `10.23915/distill.00017`.

[4] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN: 0-262-18253-X.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Aug. 2006.

[6] Y. Guan. "Introduction to Gaussian Processes for Regression." English. MA thesis. 2020.

[7] V. Fediukov, F. Dietrich, and F. Buse. "Multi-fidelity machine learning modeling for wheel locomotion." In: *11th Asia-Pacific Regional Conference of the International society for terrain-vehicle systems, ISTVS 2022*. ISTVS, Sept. 2022.

[8] J. Dallas, K. Jain, Z. Dong, L. Sapronov, M. P. Cole, P. Jayakumar, and T. Ersal. "Online terrain estimation for autonomous vehicles on deformable terrains." In: *Journal of Terramechanics* 91 (2020), pp. 11–22. ISSN: 0022-4898. DOI: `https://doi.org/10.1016/j.jterra.2020.03.001`.

[9] T. Zhang, S. Peng, Y. Jia, J. Sun, H. Tian, and C. Yan. "Slip Estimation Model for Planetary Rover Using Gaussian Process Regression." In: *Applied Sciences* 12.9 (2022). ISSN: 2076-3417. DOI: `10.3390/app12094789`.

[10] A. J. R. Lopez-Arreguin and S. Montenegro. "Machine learning in planetary rovers: A survey of learning versus classical estimation methods in terramechanics for in situ exploration." In: *Journal of Terramechanics* 97 (2021), pp. 1–17. ISSN: 0022-4898. DOI: `https://doi.org/10.1016/j.jterra.2021.04.005`.

[11] E. Hüllermeier and W. Waegeman. "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods." In: *Machine Learning* 110.3 (Mar. 2021), pp. 457–506. ISSN: 1573-0565. DOI: `10.1007/s10994-021-05946-3`.

[12] V. Nemani, L. Biggio, X. Huan, Z. hu, O. Fink, A. Tran, Y. Wang, X. Du, X. Zhang, and C. Hu. *Uncertainty Quantification in Machine Learning for Engineering Design and Health Prognostics: A Tutorial*. May 2023.

[13] A. D. Kiureghian and O. Ditlevsen. "Aleatory or epistemic? Does it matter?" In: *Structural Safety* 31.2 (2009). Risk Acceptance and Risk Communication, pp. 105–112. ISSN: 0167-4730. DOI: `https://doi.org/10.1016/j.strusafe.2008.06.020`.

[14] V.-L. Nguyen, S. Destercke, and E. Hüllermeier. "Epistemic Uncertainty Sampling." In: *Discovery Science*. Ed. by P. Kralj Novak, T. Šmuc, and S. Džeroski. Cham: Springer International Publishing, 2019, pp. 72–86. ISBN: 978-3-030-33778-0.

[15] M. Tan and Q. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6105–6114.

[16] P. Goldberg, C. Williams, and C. Bishop. "Regression with input-dependent noise: A Gaussian process treatment." English. In: *Advances in Neural Information Processing Systems 10 (NIPS 1997)*. MIT Press, 1997, pp. 493–499.

[17] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. "Most Likely Heteroscedastic Gaussian Process Regression." In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. Corvalis, Oregon, USA: Association for

Computing Machinery, 2007, pp. 393–400. ISBN: 9781595937933. DOI: 10.1145/1273496.1273546.

[18] L. Muñoz-González, M. Lázaro-Gredilla, and A. Figueiras-Vidal. "Heteroscedastic Gaussian process regression using expectation propagation." In: *IEEE International Workshop on Machine Learning for Signal Processing* (Sept. 2011), pp. 1–6. DOI: 10.1109/MLSP.2011.6064576.

[19] M. Lázaro-Gredilla and M. K. Titsias. "Variational Heteroscedastic Gaussian Process Regression." In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, pp. 841–848. ISBN: 9781450306195.

[20] V. Tolvanen, P. Jylanki, and A. Vehtari. "Expectation propagation for nonstationary heteroscedastic Gaussian process regression." In: *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, Sept. 2014. DOI: 10.1109/mlsp.2014.6958906.

[21] M. Heinonen, H. Mannerström, J. Rousu, S. Kaski, and H. Lähdesmäki. *Non-Stationary Gaussian Process Regression with Hamiltonian Monte Carlo*. 2015. arXiv: 1508.04319 [stat.ML].

[22] Z. B. Patel, N. Batra, and K. Murphy. "Uncertainty Disentanglement with Non-stationary Heteroscedastic Gaussian Processes for Active Learning." In: *CoRR* abs/2210.10964 (2022). DOI: 10.48550/arXiv.2210.10964. arXiv: 2210.10964.

[23] A. Girard and R. Murray-Smith. "Gaussian Processes: Prediction at a Noisy Input and Application to Iterative Multiple-Step Ahead Forecasting of Time-Series." In: *Switching and Learning in Feedback Systems: European Summer School on Multi-Agent Control, Maynooth, Ireland, September 8-10, 2003, Revised Lectures and Selected Papers*. Ed. by R. Murray-Smith and R. Shorten. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 158–184. ISBN: 978-3-540-30560-6. DOI: 10.1007/978-3-540-30560-6_7.

[24] A. McHutchon and C. E. Rasmussen. "Gaussian Process Training with Input Noise." In: NIPS'11. Granada, Spain: Curran Associates Inc., 2011, pp. 1341–1349. ISBN: 9781618395993.

[25] C. Wang and R. M. Neal. *Gaussian Process Regression with Heteroscedastic or Non-Gaussian Residuals*. 2012. arXiv: 1212.6246 [stat.ML].

[26] P. Dallaire, C. Besse, and B. Chaib-draa. "Learning Gaussian Process Models from Uncertain Data." In: *Neural Information Processing*. Ed. by C. S. Leung, M. Lee, and J. H. Chan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 433–440. ISBN: 978-3-642-10677-4.

[27] H. H. Bajgiran, P. Batlle, H. Owhadi, M. Samir, C. Scovel, M. Shirdel, M. Stanley, and P. Tavallali. "Uncertainty quantification of the 4th kind; optimal posterior accuracy-uncertainty tradeoff with the minimum enclosing ball." In: *Journal of Computational Physics* 471 (2022), p. 111608. ISSN: 0021-9991. DOI: `https://doi.org/10.1016/j.jcp.2022.111608`.

[28] W. T. Stephenson, S. Ghosh, T. D. Nguyen, M. Yurochkin, S. Deshpande, and T. Broderick. "Measuring the robustness of Gaussian processes to kernel choice." In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by G. Camps-Valls, F. J. R. Ruiz, and I. Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, 28–30 Mar 2022, pp. 3308–3331.

[29] J. Wagberg, D. Zachariah, T. Schon, and P. Stoica. "Prediction Performance After Learning in Gaussian Process Regression." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Singh and J. Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 1264–1272.

[30] T. Beckers, J. Umlauft, and S. Hirche. "Mean Square Prediction Error of Misspecified Gaussian Process Models." In: *2018 IEEE Conference on Decision and Control (CDC)* (2018), pp. 1162–1167.

[31] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 1015–1022. ISBN: 9781605589077.

[32] S. R. Chowdhury and A. Gopalan. "On Kernelized Multi-armed Bandits." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 844–853.

[33] A. Lederer, J. Umlauft, and S. Hirche. "Uniform Error Bounds for Gaussian Process Regression with Application to Safe Control." In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[34] Y. Yang, A. Bhattacharya, and D. Pati. *Frequentist coverage and sup-norm convergence rate in Gaussian process regression*. 2017. arXiv: `1708.04753 [math.ST]`.

[35] C. Fiedler, C. W. Scherer, and S. Trimpe. "Practical and Rigorous Uncertainty Bounds for Gaussian Process Regression." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.8 (May 2021), pp. 7439–7447. DOI: `10.1609/aaai.v35i8.16912`.

[36] W. Neiswanger and A. Ramdas. “Uncertainty quantification using martingales for misspecified Gaussian processes.” In: *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*. Ed. by V. Feldman, K. Ligett, and S. Sabato. Vol. 132. Proceedings of Machine Learning Research. PMLR, 16–19 Mar 2021, pp. 963–982.

[37] A. Saltelli, R. Marco, A. Terry, C. Francesca, C. Jessica, G. Debora, S. Michaela, and T. Stefano. “Global Sensitivity Analysis: The Primer.” In: 2008.

[38] J. Helton and W. Oberkampf. “Alternative representations of epistemic uncertainty.” In: *Reliability Engineering & System Safety* 85 (July 2004), pp. 1–10. DOI: 10.1016/j.ress.2004.03.001.

[39] A. Vehtari and J. Ojanen. “A survey of Bayesian predictive methods for model assessment, selection and comparison.” In: *Statistics Surveys* 6 (2012), pp. 142–228.

[40] T. Sullivan. *Introduction to Uncertainty Quantification*. en. Vol. 63. Texts in Applied Mathematics. Cham: Springer International Publishing, 2015. ISBN: 978-3-319-23394-9 978-3-319-23395-6. DOI: 10.1007/978-3-319-23395-6.

[41] H. Christopher Frey and S. R. Patil. “Identification and Review of Sensitivity Analysis Methods.” In: *Risk Analysis* 22.3 (2002), pp. 553–578. DOI: https://doi.org/10.1111/0272-4332.00039. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0272-4332.00039.

[42] B. Iooss and P. Lemaître. “A Review on Global Sensitivity Analysis Methods.” In: *Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications*. Ed. by G. Dellino and C. Meloni. Boston, MA: Springer US, 2015, pp. 101–122. ISBN: 978-1-4899-7547-8. DOI: 10.1007/978-1-4899-7547-8_5.

[43] M. D. Morris. “Factorial Sampling Plans for Preliminary Computational Experiments.” In: *Technometrics* 33.2 (1991), pp. 161–174. ISSN: 00401706.

[44] A. Saltelli and P. Annoni. “Sensitivity Analysis.” In: *International Encyclopedia of Statistical Science*. Ed. by M. Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1298–1301. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_509.

[45] J. E. Oakley and A. O’Hagan. “Probabilistic sensitivity analysis of complex models: a Bayesian approach.” In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66.3 (2004), pp. 751–769. DOI: https://doi.org/10.1111/j.1467-9868.2004.05304.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2004.05304.x.

[46] B. Sudret. “Global sensitivity analysis using polynomial chaos expansions.” In: *Reliability Engineering & System Safety* 93.7 (2008). Bayesian Networks in Dependability, pp. 964–979. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2007.04.002.

[47] A. Marrel, B. Iooss, B. Laurent, and O. Roustant. “Calculations of Sobol indices for the Gaussian process metamodel.” In: *Reliability Engineering & System Safety* 94.3 (2009), pp. 742–751. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2008.07.008.

[48] A. Saltelli and S. Tarantola. “On the Relative Importance of Input Factors in Mathematical Models.” In: *Journal of the American Statistical Association* 97.459 (2002), pp. 702–709. DOI: 10.1198/016214502388618447. eprint: https://doi.org/10.1198/016214502388618447.

[49] J. Jacques, C. Lavergne, and N. Devictor. “Sensitivity analysis in presence of model uncertainty and correlated inputs.” In: *Reliability Engineering & System Safety* 91.10 (2006). The Fourth International Conference on Sensitivity Analysis of Model Output (SAMO 2004), pp. 1126–1134. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2005.11.047.

[50] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. “Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index.” In: *Computer Physics Communications* 181.2 (2010), pp. 259–270. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2009.09.018.

[51] R. E. Caflisch. “Monte Carlo and quasi-Monte Carlo methods.” In: *Acta Numerica* 7 (1998), pp. 1–49. DOI: 10.1017/S0962492900002804.

[52] M. D. Mckay, R. J. Beckman, and W. J. Conover. “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code.” In: *Technometrics* 42.1 (2000), pp. 55–61. ISSN: 00401706.

[53] A. Saltelli. “Making best use of model evaluations to compute sensitivity indices.” In: *Computer Physics Communications* 145.2 (2002), pp. 280–297. ISSN: 0010-4655. DOI: https://doi.org/10.1016/S0010-4655(02)00280-1.

[54] R. I. Cukier, C. M. Fortuin, K. E. Shuler, A. G. Petschek, and J. H. Schaibly. “Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. I Theory.” In: *The Journal of Chemical Physics* 59.8 (Sept. 2003), pp. 3873–3878. ISSN: 0021-9606. DOI: 10.1063/1.1680571. eprint: https://pubs.aip.org/aip/jcp/article-pdf/59/8/3873/11219577/3873_1_online.pdf.

[55] A. Saltelli, S. Tarantola, and K. P.-S. Chan. "A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output." In: *Technometrics* 41.1 (Feb. 1999), pp. 39–56. ISSN: 0040-1706. DOI: 10.2307/1270993.

[56] L. Le Gratiet, S. Marelli, and B. Sudret. "Metamodel-Based Sensitivity Analysis: Polynomial Chaos Expansions and Gaussian Processes." In: *Handbook of Uncertainty Quantification*. Ed. by R. Ghanem, D. Higdon, and H. Owhadi. Cham: Springer International Publishing, 2017, pp. 1289–1325. ISBN: 978-3-319-12385-1. DOI: 10.1007/978-3-319-12385-1_38.

[57] G. Camps-Valls, J. Verrelst, J. Munoz-Mari, V. Laparra, F. Mateo-Jimenez, and J. Gomez-Dans. "A Survey on Gaussian Processes for Earth-Observation Data Analysis: A Comprehensive Investigation." In: *IEEE Geoscience and Remote Sensing Magazine* 4.2 (2016), pp. 58–78. DOI: 10.1109/MGRS.2015.2510084.

[58] R. Urtasun and T. Darrell. "Sparse probabilistic regression for activity-independent human pose inference." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587360.

[59] E. Snelson and Z. Ghahramani. "Sparse Gaussian Processes Using Pseudo-Inputs." In: *Proceedings of the 18th International Conference on Neural Information Processing Systems*. NIPS'05. Vancouver, British Columbia, Canada: MIT Press, 2005, pp. 1257–1264.

[60] M. Seeger, C. Williams, and N. Lawrence. "Fast Forward Selection to Speed Up Sparse Gaussian Process Regression." English. In: *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*. 2003.

[61] A. J. Smola and P. Bartlett. "Sparse Greedy Gaussian Process Regression." In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS'00. Denver, CO: MIT Press, 2000, pp. 598–604.

[62] J. Quiñonero-Candela and C. E. Rasmussen. "A Unifying View of Sparse Approximate Gaussian Process Regression." In: *Journal of Machine Learning Research* 6.65 (2005), pp. 1939–1959.

[63] J. Wenger, G. Pleiss, P. Hennig, J. Cunningham, and J. Gardner. "Preconditioning for Scalable Gaussian Process Hyperparameter Optimization." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 23751–23780.

[64] A. Artemev, D. R. Burt, and M. van der Wilk. "Tighter Bounds on the Log Marginal Likelihood of Gaussian Process Regression Using Conjugate Gradients." In: *International Conference on Machine Learning*. 2021.

[65] J. Gardner, G. Pleiss, D. Bindel, K. Weinberger, and A. Wilson. "Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration." English (US). In: *Advances in Neural Information Processing Systems* 2018-December (2018). Funding Information: JRG and AGW are supported by NSF IIS-1563887 and by Facebook Research. GP and KQW are supported in part by the III-1618134, III-1526012, IIS-1149882, IIS-1724282, and TRIPODS-1740822 grants from the National Science Foundation. In addition, they are supported by the Bill and Melinda Gates Foundation, the Office of Naval Research, and SAP America Inc. Publisher Copyright: © 2018 Curran Associates Inc.All rights reserved.; 32nd Conference on Neural Information Processing Systems, NeurIPS 2018 ; Conference date: 02-12-2018 Through 08-12-2018, pp. 7576–7586. ISSN: 1049-5258.

[66] H. Harbrecht, M. Peters, and R. Schneider. "On the low-rank approximation by the pivoted Cholesky decomposition." In: *Applied Numerical Mathematics* 62.4 (2012). Third Chilean Workshop on Numerical Analysis of Partial Differential Equations (WONAPDE 2010), pp. 428–440. ISSN: 0168-9274. DOI: `https://doi.org/10.1016/j.apnum.2011.10.001`.

[67] F. Schäfer, T. J. Sullivan, and H. Owhadi. "Compression, Inversion, and Approximate PCA of Dense Kernel Matrices at Near-Linear Computational Complexity." In: *Multiscale Modeling & Simulation* 19.2 (2021), pp. 688–730. DOI: `10.1137/19M129526X`. eprint: `https://doi.org/10.1137/19M129526X`.

[68] G. Ferrari-Trecate, C. Williams, and M. Opper. "Finite-Dimensional Approximation of Gaussian Processes." In: *Advances in Neural Information Processing Systems*. Ed. by M. Kearns, S. Solla, and D. Cohn. Vol. 11. MIT Press, 1998.

[69] H. Zhu, C. Williams, R. Rohwer, M. Morciniec, and M. Hammel. *Gaussian Regression and Optimal Finite Dimensional Linear Models*. English. WorkingPaper. 1997.

[70] Z. Yang, A. Wilson, A. Smola, and L. Song. "A la Carte – Learning Fast Kernels." In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA: PMLR, Sept. 2015, pp. 1098–1106.

[71] C.-A. Cheng and B. Boots. "Variational Inference for Gaussian Process Models with Linear Complexity." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[72] H. Salimbeni, C.-A. Cheng, B. Boots, and M. Deisenroth. "Orthogonally Decoupled Variational Gaussian Processes." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[73] A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing. "Stochastic Variational Deep Kernel Learning." In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[74] J. Wenger, G. Pleiss, M. Pförtner, P. Hennig, and J. P. Cunningham. "Posterior and Computational Uncertainty in Gaussian Processes." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 10876–10890.

[75] A. Capone, G. Pleiss, and S. Hirche. *Sharp Calibrated Gaussian Processes*. 2023. arXiv: `2302.11961 [cs.LG]`.

[76] A. Capone, A. Lederer, and S. Hirche. "Gaussian Process Uniform Error Bounds with Unknown Hyperparameters for Safety-Critical Applications." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 2609–2624.

[77] D. Levi, L. Gispan, N. Giladi, and E. Fetaya. "Evaluating and Calibrating Uncertainty Prediction in Regression Tasks." In: *Sensors* 22.15 (2022). ISSN: 1424-8220. DOI: `10.3390/s22155540`.

[78] M. P. Naeini, G. F. Cooper, and M. Hauskrecht. "Obtaining Well Calibrated Probabilities Using Bayesian Binning." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 2901–2907. ISBN: 0262511290.

[79] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. Schön. "Evaluating model calibration in classification." In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by K. Chaudhuri and M. Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 16–18 Apr 2019, pp. 3459–3467.

[80] J. Wenger, H. Kjellström, and R. Triebel). "Non-Parametric Calibration for Classification." In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 26–28 Aug 2020, pp. 178–190.

[81] D. Widmann, F. Lindsten, and D. Zachariah. "Calibration tests in multi-class classification: A unifying framework." In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[82] M. Minderer, J. Djolonga, R. Romijnders, F. Hubis, X. Zhai, N. Houlsby, D. Tran, and M. Lucic. "Revisiting the Calibration of Modern Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 15682–15694.

[83] F. Küppers, J. Schneider, and A. Haselhoff. "Parametric and Multivariate Uncertainty Calibration for Regression and Object Detection." In: *Computer Vision – ECCV 2022 Workshops*. Ed. by L. Karlinsky, T. Michaeli, and K. Nishino. Cham: Springer Nature Switzerland, 2023, pp. 426–442. ISBN: 978-3-031-25072-9.

[84] V. Kuleshov, N. Fenner, and S. Ermon. "Accurate Uncertainties for Deep Learning Using Calibrated Regression." In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2796–2804.

[85] H. Song, T. Diethe, M. Kull, and P. Flach. "Distribution Calibration for Regression." English. In: *International Conference on Machine Learning, 9-15 June 2019, Long Beach, California, USA*. Ed. by K. Chaudhuri and R. Salakhutdinov. Proceedings of Machine Learning Research. Proceedings of Machine Learning Research, May 2019, pp. 5897–5906.

[86] D. Feng, L. Rosenbaum, C. Glaeser, F. Timm, and K. Dietmayer. *Can We Trust You? On Calibration of a Probabilistic Object Detector for Autonomous Driving*. 2019. arXiv: `1909.12358 [cs.RO]`.

[87] P. Cui, W. Hu, and J. Zhu. "Calibrated Reliable Regression using Maximum Mean Discrepancy." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 17164–17175.

[88] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. "A Kernel Two-Sample Test." In: 13.null (Mar. 2012), pp. 723–773. ISSN: 1532-4435.

[89] D. Bhatt, K. Mani, D. Bansal, K. Murthy, H. Lee, and L. Paull. "f-Cal: Aleatoric uncertainty quantification for robot perception via calibrated neural regression." In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 6533–6539. DOI: `10.1109/ICRA46639.2022.9811903`.

[90] A. Niculescu-Mizil and R. Caruana. “Predicting Good Probabilities with Supervised Learning.” In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML ’05. Bonn, Germany: Association for Computing Machinery, 2005, pp. 625–632. ISBN: 1595931805. DOI: `10.1145/1102351.1102430`.

[91] J. de Leeuw, K. Hornik, and P. Mair. “Isotone Optimization in R: Pool-Adjacent-Violators Algorithm (PAVA) and Active Set Methods.” In: *Journal of Statistical Software* 32.5 (2009), pp. 1–24. DOI: `10.18637/jss.v032.i05`.

[92] M. Kull, T. S. Filho, and P. Flach. “Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers.” In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Singh and J. Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 623–631.

[93] L. Song, X. Zhang, A. Smola, A. Gretton, and B. Schölkopf. “Tailoring Density Estimation via Reproducing Kernel Moment Matching.” In: ICML ’08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 992–999. ISBN: 9781605582054. DOI: `10.1145/1390156.1390281`.

[94] J. Hensman, A. Matthews, and Z. Ghahramani. “Scalable Variational Gaussian Process Classification.” In: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Lebanon and S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA: PMLR, Sept. 2015, pp. 351–360.

[95] M. Fasiolo, S. N. Wood, M. Zaffran, R. Nedellec, and Y. Goude. “Fast Calibrated Additive Quantile Regression.” In: *Journal of the American Statistical Association* 116.535 (Mar. 2020), pp. 1402–1412. DOI: `10.1080/01621459.2020.1725521`.

[96] T. Gneiting, F. Balabdaoui, and A. E. Raftery. “Probabilistic forecasts, calibration and sharpness.” eng. In: *Journal of the Royal Statistical Society* 69.2 (2007), pp. 243–268. ISSN: 0035-9246 and 1369-7412.

[97] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez. “Quality of Uncertainty Quantification for Bayesian Neural Network Inference.” In: *proceedings at the International Conference on Machine Learning: Workshop on Uncertainty & Robustness in Deep Learning (ICML)*. 2019.

[98] G. Golub, V. Chuba, and S. Kukharets. “Determining the magnitude of traction force on the axes of drive wheels of self-propelled machines.” In: *Eastern-European Journal of Enterprise Technologies* 4 (2017), pp. 50–56.

[99] F. Buse. “Development and Validation of a Deformable Soft Soil Contact Model for Dynamic Rover Simulations.” PhD thesis. Tohoku University, 2022.

[100] F. Buse. "Using superposition of local soil flow fields to improve soil deformation in the DLR Soil Contact Model-SCM." In: *5th Joint International Conference on Multibody System Dynamics*. 2018.

[101] D. Duvenaud. "Automatic model construction with Gaussian processes." In: (2014). DOI: 10.17863/CAM.14087.

[102] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. "GPflow: A Gaussian process library using TensorFlow." In: *Journal of Machine Learning Research* 18.40 (Apr. 2017), pp. 1–6.

[103] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[104] K. Königsberger. "Das Lebesgue-Integral." de. In: *Analysis 2*. Ed. by K. Königsberger. Springer-Lehrbuch. Berlin, Heidelberg: Springer, 2000, pp. 227–260. ISBN: 978-3-662-05702-5. DOI: 10.1007/978-3-662-05702-5_7.

[105] J. C. Helton and F. J. Davis. "Illustration of Sampling-Based Methods for Uncertainty and Sensitivity Analysis." In: *Risk Analysis* 22.3 (2002), pp. 591–622. DOI: https://doi.org/10.1111/0272-4332.00041. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0272-4332.00041.

[106] T. Iwanaga, W. Usher, and J. Herman. "Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses." In: *Socio-Environmental Systems Modelling* 4 (May 2022), p. 18155. DOI: 10.18174/sesmo.18155.

[107] J. Herman and W. Usher. "SALib: An open-source Python library for Sensitivity Analysis." In: *The Journal of Open Source Software* 2.9 (Jan. 2017). DOI: 10.21105/joss.00097.

[108] G. E. B. Archer, A. Saltelli, and I. M. Sobol. "Sensitivity measures, anova-like Techniques and the use of bootstrap." In: *Journal of Statistical Computation and Simulation* 58.2 (1997), pp. 99–120. DOI: 10.1080/00949659708811825. eprint: https://doi.org/10.1080/00949659708811825.

[109] R. L. Iman and W. J. Conover. “A distribution-free approach to inducing rank correlation among input variables.” In: *Communications in Statistics - Simulation and Computation* 11.3 (1982), pp. 311–334. DOI: `10.1080/03610918208812265`. eprint: `https://doi.org/10.1080/03610918208812265`.

[110] G. Pleiss, J. Gardner, K. Weinberger, and A. G. Wilson. “Constant-Time Predictive Distributions for Gaussian Processes.” In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 4114–4123.

[111] J. Wenger. personal communication. Aug. 8, 2013.