SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Gaussian process models for wheel locomotion

Witold Merkel

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Gaussian process models for wheel locomotion

Gaußprozesse als Modell für Radbewegung

Author:	Witold Merkel
Supervisor:	Dr. Felix Dietrich
Advisor:	Vladyslav Fediukov
Submission Date:	15.06.2023

I confirm that this master's thesis in data engineering and analytics is my own work and I have documented all sources and material used.

Munich, 15.06.2023 Witold Merkel

Acknowledgments

First and foremost I have to thank my advisor, Vladyslav Fediukov. The guidance he offered was irreplaceable and crucial for the completion of this thesis. He helped me on daily bases to push further with my effort, by giving ideas and consulting on what was already there.

additionally I would like to thank my supervisor, Dr. Felix Dietrich who helped me with the organisation of the work and found time for all the meetings we had to discuss all the doubts I had. His experience turned out to be indispensable in completing my work, by guiding my step by step through the whole period of work.

Abstract

This thesis explores the problem of finding an optimal kernel for Gaussian Process Regression. Gaussian Process Regression is a widely used technique for modeling nonlinear relationships between variables. The choice of kernel function is critical to the performance. However, the search for an optimal kernel is a challenging task due to the high-dimensional parameter space and the lack of a universal criterion for kernel selection. In this thesis, we propose multiple ways to search for an optimal kernel. To evaluate the proposed methods, we conduct experiments on two datasets. Data that will be used is a wheel-soil interaction data, with the task of predicting the force acting on the wheels. The results show that the automated methods can be effective in finding good kernels and may significantly reduce the time and effort required for kernel selection. Overall, this thesis provides insights into the problem of kernel selection for Gaussian Process Regression and proposes multiple ways to tackle this problem. The proposed methods can be applied to various applications. To summarise the thesis, the final section will conclude our results and provide directions for the further development.

The dataset has multiple overlapping features present, in order to address the problem of performance and combat memory dimensionality reduction techniques are implemented. We will try to use different methods to choose the best possible kernel for given data. Those will be based on the data and its specifics. In order to get results as good as possible at the later phases we will look into combinations of kernels. The final models that are benchmarked and compared according to their accuracy, the number of required data points, and possible further improvements.

Contents

1 Problem introduction

Over the last few decades, among everything two very important phenomena have been happening, and with each passing year, they are advancing exponentially even further. The first one is the realization of how much potential is hidden within the data. Humans have understood the importance of generating, gathering, and analyzing data, and as a result, there has been explosive growth in the amount of data that is being produced each day. With the appearance of more data, the natural need for storage and fast access to it has appeared to. The second phenomenon is the popularisation of data-driven solutions to problems. The copious amounts of data have given rise to new methods and techniques to leverage it. One way of utilizing the data is to create a machine learning system that takes all the data in and gives beneficial information that can improve or even replace existing mechanisms. Machine learning algorithms can learn from the data and improve their performance over time. With more data coming in, machine learning models become more accurate and reliable. This led to a lot of new applications that were not possible before, including self-driving cars, image and speech recognition, and predictive analytics. One of many examples of machine learning is the Gaussian processes, one of their advantage is their reliability and the ability to include uncertainties, they can be also coupled with other methods such as neural networks or even stacked together to work on multiple levels of data fidelity to achieve better performance and solve problems for easily.

Machine learning can be applied to a wide variety of fields, starting from chatbots to self-driving cars. I chose to apply machine learning for space rovers, because the high cost of space exploration is well-known, with each mission often requiring billions of dollars and extensive preparations before they even start. To drive technological progress forward, scientists and engineers have devised various experiments that are supposed to reduce costs. Even more, researchers have attempted to replicate the environment of a different planet in a laboratory setting to enable relatively low-cost and frequent experiments. Fortunately, both simulated and real-world experiments can yield vast amounts of data.

The motivation behind our work is quite simple and comes from the need to solve a very crucial problem. The goal of this thesis is to develop a method for selecting the optimal kernel for the Gaussian process regression models. In our case, we aim to predict the traction force for a wheel of a Mars rover by using Gaussian processes and try to find the best possible kernel for this. The Gaussian process regression is a powerful tool that can be used to model and predict the motion of the Mars rovers. However, the accuracy of said model is heavily dependent on the selection of the right kernel function. Therefore, the main objective of our work is to identify the best kernel function for the Gaussian process regression model, the results are simply presented on this data.

In section 2 at the beginning, we take a closer look at the mechanics of the rover machines to have a basic understanding of the physics problem behind our code. After that, we make an introduction to the Gaussian processes starting with statistics and going deeper. In the end, we give an example of the practical usage of those to help the reader visualize their work. After that, we get to a mathematical explanation of kernels, and how they are incorporated into Gaussian processes, and at the end we give an overview of an automatic kernel selection process based on an article. In the end, we give an introduction to embedding methods such as principal components analysis and t-distributed stochastic neighbor embedding, as they are used in our work.

Section 3 describes the work done and the results of this thesis. At first, we write the motivation behind the work, and after that, we introduce the task at hand. We then approx the datasets that are being used in our work, and we describe which metrics will be used in the experiments. After that we start describing the work done, firstly the exploratory data analysis that allowed us to understand the data more deeply and decide for example which features to use later on and which to omit and far more. Later on, we introduce our algorithmic approaches for the solutions of our search in descriptions. Next, we present the way we have decided to implement them and after that, we show the results of the experiments performed followed by a comparison of the results achieved.

In the end, we summarise the thesis by concluding our results and suggesting the next steps that can be performed to achieve more.

2 Theoretical background

2.1 Wheel locomotion model

2.1.1 History

The problem of wheel-soil interactions has been present since the first developments of unmanned rovers used in space research. Mars has been a major point of interest for space exploration since the 1960s [1]. With the technological development, the goal has become closer as the Mars Pathfinder was deployed on December 4, 1996, successfully landing on Mars 7 months later. Since then multiple rovers have explored Mars' soil. The major challenge for successful travel on the planet is the uncertainty of wheel interactions with soft soils such as sand or mud. With the latest rover landing in the Jezero crater, the problem of efficient traversal became crucial [2]. With the experience gained from multiple previous missions as well as experiments performed on Earth, the physics model and properties of wheel interactions are being analyzed utilizing multiple ground contact models in various levels of detalization. Machine learning and models combining could provide useful, real-time locomotion predictions [3]. Example of a rover presented in figure 2.1.

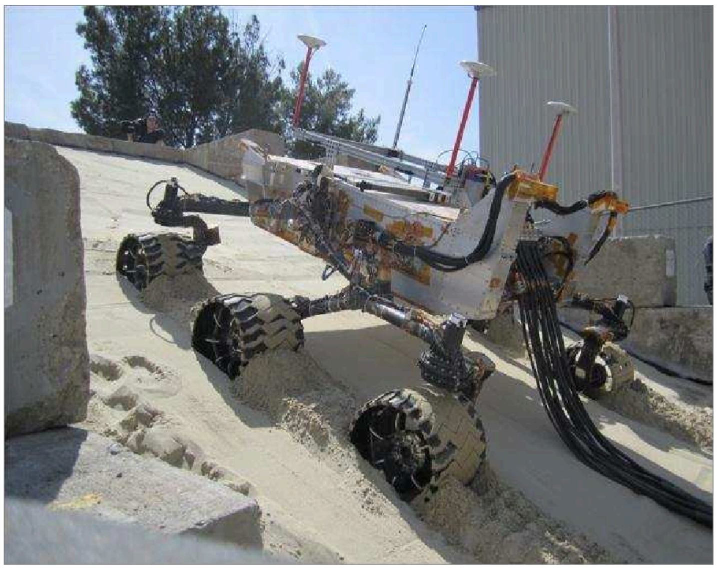


Figure 2.1: Mars Science Laboratory (MSL) engineering model tested on dry, loose sand taken from [4].

2.1.2 Terramechanics for Real-time Application (TerRA)

One of the wheel locomotion models that is used in this thesis is called Terramechanics for Real-time Application or shortcut TerRA and is described by Barthelmes [5]. The developed model can simulate traction force acting on the wheel, which we will use as a target function in our predictions. The model was developed with a focus on the forces of wheeled mobile robots especially in sandy terrains such as Mars soil. TerRA is a one-point model that should be much faster than real-time models evaluated on standard personal computers.

The need for a model that simulates and allows for corrections to be made is visible from the fact that even in successful missions such as:

- Spirit described in [6],
- Opportunity described in [7],
- Curiosity described in [8],

There was still a lot of uncertainty regarding the wheel's interaction with the sandy soil. Those three missions despite being regarded as huge successes had their share of technical problems. Each of the rovers has bigger or smaller problems with slippage and sinkage on sandy surfaces. Those problems are described in [6, 7, 8, 9]. Even a decision was made to avoid certain areas known to be problematic, which results in less data being collected for some places.

TerRA has specified goals and to meet those certain properties were picked and they are mandatory for the model to be relevant as written in [5]. In table 2.1 we can see all those effects with their description.

Table 2.1: Properties of TerRA data presented with descriptions.

Property	Description
Plastic sinkage	Results from deformation of sand when a wheel moves in it
Slip-sinkage	Sinkage developing with wheel slippage can result in embedding
Slip and traction force	Forces that are responsible for the positive acceleration
Resistance force	Resistance of the soil counteracts the traction force
Convergence	Stable and reproducible state has to be achieved after transient phase

The task in terms of predicting here is to predict the traction force 2.1. Which is given by:

$$F_{tr} = \int_{x_0}^{x_e} \int_{-w/2}^{w/2} \tau_{max}(1 - e^{(j(t)/k_j))} dy\, dx, \tag{2.1}$$

where:

- τ_{max} - maximum stress that the soil can bear, formula taken from [10],
- $j(t)$ - slip length from [10],
- k_j - soil parameter from [10].

In 2.1 the first integration is over the length of a chord of the wheel between the two points of intersection with sand, so it is the area where the traction force is generated. The second integration is over the wheel width. The function that we integrate over is the formula for stress that the soil can withstand.

We can also look at the slippage and sinkage graphs of an example run of a rover. It will show us if and how those two effects are connected. It will also help to visualize how the parameters of the model and environment affect the work of our rover.

In figure 2.2 we can see an example of slippage in one run of a rover, it is visible that it depends on the angle of the slope and at the start of the simulation is equal to one, before the rover can grip some traction.

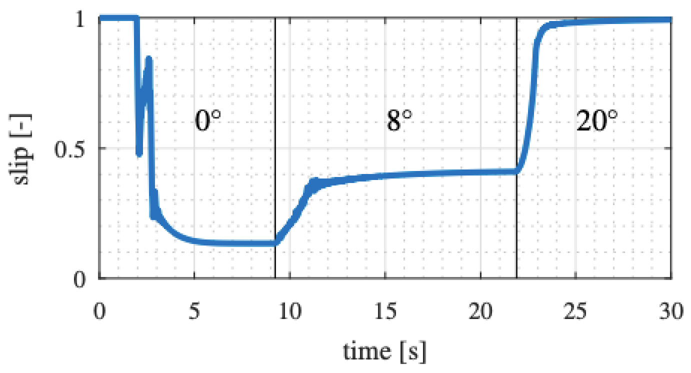


Figure 2.2: Presented example of slippage through the time taken from [5].

If we look at figure 2.3 and then at figure 2.2 we can see that there is a link between sinkage and slipage. The bigger the slippage gets the lower the rover sinks into the sand. It means that it digs itself into the ground.

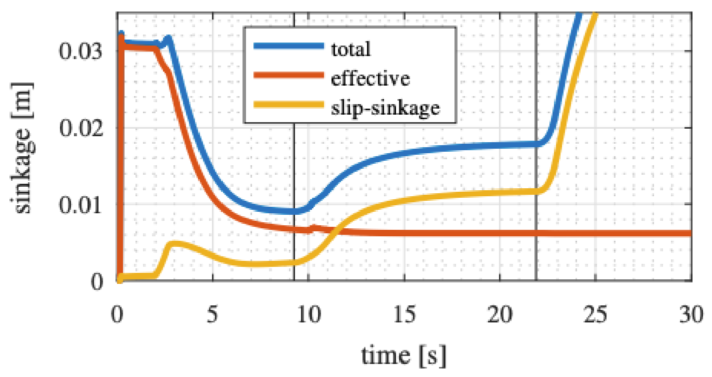


Figure 2.3: Presented example of a sinkage through the time taken from [5].

Figures 2.2 and 2.3 show that the situation of a rover can change drastically in a matter of moments in the TerRA model. That is a reason why there is a big need for a valid and accurate prediction model, that will help with modeling based on the incoming terrain data. It can help in avoiding situations such as the rover burying itself or slipping off a hill, by allowing to simulation of similar situations and providing a pattern of behavior that is right for each action and situation.

2.1.3 Soil Contact Model (SCM)

The second wheel locomotion model used in this thesis is called Soil Contact Model or SCM for a shortcut. It is described in [11]. Nearly all of the numerical solutions of the simulation of plastic soil can be divided into two main categories:

1. semi-empirical models,
2. physics-based models.

The soil contact model is semi-empirical, it is based on a small number of parameters, that through experimentation are cheap to get. Once the model has parameters like:

- soil strength,
- stress distributions,

it predicts pressure in the contact patch and traction performance using empirically known relations based on assumptions that make the problem analytically available.

The theoretical model aims at simulating the real physical phenomena at the interface of the tire and terrain with the help of complex and much more computationally demanding, numerical approaches. This approach in general lead to more detailed results, but the down-side is that they require much more computational power. Another upside is that they do not have the same limitations as semi-empirical models.

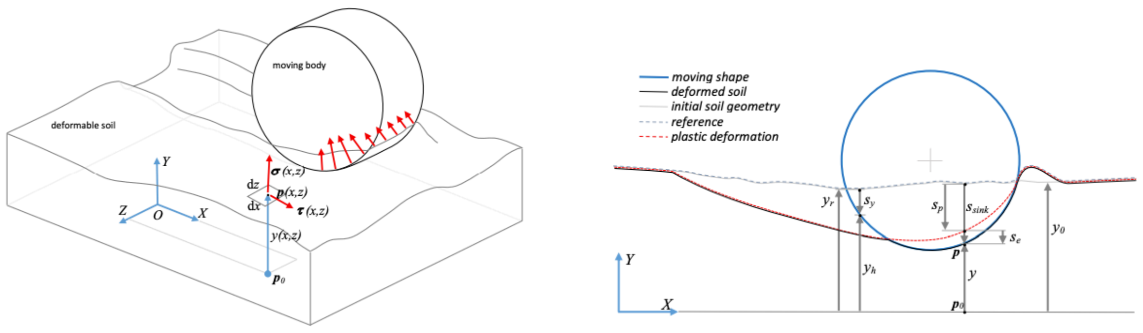
The soil contact model is based on a semi-empirical model with a minimal amount of parameters for it to be easy to adjust and fast to do computations. SCM maybe understood as a more general version of the Bekker - Wong model in terms of the wheels' interactions. There is a formula called the Bekker formula presented in equation 2.2, which gives the movement of the wheel on deformable soil as sand. The relationship between pressure and sinkage is given by an equation of σ, which is the contact patch pressure:

$$\sigma = (\frac{k_c}{b} + k_\phi) y^n, \tag{2.2}$$

where:

- k_c - an empirical coefficient representing the cohesive effect of the soil,
- b - the length of the shorter side of the rectangular contact area,
- k_ϕ - is an empirical coefficient representing the stiffness of the soil,
- y - wheel sinkage,
- n - the exponent of sinkage.

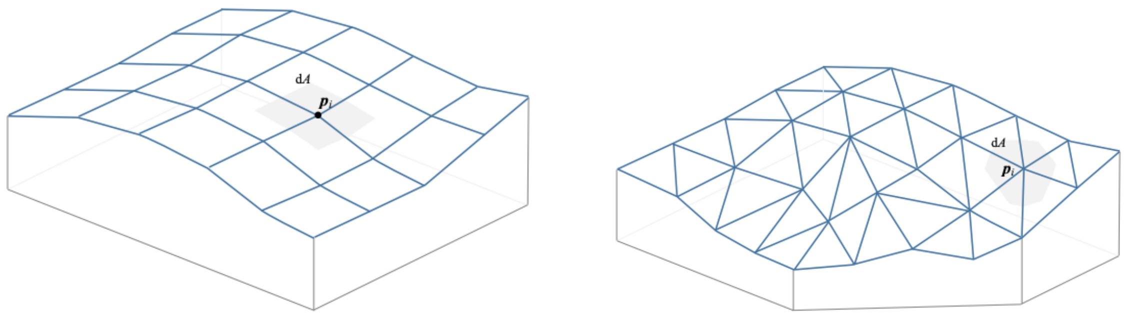
We can see in figure 2.4 the dynamic of the movement of a body in plastic soil and how the process runs.



(a) 3D view of the movement and soil deformation. (b) Cross section of the movement and soil deformation with variables.

Figure 2.4: Object movement and soil deformation that is caused by it taken from [12].

The implementation of the soil contact model, we are using is slightly different in terms of the mesh of the grid that is being used. Instead of the standard rectangles we are using triangles, which is helpful in certain examples such as along roads, in our case of Martian travel it is relevant. We can see the difference in figure 2.5.



(a) Standard grid used in SCM. (b) Triangle grid used in our implementation of SCM.

Figure 2.5: Different mesh grids used in SCM taken from [12].

The upsides of this model are that it supports an optional refinement of the gird mesh to be able to introduce fine details like tire threads and lugs. Furthermore, it can work with tiers that change shape and generate basic obstacles. The biggest minus is the fact that the model has problems with the occurrence of wheel steers happening in a place and pushing material apart.

2.2 Gaussian processes

Gaussian processes are the basis of models created in this thesis, so it is necessary to understand what is their basis, how they are created, how the training is done and what is Gaussian process regression.

2.2.1 Gaussian distribution

The Gaussian distribution 2.3 know also as normal distribution was introduced by Carl Gauss in 1809 in [13]. In one dimension density function, it is given by equation:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \tag{2.3}$$

where:

- μ - mean of the distribution, responsible for where the middle of the distribution curve is,
- σ - standard deviation of the distribution, responsible for the width spread of the distribution curve.

When we are speaking of a random variable we can use the notation $X \sim N(\mu, \sigma)$ it means that the random variable X is from a normal distribution with a mean equal to ν and standard deviation equal to σ. The most popular normal distribution is a standard normal distribution, meaning it has a mean equal to 0 and a standard deviation equal to 1 - $X \sim N(0, 1)$. We can transform any one-dimensional normally distributed random variable to standard by using this transformation: $X_{new} = \frac{X_{old}-\mu}{\sigma}$.

In figure 2.6 we can see how the parameters such as mean and standard deviation affect the density function of a Gaussian distribution.

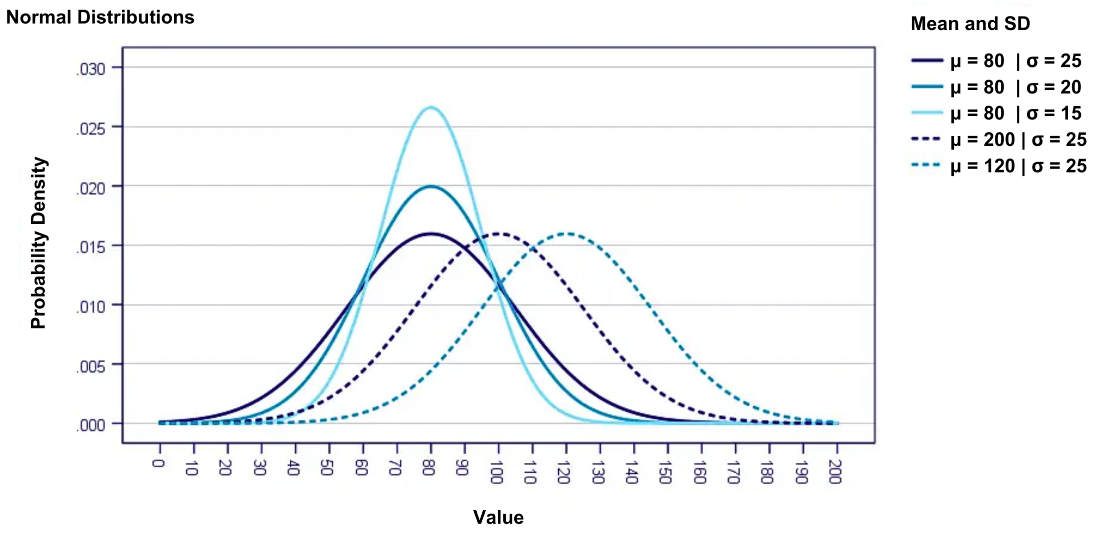


Figure 2.6: Standard normal distribution taken from [14].

Once we are familiar with the one-dimensional Gaussian distribution we can move on to multi-dimensional Gaussian distributions. This is relevant because a Gaussian process is a stochastic process in which every finite subset of random variables has a multivariate normal distribution.

In the case of a d dimensional normal distribution, its density function is given by equation:

$$f(x) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)). \tag{2.4}$$

In equation 2.4 we can name three parameters:

1. d - number of dimensions,
2. μ - vector of expected values (means) of each distribution,
3. Σ - covariance matrix.

2.2.2 Bayesian probability theory

The whole idea behind the Gaussian process regression is based on the Bayesian inference problem as stated in [15]. In statistics inference is the process of concluding a parameter or parameters one is seeking to measure or estimate as said in [16]. Bayesian probability theory gives us the tools for performing mathematical inference using probability. The whole concept is based on the joint probability of two events, which equation is given by:

$$P(AB) = P(A|B)P(B) = P(B|A)P(A). \tag{2.5}$$

Based on 2.5 and naming H as a hypothesis and D as data we can get the formula for the probability of a hypothesis given the data, given by:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}, \tag{2.6}$$

like in [17].

The equation presented in 2.6 is called Bayes' rule, in which we have different entities such as:

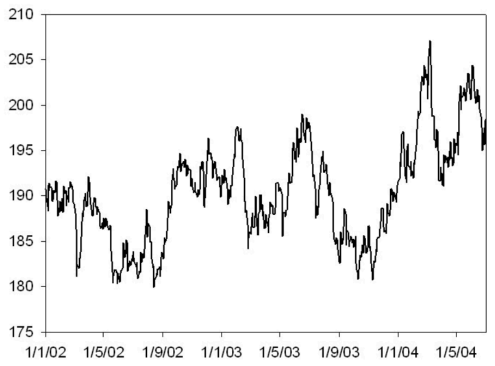
- $P(H|D)$ - posterior probability, it reflects the probability of the hypothesis after considering the data,
- $P(D|H)$ - likelihood function, it gives the probability of the data arising from the tested hypothesis,
- $P(H)$ - prior probability, it reflects the prior knowledge before the data is taken into consideration,
- $P(D)$ - normally plays the part of a normalizing constant and is integrated out by integrating $P(D|H) * P(H)$ over all H using the Bayesian rule in 2.6

as explained in [17].

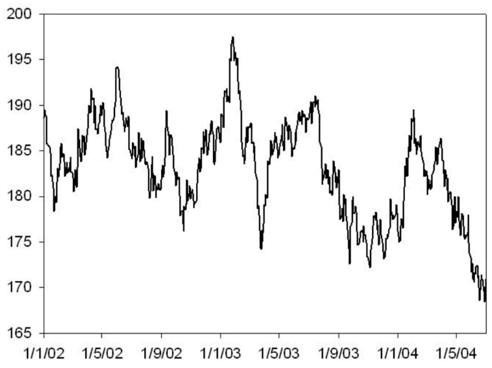
2.2.3 Stochastic processes

Stochastic process $X = X(t)$ where $t \in T$ is a collection of random variables. T is an index set and for each $t \in T$ a $X(t)$ is a random variable. A common interpretation is for t to be regarded as time and then we can understand the $X(t)$ as the state of our process at the time t. If the index set is countable we call the $X(t)$ a discrete-time stochastic process and when the index set is a continuum we call the $X(t)$ a continuous time process [18].

An example realization of a stochastic process $X(t)$ is called a sample path. Those can model real-life events like stock exchange share prices or epidemic parameters like new daily infections. In figure 2.7 we can see the exchange rate between the British pound (GBP) and Japanese yen over (JPY) from January first, 2002 till July thirty-first, 2004.



(a) True exchange rate from GPY to JPY.



(b) Simulation of exchange rate from GPY to JPY created using stochastic process.

Figure 2.7: Example occurrence of stochastic processes in exchange rate taken from [19].

If we consider a continuous time stochastic process $X(t)$, where $t_i \in T$, we can say that it has independent increments if for all $t_i \in T$ $t_0 < t_1 < t_2 < ... < t_n$ random variables $X(t_{i+1}) - X(t_i)$ are independent. Furthermore, it is said that a stochastic process possesses stationary increments if $X(t + s) - X(t)$ has the same distribution for all $t \in T$. Meaning that the stochastic process possesses independent increments if the changes in the processes' value over nonoverlapping time intervals are independent random variables and it possesses stationary increments if the distribution of the change in value between any two points depends only on the distance between those points [18].

2.2.4 Gaussian Processes

We want to model output force as a function of different parameters of the rover, to do so we can consider giving a prior probability to every possible function that can fit, and we can give a higher probability to functions that in our consideration are more likely to be correct. The functions that will have a higher probability can be found by looking at our data and analyzing some patterns along with factoring in the knowledge of the problem. This way of thinking can be problematic as we can have a problem with an uncountable set of functions that we want to process in a finite possibly the lowest possible time. The answer to this problem can be presented in Gaussian processes because standard machine learning ideas like deep neural networks, random forests, or linear regression do not work well with the infinity concept [15]. Gaussian process Z depends on two functions:

1. mean function $m(x) = E(Z(x))$,
2. covariance function $k(x, x') = cov(Z(x), Z(x'))$

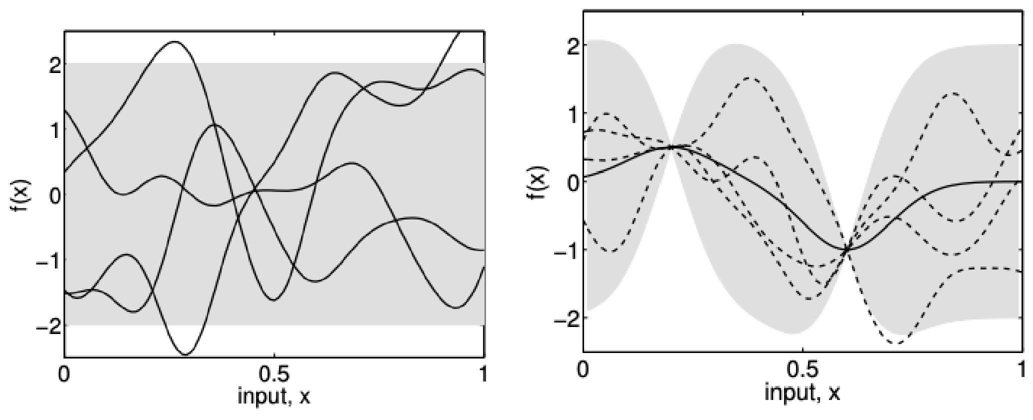
based on [20]. The covariance function is also called a kernel, more on that in 2.3.

We can consider Gaussian processes as the generalization of Gaussian distribution. Probability distributions are used in describing random variables, which are scalar or vector values respectively for one-dimensional and multi-dimensional distributions. When we want to describe functions accurately we shall use stochastic processes.

In general Gaussian process is a non-parametric model that can be used to represent the distribution of functions. Contrary to parametric models like decision trees or neural networks, a non-parametric model cannot be represented by using a fixed set of parameters, because of this we use distribution functions to sample the parameters. Another way of looking at a Gaussian process is a collection of random variables, where any subset of them has multivariate Gaussian distribution [15].

Creating a regression model base on Gaussian processes is based on the idea of Bayesian inference. Where the data information is just the training data. We can also understand the kernel function as the prior, more on kernel functions in section 2.3. To visualize the prior distribution of the functions, which is determined by the kernel, we can sample several functions. Once the training data is taken into account then the Gaussian process makes the prediction based on the posterior distribution.

In figure 2.8 we can observe the process of training the Gaussian process regression. Figure 2.8 depicts the process of sampling from prior distribution and also depicts functions sampled from posterior distribution (dashed line) and their mean (solid line). On both of those figures, the grey zones are confidence intervals, we can see that they are only points in the points of training data.



(a) Prior function distribution (b) Posterior function distribution

Figure 2.8: Gaussian process regression training process taken from [15].

The training process is commonly done by the maximization of the log-likelihood function of the training data concerning each hyperparameter of the model. In this process there is a similarity to the optimization of artificial neural networks, a gradient-based algorithm is most likely used [21, 22].

2.2.5 Examples of practical use

Gaussian processes can be used likewise for regression and classification problems and achieve excellent results. In this thesis, we are making a predictor where a target variable is a force, so we have a regression problem, that is why we will use a regression example in this subsection.

A good example of the usage of kernels in models is presented in [23]. This article explores the impact of the composition of concrete on its strength. Concrete strength is how much pressure it can withstand before crashing. Trying to model the strength of the concrete by using the physical process of creating the concrete, which is just adding elements together, because of that it is relevant to use a simple additive model. In this example the strength is a function of different components, those being: the number of ingredients used, the age of the concrete, and noise like in [24].

The output function is the sum of 8 different functions of one-dimensional features and noise as presented in 2.7 from [24].

$$
\begin{aligned}
f(x) = f_1(cement) + f_2(slag) + f_3(flyash) + f_4(water) \\
+ f_5(plasticizer) + f_6(coarse) + f_7(fine) + f_8(age) + noise
\end{aligned}
\tag{2.7}
$$

All the functions were modeled using a Gaussian process with a squared exponential kernel. If we add them all together with the noise then we get a single Gaussian process model in which the kernel has nine additive components. The last thing that we can look at is figure 2.9 in which we can see the models after the process of learning, which was done by the maximization of the marginal likelihood.

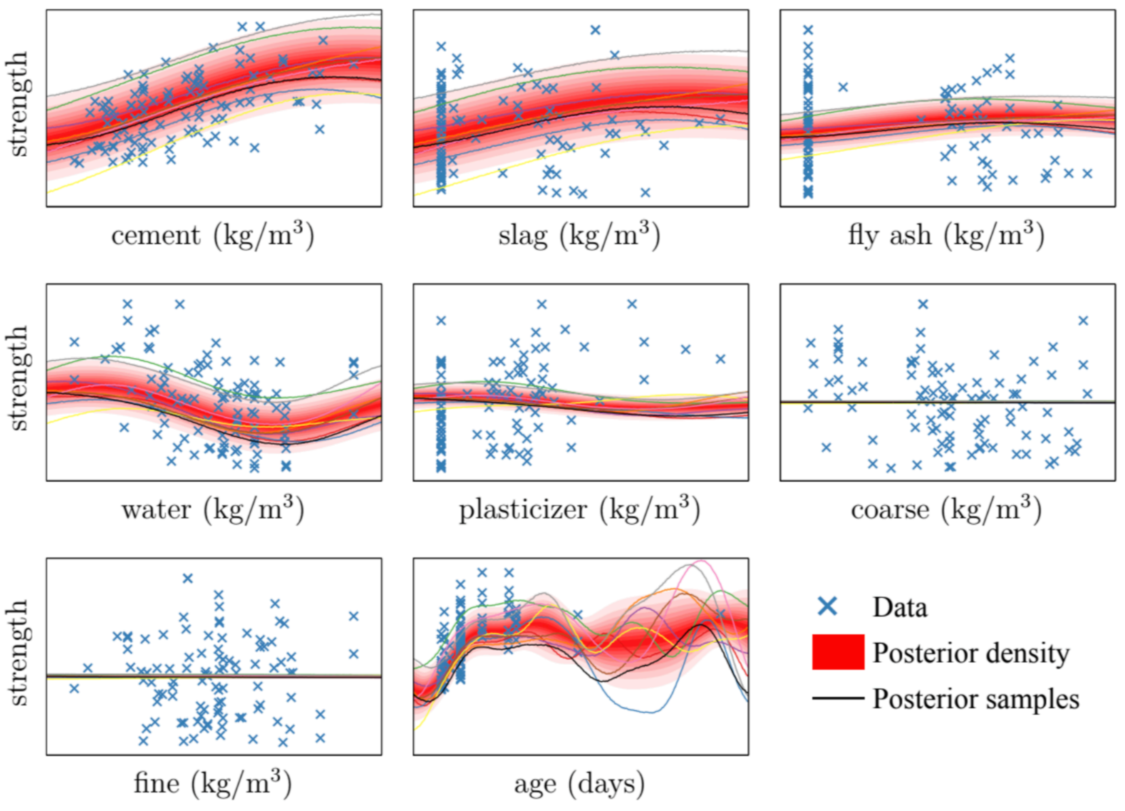


Figure 2.9: Posterior distributions of Gaussian processes models taken from [24].

2.3 Kernels

2.3.1 Mathematical background

A positive definite kernel is a generalization of a positive definite function. Symmetric function f is positive 2.8 when it fulfils the equation:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j f(x_i x_j) \geq 0, \tag{2.8}$$

on a non empty indexed set $\mathbb{X}$ for every $x_1, \ldots, x_n \in \mathbb{X}$, given $n \in \mathbb{N}, c_1, \ldots, c_n \in \mathbb{R}$.

When we consider a family of kernels then we get two important properties that will allow us to combine kernels. Those are that if $(K_i)_{i \in \mathbb{N}}$ then:

- $\sum_{i=1}^{n} \lambda_i K_i$ given $\lambda_1, \ldots, \lambda_n \geq 0$ is also a kernel,
- $\prod_1^n K_i^{a_i}$ given $a_1, \ldots, a_n$ is also a kernel.

In this thesis, the x and x′ will be vectors in Euclidean space, but in general, kernels can be defined on text, graphs, or even images [24]. Euclidean space is a space over the real numbers such that the associated vector space, is a space with an inner product and has a finite number of dimensions. Gaussian process models use kernels to define the prior distribution by its covariance 2.9 between any two function values shown in equation:

$$Cov[f(x), f(x')] = k(x, x'). \tag{2.9}$$

We will examine basic kernels, their covariance function, plots of those functions, and functions sampled from a Gaussian process with this prior distribution. All three different types of kernels can be distinguished and more can be achieved by creating a combination of those three. Those three are:

1. Squared exponential,
2. Periodic,
3. Linear.

This distinction is taken from [24].

The squared exponential kernel 2.10 has a covariance function presented in equation:

$$k(x, x') = \sigma_f^2 exp(-\frac{(x - x')^2}{2l^2}). \tag{2.10}$$

We can also see a plot of the covariance function in figure 2.10 (a) in the upper graph and some functions sampled form a Gaussian process with this prior distribution in figure 2.10 (a) in the lower graph.

The periodic kernel 2.11 has a covariance function presented in equation:

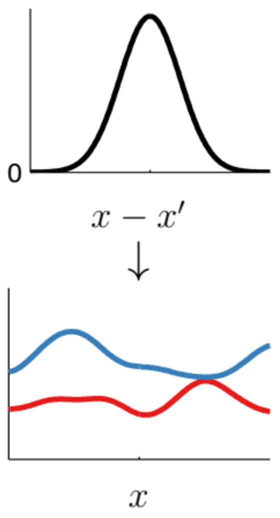
$$k(x,x') = \sigma_f^2 exp(-\frac{2}{l^2}\sin^2(\pi\frac{x-x'}{p})). \tag{2.11}$$

We can also see a plot of the covariance function in figure 2.10 (b) in the upper graph and some functions sampled form a Gaussian process with this prior distribution in figure 2.10 (b) in the lower graph.

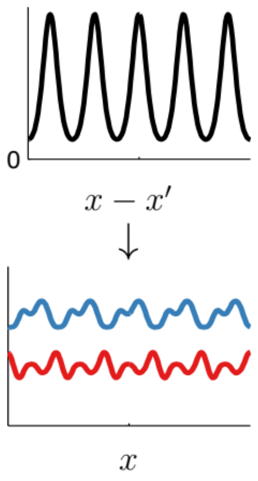
The linear kernel 2.12 has a covariance function presented in equation:

$$k(x,x') = \sigma_f^2(x-c)(x'-c). \tag{2.12}$$

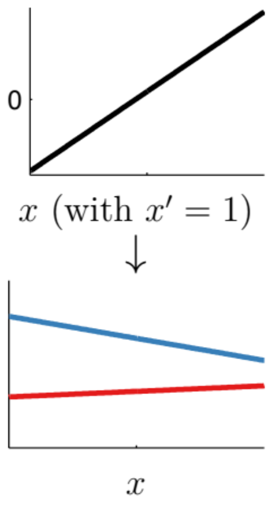
We can also see a plot of the covariance function in figure 2.10 (c) in the upper graph and some functions sampled form a Gaussian process with this prior distribution in figure 2.10 (c) in the lower graph.



(a) Plot of covariance function and sample functions of a squared exponential kernel from equation 2.10.



(b) Plot of covariance function and sample functions of a periodic kernel from equation 2.11.



(c) Plot of covariance function and sample functions of a linear kernel from equation 2.12.

Figure 2.10: Presentation of examples of different kernels taken from [24].

Often there is a need to use more complicated kernels, one way to achieve it is to combine kernels using addition and multiplication. Those operations keep the property of positive definite of a function, so there is no need of checking if the final kernel is still a kernel.

2.3.2 Kernels in Gaussian processes

As mentioned before the kernel in the Gaussian process is the covariance function. It gives us the prior distribution and has a part in the posterior distribution as presented in 2.6.

In figure 2.11 we can see that it is extremely important to choose the right kernel as it has a big impact on the confidence intervals.

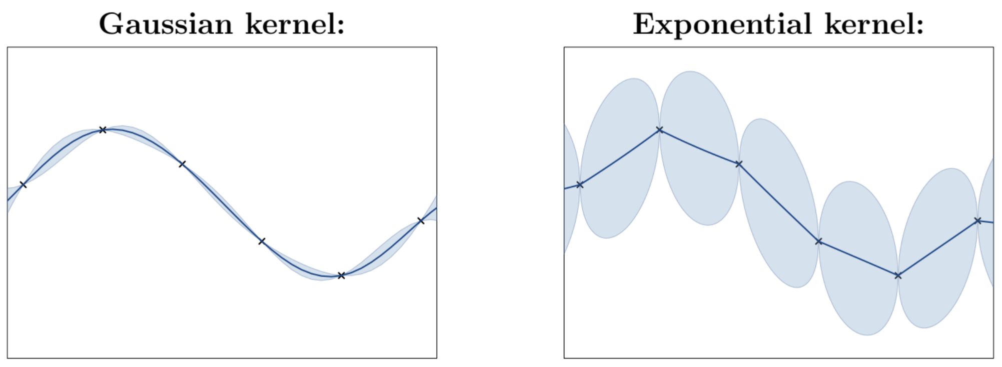


Figure 2.11: Impact of the kernel on the Gaussian process taken from [20].

In the exponential kernel part of figure 2.11 we can see that the confidence intervals outside of the interpolation points are widely inaccurate when compared to the Gaussian kernel part. We need to answer the following questions among others to find the best suiting kernel for a given situation:

- is the function stationary, meaning it is a realization of a stationary stochastic process [18],
- is the function differentiable and what is its regularity,
- do we expect particular patterns in the function,
- do we expect particular trends in the function?

Like presesnted in [20].

Once we answer those questions we may be able to greatly improve the accuracy of our model when compared to a random kernel.

2.3.3 Automatic kernel selection

The most important part of this thesis will be about trying to find the best suitable kernel for the data provided by the TerRA model [5]. The usage of Gaussian processes regression and Bayesian formulation is becoming more and more popular thanks to the fact that it not only incorporates uncertainty but also allows for the avoidance of all the undesirable features of artificial neural networks [25]. One of the most time-consuming aspects of neural networks is looking for the best hyperparameters and the best starting condition. Thanks to the fact that Gaussian processes are computationally effective and that non-linear learning is easy the processes gain more popularity [25].

Selection of the kernel shall be greatly important as it affects the mean prediction, accuracy of the model, but also confidence intervals of our prediction. This is the reason that selecting an appropriate kernel is so important, it gives way to a problem in which we need to select a kernel, then evaluate the Gaussian process, and at the end compare the results. This specific problem may get extremely complicated as more than one kernel can accurately represent the data, much like more than one artificial neural network can have high accuracy for a given problem. This problem has an additional difficulty as most of the Gaussian process regression models are treated as black boxes, meaning we do not understand the model's reasoning for decision-making [25].

The method for the kernel selection is the approximate Bayesian computation like said in article [25]. One advantage of this method is that it is very general, there is no need for any assumptions about the likelihood functions and also there is no need for any extra evaluation that can discriminate some kernels. Another advantage is that the algorithm is not affected by the dimensionality of the kernels and can deal even with kernels of different dimensions comparing them on the go. In this algorithm the goal is to obtain the best and most efficient approximation of the posterior distribution [25].

The algorithm is based on comparing the simulated data and the observed data and accepting the simulation if it is close enough to the real data, the distance threshold is set by the user. All the mathematical formulas along with two examples of usage are presented in [25]. In one of those examples, the difference between the different kernels is not significant enough to create a distinguishable difference, hence the algorithm chooses the simpler one. In the second example with the progress of the work and with the minimization of the threshold, the algorithm begins to discard the simpler kernels as they are not able to model the data properly. This method gives a way to choose kernels by following steps in a systematic and somewhat consistent way for different situations.

The examples illustrate that for not too complicated data standard simple kernels can be enough to reach satisfactory results, but it also shows that in examples that have underlying difficulties, there is a need for complicated kernels. When the threshold parameters are tightened then the more complicated kernels are getting a lead in case of complicated underlying relationships in the dataset.

2.4 Embedding methods

2.4.1 Principal Components Analysis

Principal component analysis was created before the 1940s but had to wait for the possibility of width application because the computations were too complicated for the times, similar to artificial neural networks. Thanks to the advancement of computers and their calculation power a lot of statistical methods become popular such as principal components analysis [26]. This method gives the possibility of reducing the high dimensionality of a dataset to only a few without losing too much information, by projecting data points onto principal components [27]. The basis of the principal components is the same as the major axis regression, because of that the two methods share the same problems and restrictions, for example, the need for normality. Before applying the method the variables should be from a normal distribution and if they are not they should be transformed so that they have it. One important thing to look out for is outliers that can highly dominate and distort the results [28].

Dimensionality reduction in this case is gained by obtaining new variables that are linear combinations of the old ones. Those new variables are called principal components and they must satisfy specific mathematical conditions for them to be valid. We have the dataset in a matrix form $\mathbb{X}$ [26].

$$\mathbb{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & x_{p3} & \dots & x_{pn} \end{bmatrix}$$

From this matrix, each of the columns represents a point in the p-dimensional space. Since the matrix is finite the goal of the principal component analysis is to find linear combinations such as their variance is the maximum possible while still fulfilling certain properties. If we denote the covariance matrix of $\mathbb{X}$ as $\mathbb{S}$ then we can get the principal components from the equation:

$$\mathbb{V} = \mathbb{A}'\mathbb{X}, \tag{2.13}$$

where:

- $\mathbb{V}$ - the matrix with new variables,
- $\mathbb{A}$ - the matrix of orthonormal eigenvectors of the matrix $\mathbb{S}$,
- $\mathbb{X}$ - the matrix with data,

like in [26].

To find what principal components that we looking for in equation 2.13, firstly we need to solve equation:

$$|\mathbb{S} - l\mathbb{I}| = 0, \tag{2.14}$$

where:

- $\mathbb{S}$ - the variance-covariance matrix of order p,
- l - the characteristic root of the determinant equation,
- $\mathbb{I}$ - the unit matrix of order p.

The form of equation 2.14 is a polynomial equation with degree p with the unknown variable being l, hence it has p distinctive roots which can be ordered in a nonincreasing way: $l_1 \geq l_2 \geq l_3 \geq \cdots \geq l_p \geq 0$. In matrix $\mathbb{A}$ there is a pair of an orthonormal column vector $\mathbb{A}$ and root l_i. From equation 2.13 we can take the vector $\mathbb{V}_{\mathbb{1}}$ and it has the maximum variance equal to l_1 and it is the first principal component, each next column of $\mathbb{V}$ is the next principal component with variance equal to the corresponding l_i [26].

When we compare this fact with the fact that $l_1 + l_2 + \cdots + l_p = trace(\mathbb{S})$ and that it is also equal to the variance of the matrix $\mathbb{S}$, then the single l's are the shares of the variance explained by each of the principal components. Now if we look at the values $\frac{l_1}{trace(\mathbb{S})}100, \frac{l_2}{trace(\mathbb{S})}100, \ldots, \frac{l_p}{trace(\mathbb{S})}100$ we get the percentage share of the variance. The algorithm that is used for the calculation of the principal components is devised in such a way that this is a decreasing sequence. The last important fact is that the number of principal components is the same as the number of starting variables, but we can exclude some of them and lose next to nothing in terms of variance [26].

2.4.2 t-distributed Stochastic Neighbour Embedding

t-SNE is a technique for high-dimensionality data visualization. It was first proposed in [29]. It allows to give points two or three-dimensionality points for visualization and checking if there are some groups of points. This method is an upgrade from simple stochastic neighbor embedding and is easier to optimize and has one other important upgrade, it does not tend to focus the points in the center of the map in a single cluster. This characteristic is important for data that is high dimensional, but some subgroups are in the same lower dimensionality [29].

In the case of the t-distributed Stochastic Neighbour Embedding we have a conversion of the high dimensional dataset $\mathbb{X} = \{x_1, x_2, \ldots, x_n\}$ into a low dimensional data $\mathbb{Y} = \{y_1, y_2, \ldots, y_n\}$, this means two or three dimensions that can be plotted on a scatterplot [29].

The basis of t-SNE is as mentioned the Stochastic Neighbour Embedding it converts the distance between points in original space into probabilities that can be interpreted as similarities [29].

We can calculate the similarity of point x_j to point x_i is the conditional probability given by $P(j \mid i)$. The meaning behind this is the probability that x_i would pick x_j as its neighbor if the picking was based on proportion to their probability density under Gaussian distribution 2.3 with center at x_i. If we denote the standard deviation of the Gaussian distribution as σ, then the probability 2.15 is given by equation:

$$P(j \mid i) = \frac{exp(\frac{-||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{k \neq i} exp(\frac{-||x_i - x_k||^2}{2\sigma_i^2})}. \tag{2.15}$$

If we look at the points y_i and y_j, that is corresponding to x_i and x_j, we can calculate a similar quantity that is denoted as $Q(j \mid i)$ 2.16, in this instance the standard deviation is set to $\frac{1}{\sqrt{2}}$. The equation for $Q(j \mid i)$ is then given by the equation:

$$Q(j \mid i) = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2))}. \tag{2.16}$$

Now if the points x's and y's are matched correctly then the probabilities p and q will be the same. Based on that the SNE algorithm aims to minimize the error between those two while finding the low-level representation of the data. In this case, the function that is the error function is the Kullback-Leibler divergence and it is minimized over all the points with the usage of the gradient descent with the cost function 2.17 given in equation:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j log(P(j \mid i) \frac{P(j \mid i)}{Q(j \mid i)}). \tag{2.17}$$

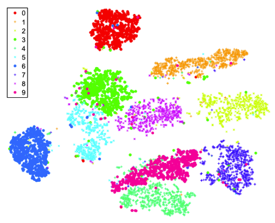
The things that are done differently in t-SNE are that there is a change in the assumed distribution and a change in the cost function. The cost function here is symmetrical and also has gradients that are easier to calculate, it solves the problem of cost efficiency. The second change is using the t-student distribution 2.18 function:

$$f(t,n) = \frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2}\sqrt{n\pi})}(1+\frac{t^2}{n})^{-\frac{n+1}{2}}, \tag{2.18}$$

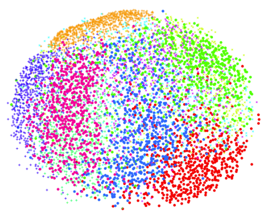
instead of the Gaussian distribution. This procedure gives us a distribution that has heavy tails, meaning the probability of being far from the center is higher than in the case of the Gaussian distribution, so the crowding in the middle problem vanishes. It also helps with the optimization [29].

Where the Γ is the gamma function, that is the extension of the factorial $\Gamma(n) = (n-1)!$. For real numbers it is defined as $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$.

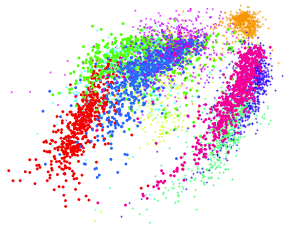
We can use a simple example to show how effective is the t-SNE in real-life situations. We will look at the MNIST dataset, which is a set of handwritten digits. Techniques there were used four mapping techniques: t-SNE, Sammon mapping, Isomap, and LLE presented in [29]. The results are presented in 2.12. We can see that the visualization that is the result of the t-SNE is better at visually separating different clusters in the data.



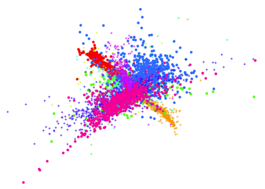
(a) t-SNE visualisation.



(b) Sammon visualisation.



(c) Isomap visualisation.



(d) LLE visualisation taken.

Figure 2.12: The results of different mapping techniques presented from [29].

3 Gaussian process models for wheel locomotion

3.1 Motivation

Firstly before starting the main part of the thesis, we will explain the goal of this project, write about the underlying motivation, short overview of the data and steps that will be taken.

The goal of this project is to find a way of automatically choosing the optimal kernel for a Gaussian process regression model on the example of data regarding the movement of a rover on Mars. We have available two datasets that are independently describing the movement of the rover in different qualities [30]. Both of the datasets have variables such as forces, torques, velocities, positions, angular velocities, and gravities that the robot experiences as well as the height map of the soil beneath the rover. The prediction task is to predict the force based on the information from the other variables and the photo.

The underlying motivation in this thesis is to enable finding the best kernels for physical phenomena hidden behind certain datasets. In our case the data is part of a bigger problem, a multi-fidelity problem of finding the forces of wheels. Optimizing the kernels will allow us to create better Gaussian processes regressions and that can lead to a better architecture of the multi-fidelity model and that finally can enhance the overall performance.

Simulations of runs of the Mars Rover have been conducted using different terra-mechanical models. One simulation consists of 151 rows of data and each row is composed of the physical parameters of the rover at a given moment and a height map of the surface below. Each record has parameters that describe output forces, output torques, coordinates, velocities, angular velocities, and gravity normal. The variables listed above all are present in three directions:

1. X,
2. Y,
3. Z.

This means that there is a total of 18 variables. Additionally to the columns from 18th to the 4112 is the image, which is a 64 x 64 picture of a 15 cm x 15 cm surface below the wheels with an example shown in figure 3.1.

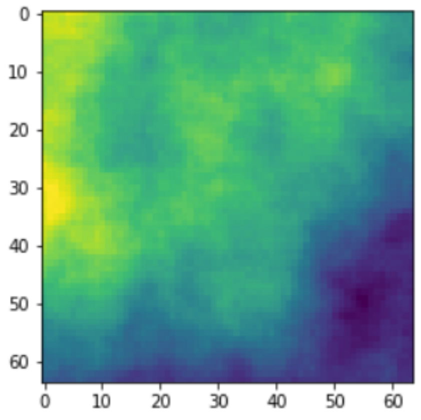


Figure 3.1: Presentation of an example of soil photo.

In the following sections, firstly we will look into the data with greater detail and then move on to the different implementations of the search. Once the implementations are detailed we will move on to experiments in different circumstances. Those will help us find the strong and weak sides of the solutions and give quantitative results for comparison. In the comparison of the results, we will look into parameters such as the error on different metrics and the time needed for evaluation.

3.2 Task description

As stated before we have two tasks. The first one is more visible that can be quantified with an error measure like MAE (Mean Absolute Error) 3.1 presented in equation:

$$MAE(y, \hat{y}) = \frac{\sum_{i=0}^{N-1} |y_i - \hat{y}_i|}{N}, \tag{3.1}$$

where:

- n - number of samples,
- y_i - true values,
- $\hat{y}_i$ - predicted values.

or RMSE (Root Mean Squared Error) 3.2 presented in equation:

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}}, \tag{3.2}$$

where:

- n - number of samples,
- y_i - true values,
- $\hat{y}_i$ - predicted values.

or time spent on learning and evaluation. This task is the prediction of the force of the rover's wheels.

This measure treats all errors the same, so we will know in general how good our prediction is.

This measure penalizes the big errors more as there is a squared formula. It means that by pairing it up with MAE 3.1 we will know if our model is making some predictions that are far away from the truth.

Another important metric is time, it does not validate the accuracy of the model, but is very important. For all the calculations done in this thesis, the hardware is a personal computer with a 2.3 GHz 8-Core Intel Core i9 and 16GB of RAM. A significant problem of Gaussian process regression is the cubic computation complexity, which means we have to monitor the time of the training as it can be unacceptable in cases of big data usage. The time measuring technique will simply subtract the timestamp of the start from the timestamp of the start. All the computations will be done more than once and then an average will be taken across all the simulations.

The second task is what is more important here - finding a way to automatically select a kernel for Gaussian processes regression given the dataset. In completion of this task, the first task will be helpful as it will allow us to have a measure of correctness of the kernel that we have chosen. More on the background of the actions are in the subsections of 3.5. Mainly why every attempt at finding the solution, what is the reasoning and mathematical theory behind it?

3.3 Dataset description

In this task alongside the surface height map, we have 18 features describing the rover and its state as the result of the TerRA or SCM model simulations. Those are force, torque, coordinate, velocity, angular velocity, and gravity in three directions.

Those 18 features can be divided into two main categories: passive and active. Passive features are those that are not directly controlled by the rover and in fact, those are treated as explanatory variables. Those features come from the readings of the external sensors and in our example features like coordinate, velocity, angular velocity, gravity as well as surface photo are included in that. If we had more sensors then features like the speed of the wind, atmospheric pressure, air temperature, humidity, or even magnetic field can be added as such features. Active features are force and torque, which are directly controlled by the rover mechanics, so we can treat them as dependent variables, as they depend on the passive features. In this thesis, we have decided to choose the force alongside the x-axis as the main variable to predict. More on the data operations and decisions will be in section 3.4.

An important aspect of this thesis is the data split. One of the common approaches is to just randomly divide data into train, validation, and test sets. After long deliberations, we decided to simply take 70% of runs as the training dataset, 20% as the validation dataset, and the last 10% as the test dataset. In the 3.4 we will show the distribution of the target variable across all the sets. The ideal situation would be for the distribution to be similar.

3.4 Exploratory data analysis

3.4.1 Introduction

The first part of this project is exploratory data analysis in which we closely examine the data and look for correlations between the variables and see how we can find maybe potential similarities and clues to look for the best-fitting kernels. Data from both models are created in the same way. Simulations of runs of the Mars Rover have been conducted using different terra mechanical models, those are TerRA and SCM. The data generated with the soil contact model is more precise, but it uses more computational power to generate.

The TerRA method has been used to generate what we would consider low-fidelity data for the multi-fidelity task. Despite that fact it can be valuable to the final model, that is because it was proven that low computational power is required to work with data coming from this model, so it is necessary to work with the best we can while trying to compose it into the final data.

SCM produces more precise data. This model results in more reliable and more accurate data points, but because of that working with them requires more computational power. To simulate a real-world scenario in our implementation, modeling, and experiments there are fewer medium fidelity points, than low fidelity.

For this part of the thesis, we have decided to use 500 simulations from each fidelity. This comes up to 151000 records used, which is the maximum amount that fits in our hardware. This can change in the experimental part of the thesis to keep the medium fidelity data count lower than the low fidelity as it is done in real life.

As the goal of the prediction task, we decided to predict the force along the x-axis. As forces and torques are correlated with coefficient one, we decided to drop all three variables describing torque, to avoid overfitting of the models. We also decided to remove two other force variables from the model as those are outputs of the Mars rover and will not be given in a real-life scenario.

In the following sections, we will consider firstly the data generated by the Terramechanics for Real-time Application model and after that, we will work with the data generated by the Soil Contact Model. We will at the beginning try to analyze standard position statistics of the whole data and after that, we will work on the densities of different variables. After that, we will work with correlations and new variables. More on what will be done will be in subsections 3.4.2 and 3.4.3.

It is important to highlight that even though those two data sets are a part of a multi-fidelity machine learning process we will be considering them alone.

3.4.2 Analysis of TerRA data

In the given Terra data, the following statistical values have been analyzed.

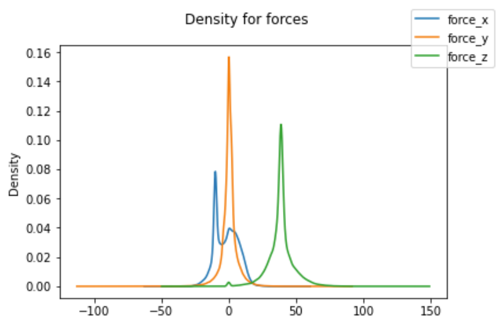
- Mean,
- Standard deviation,
- Minimum,
- Median,
- Maximum.

This analysis resulted in table 3.1, it helped detect variables that had zero or close to zero variance, meaning they would not be of much help in the models. Hence we decided to eliminate: coordinate_y, coordinate_z, velocity_y, angular_velocity_x, angular_velocity_z, and gravity_y from further work.

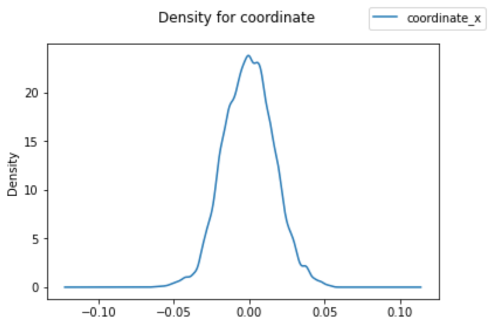
Table 3.1: Show of positional statistics for TerRA data.

	force_x	force_y	force_z
mean	-1.68	0.42	38.64
std	8.52	5.75	9.48
min	-31.79	-61.60	0.0
max	29.77	40.58	99.43
	torque_x	torque_y	torque_z
mean	0.05	0.21	-0.0
std	0.72	1.06	0.0
min	-7.70	-3.72	-0.0
max	3.97	3.97	0.0
	coordinate_x	coordinate_y	coordinate_z
mean	-0.00	0.0	0.12
std	0.02	0.0	0.0
min	-0.06	-0.0	0.11
max	0.05	-0.0	0.14
	velocity_x	velocity_y	velocity_z
mean	0.11	0.0	0.0
std	0.08	0.0	0.01
min	-0.07	0.0	-0.04
max	0.31	0.0	0.05
	anqular_velocity_x	anqular_velocity_y	anqular_velocity_z
mean	0.0	0.98	0.0
std	0.0	0.67	0.0
min	0.0	0.0	0.0
max	0.0	2.0	0.0
	gravity_x	gravity_y	gravity_z
mean	0.00	0.0	-0.99
std	0.14	0.0	0.01
min	-0.42	0.0	-1.0
max	0.47	0.0	-0.88

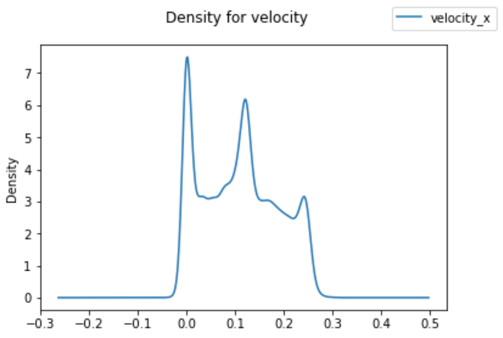
The next step was to visualize the distributions of all the chosen variables, it is presented in figure 3.2. We opted for density plots, as they depict the tendencies in the data in a good manner.



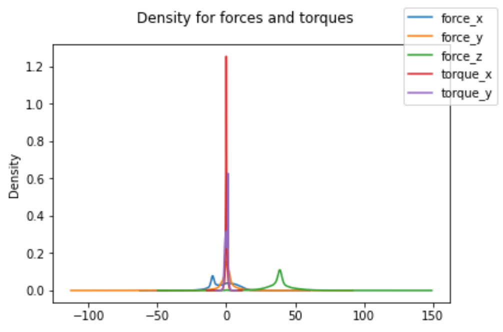
(a) Density for forces.



(b) Density for coordinates.



(c) Density for velocities.



(d) Density for forces and torques.

Figure 3.2: The result of density analysis for TerRA data.

From 3.2 subplot we can see that there is a lot of change in all three of the forces. Both in velocity and coordinate variables the x coordinate is spread out. The last subplot d is placed here to show why we decided to opt out of working with torques, they are too similar to the forces for us to get meaningful results. Keeping them in the training dataset may lead to overfitting.

For further analysis of the relationships between variables, we decided to do a correlation analysis of the remaining dataset. We present the results in figure 3.3.

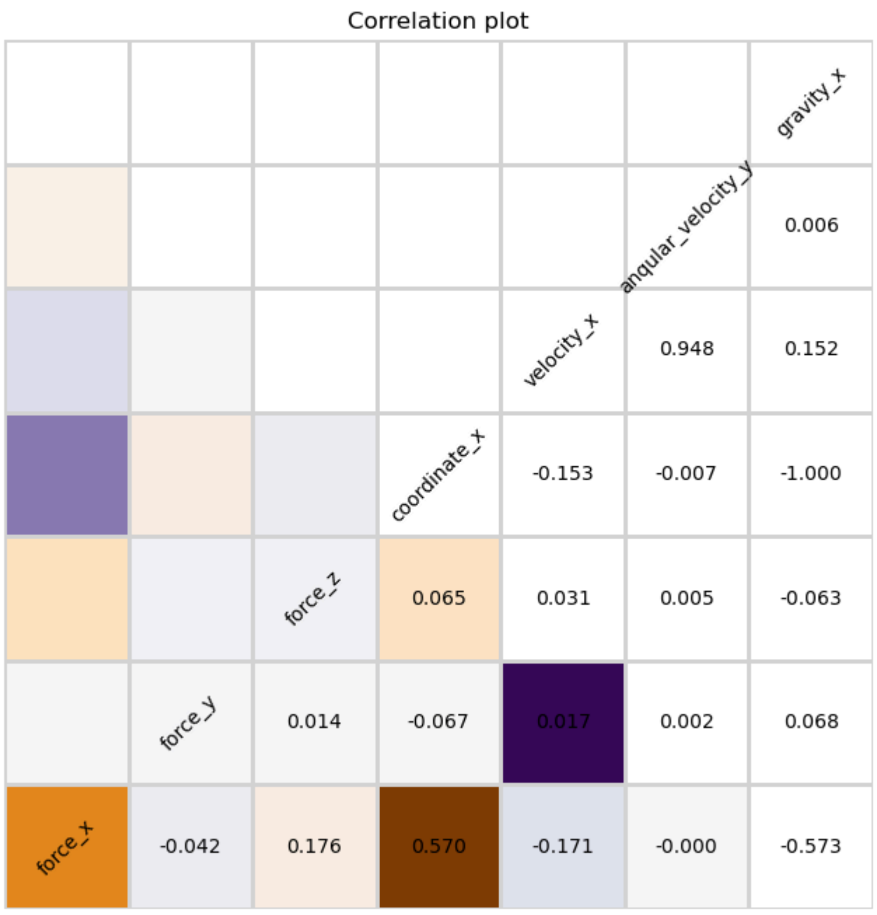
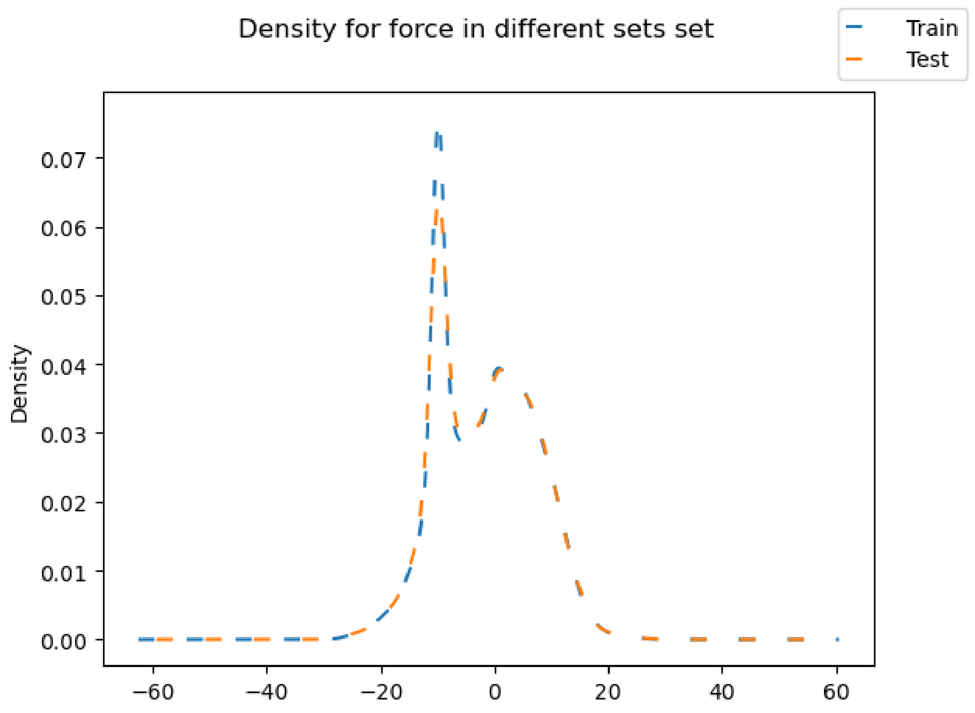


Figure 3.3: Show of correlation plot for TerRA data.

This allowed us to decide on the final set of features we will choose from this level of fidelity. Those are: coordinate_x, velocity_x, velocity_z, angular_velocity_y, gravity_x, gravity_z.

The next thing we should check is if, after the division of data, the distribution of the target variable is the same in different sets. In our case, the similarity must be there. We have three datasets: training, validation, and testing. We will check only for the distribution of training and validation as the test dataset should be something that imitates new data, so it has to be unseen. In figure 3.4 we can see that in fact, the densities are very similar, so it can point us to the conclusion that the task is well formed. There is a slight difference in the two plots, for example in the height of the peak, but it is negligible.

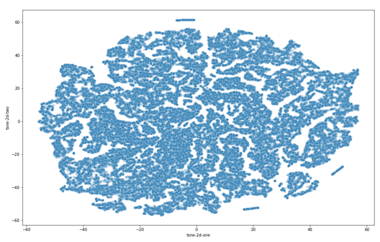


(a) Density for target variable in train and test sets

Figure 3.4: The result of comparison of target variable density for TerRA data.

In this case, it means that the model will operate on a similar dataset that it was trained on. It is not always the case and not always the goal, but in our case, it is both.

The next step is to look for some dimension reduction. Firstly we tried t-SNE to see if there is some hidden structure in the data. The results of 2d and 3d efforts can be seen in figure 3.5. We can see that there are no subgroups in the data that we were able to find out. Both of those attempts were done on 1000 iterations and considered 40 nearest neighbors in the calculation of the loss function.


(a) 2D t-SNE for TerRA data.

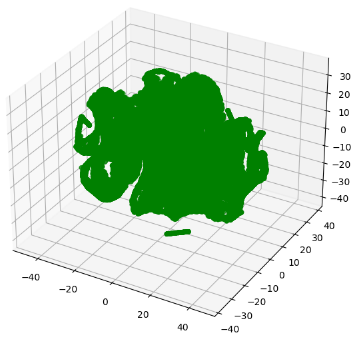

(b) 3D t-SNE for TerRA data.

Figure 3.5: The result of t-SNE analysis on TerRA data.

After consideration, we decided to try and reduce the amount of data that we give to the model without reducing the amount of information, so the last step is to do a principal component analysis on the dataset. In the previous step, we selected six variables to be the input of our models. Those variables are: coordinate_x, velocity_x, velocity_z, angular_velocity_y, gravity_x, gravity_z.

After running the algorithm on the training set without the target variables we get the following results, presented in figure 3.6.

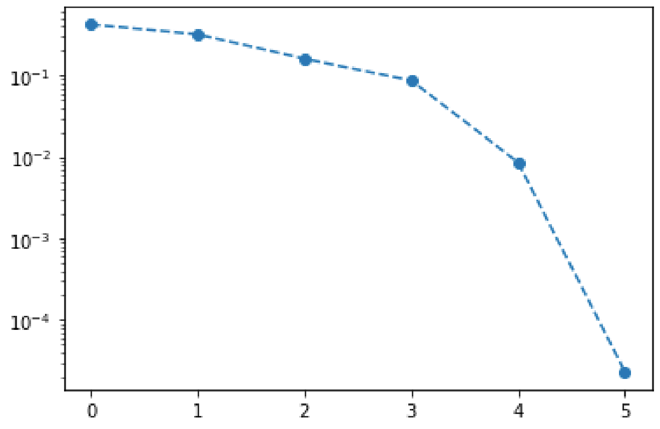


Figure 3.6: Semilogy plot of variance explained by principal components in TerRA data.

From a closer examination we can get that the percentage of explained variance in the data is the following:

- 90,43% - if we take the first three principal components,
- 99,14% - if we take the first four principal components,

- 99,99% - if we take the first five principal components.

This observation suggests that it has the sense to try working with principal components, as we drastically change the size of the data, but we keep almost all of the information. We will later try to work on the version with three and four components.

3.4.3 Analysis of SCM data

In the given SCM data, the following statistical values have been analyzed.

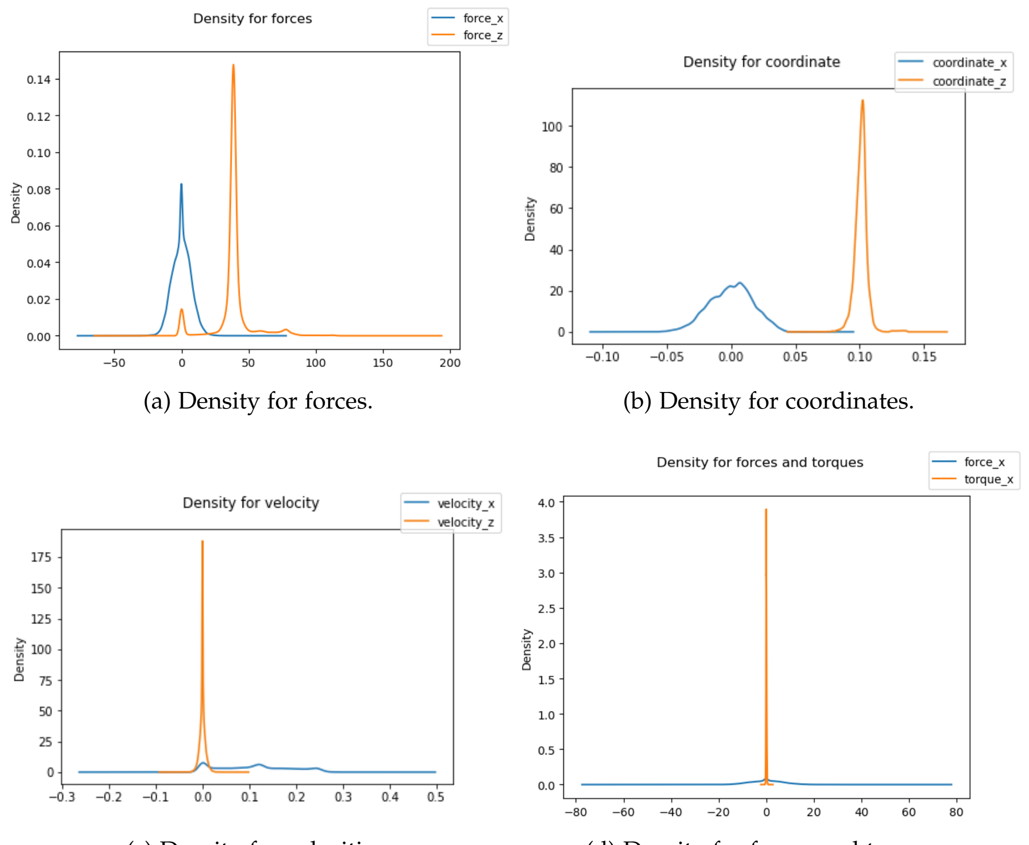
- Mean,
- Standard deviation,
- Minimum,
- Median,
- Maximum.

This analysis resulted in table 3.2, it helped detect variables that had zero or close to zero variance, meaning they would not be of much help in the models. Hence we decided to eliminate: coordinate_y, velocity_y, angular_velocity_x, angular_velocity_z, and gravity_y from further work.

Table 3.2: Show of positional statistics for SCM data.

	force_x	force_y	force_z
mean	0.16	0.68	38.78
std	6.84	0.33	13.96
min	-38.60	-5.07	-0.21
max	39.07	14.05	120.09
	torque_x	torque_y	torque_z
mean	0.08	-1.11	0.02
std	0.14	1.00	0.02
min	-0.96	-5.64	-0.48
max	1.56	4.03	0.39
	coordinate_x	coordinate_y	coordinate_z
mean	-0.00	0.0	0.10
std	0.02	0.0	0.1
min	-0.06	-0.0	0.07
max	0.04	-0.0	0.14
	velocity_x	velocity_y	velocity_z
mean	0.11	0.0	0.0
std	0.08	0.0	0.02
min	-0.08	-0.1	-0.24
max	0.33	0.04	0.09
	anqular_velocity_x	anqular_velocity_y	anqular_velocity_z
mean	0.0	0.98	0.0
std	0.0	0.67	0.0
min	0.0	0.0	0.0
max	0.0	2.0	0.0
	gravity_x	gravity_y	gravity_z
mean	-0.00	0.0	-0.99
std	0.16	0.0	0.02
min	-0.47	0.0	-1.0
max	0.47	0.0	-0.88

The next step was to visualize the distributions of all the chosen variables, it is presented in figure 3.7. We opted for density plots, as they depict the tendencies in the data in a good manner.



(a) Density for forces. (b) Density for coordinates.

(c) Density for velocities. (d) Density for forces and torques.

Figure 3.7: The result of density analysis for SCM data.

From 3.7 subplot we can see that there is a lot of change in some of the three forces compared to others. Both in velocity and coordinate variables the x coordinate is spread out, but the y coordinate has visible change also. The last subplot d is placed here to show why we decided to opt out of working with torques, they are too similar to the forces for us to get meaningful results. Keeping them in the training dataset may lead to overfitting.

For further analysis of the relationships between variables, we decided to do a correlation analysis of the remaining dataset. We present the results in figure 3.8.

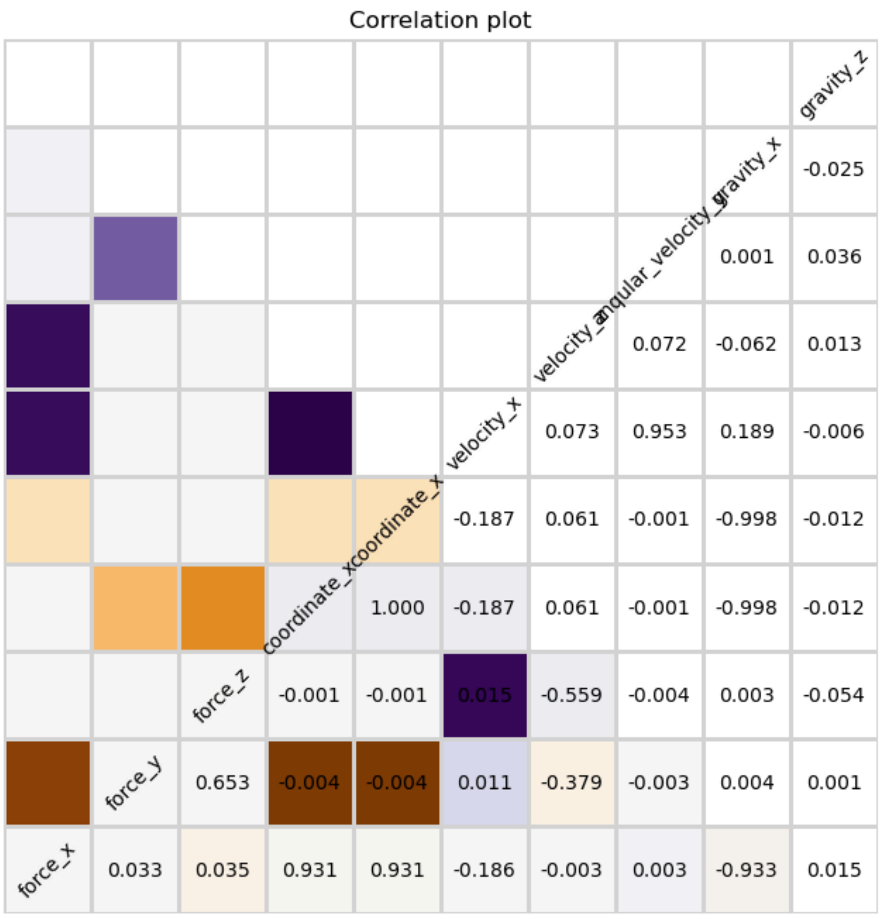
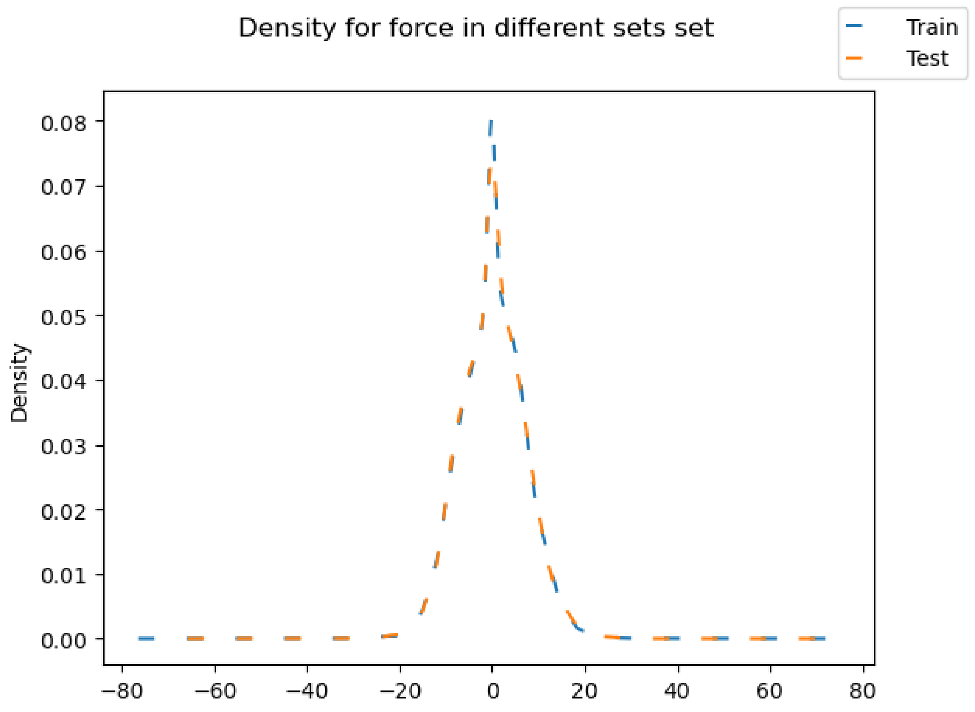


Figure 3.8: Show of correlation plot for SCM data.

This allowed us to decide on the final set of features we will choose from this level of fidelity. Those are: coordinate_x, coordinate_z, velocity_x, velocity_z, angular_velocity_y, gravity_x, gravity_z.

The next thing we should check is if, after the division of data, the distribution of the target variable is the same in different sets. In our case, the similarity must be there. We have three datasets: training, validation, and testing. We will check only for the distribution of training and validation as the test dataset should be something that imitates new data, so it has to be unseen. In figure 3.9 we can see that in fact, the densities are very similar, so it can point us to the conclusion that the task is well formed. There is a slight difference in the two plots, for example in the height of the peak, but it is negligible.

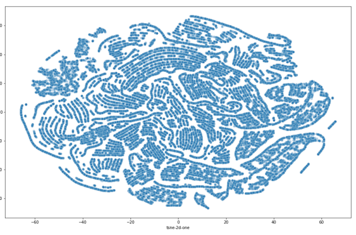


(a) Density for target variable in train and test sets.

Figure 3.9: The result of comparison of target variable density for SCM data.

In this case, it means that the model will operate on a similar dataset that it was trained on. It is not always the case and not always the goal, but in our case, it is both.

The next step is to look for some dimension reduction. Firstly we tried t-SNE to see if there is some hidden structure in the data. The results of 2d and 3d efforts can be seen in figure 3.10. We can see that there are no clear subgroups in the data that we were able to find out, but some structure can be seen. Both of those attempts were done on 1000 iterations and considered 40 nearest neighbors in the calculation of the loss function.


(a) 2D t-SNE for SCM data.

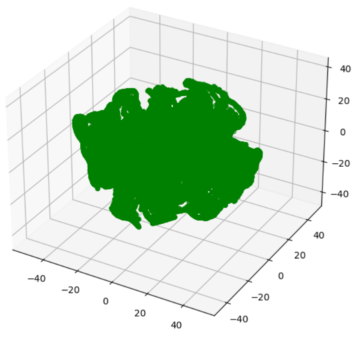

(b) 3D t-SNE for SCM data.

Figure 3.10: The result of t-SNE analysis on SCM data.

After consideration, we decided to try and reduce the amount of data that we give to the model without reducing the amount of information, so the last step is to do a principal component analysis on the dataset. In the previous step, we selected seven variables to be the input of our models. Those variables are: coordinate_x, coordinate_z, velocity_x, velocity_z, angular_velocity_y, gravity_x, gravity_z.

After running the algorithm on the training set without the target variables we get the following results, presented in figure 3.11.

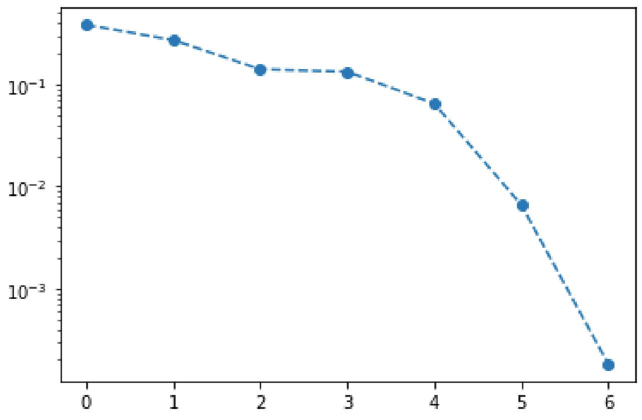


Figure 3.11: Semilogy plot of variance explained by principal components in SCM data.

From a closer examination we can get that the percentage of explained variance in the data is the following:

- 79,53% - if we take the first three principal components,
- 92,85% - if we take the first four principal components,

- 99,32% - if we take the first five principal components.

This observation suggests that it has the sense to try working with principal components, as we drastically change the size of the data, but we keep almost all of the information. We will later try to work on the version with four components as it provides a reduction in size and not much loss of information in contrast to three losing out a lot of variation and five not reducing the size that much.

3.5 Approach

In this section, we will discuss the reasoning and mathematical background behind all the approaches that will be used to find the optimal kernels.

3.5.1 Halving the interpolation points

An interesting understanding of machine learning, which states that we can view the process of learning as estimating an unknown function using data as interpolation points is shown in [31]. A method in statistics called kriging assumes that we have to specify the kernel before action. This approach showed a lot of promise and gives good results, but the problem is to select the correct kernel for a given data. To tackle this problem our work looks for a numerical solution to selecting a suitable kernel.

The goal of this research is to find a way for identifying a kernel that is a good fit for a given dataset. To try and see if a testes kernel is relevant, we propose a method that halves the number of interpolation points and compares the performance before and after that. If the performance is similar in both cases then the kernel can be deemed effective.

This approach gives us a way for somewhat systematic testing of all the possible kernels to find the best one for a problem. It can lead to saving time and computation power when we compare it to a simple experiential method of checking each one on a full model. One more important perk of this is that it can show us some properties and applications of certain kernels.

For the process of determining if the kernel is valid in our case, we will use Gaussian process regression. We will estimate the validity of the kernel by the performance of my Gaussian process regression on datasets, using root mean squared error and mean absolute error. Firstly we create and train based on the whole dataset, after that, we repeat the steps with half the data, and both times we use the same kernel. Once we have both models, we test their performances and compare if they are close. If they are, we deem the kernel valid.

Once we find a kernel that we think performs well on only half of the data we can assume that it will do well on all the data. This process can be repeated with multiple kernels to select a subset of those that are working well. After that, we can even create combinations and test them further with the same approach. This way of working gives way to a lot of possibilities that can speed up the process of creating a sufficient regression model.

The way we will be removing points is at random, to make it as general as possible.

3.5.2 Comparing kernel evaluations of principal components to target function

We performed a principal component analysis in the exploratory data analysis. As mentioned before it worked as intended with a sufficient reduction of dimensions, because of that we decided to use the resulting principal components in the following way. We take each one, transform them with kernel function, and compare the results to the target variable.

At the start of this method, we will select the principal components that explain a certain portion of the variance in the dataset that we have agreed upon. We can determine the appropriate number of principal components to use by looking at the scree plot or by setting a threshold for the percentage of variance we want to explain. We decided to set it to a threshold of 80%.

Using the technique explained above we will have selected the principal components. With them, we will evaluate how they are transformed. One by one we will go through principal components and transform them by using the kernel function $K(x, 0)$, after that we compare it to the target variable. We decided to use the root means squared error. This will be done for all the kernels.

For available kernels in the package that we will be using testing will be done to find the ones that fit the best. We will use different types of kernels. Using the simple comparison to the target variable with the metric of root mean squared error between points, we will be able to identify which kernels are performing well.

One more thing that we can do is that if for different principal components, different kernels will be deemed the best, we can try to create combinations. This could be a simple mechanism for performance improvement. For example, if kernel A performs best for the first principal component, and kernel B performs best for the second principal component, we can try combining them using additivity to create a new kernel.

This approach will enable us to identify all of the best kernel functions, but also unveil some information about the data that we have, that may have been unknown beforehand. We need to remember that the task of choosing the correct kernel is not that straightforward and can be hard for certain datasets. It means that a situation in which non of the basic kernels are good may happen. In such a situation this method may not be that good.

3.5.3 Greedy approach with information criteria

We decided to use an approach that is presented in [32]. It is based on the information criterion, to be more precise on the Bayesian information criterion. We will be using this as a determinant of how good a model is, so basically how goodly the kernel fits into the data. This approach is concentrated on balancing the complexity of the model and its performance. It means that we are punishing the model for being too complex in terms of a number of its parameters. It will also help with the problem of overfitting and will enable a more robust model. The resulting final model will then be both accurate and not too big, it may allow it to capture the underlying relations and not adjust to the training set.

This approach is based on a tool of a criterion. This criterion takes into account the model performance, but also looks at the complexity and the size of the training set, as factors that if are bigger are not advisable. There is more than one criterion. We decided to use this one, as we see it as the best solution here because the data is close to the assumptions made in this method.

The BIC 3.3 is defined by equation:

$$BIC = 2ln(L) + kln(n), \tag{3.3}$$

where:

- $\ln(L)$ - the natural logarithm of the maximum likelihood,
- k - the number of parameters in the model
- n - the number of data points.

The formula is composed of two things. The first one is the punishment for being off with the predictions, the second one is the punishment for having too many parameters. Generally if one goes up the second goes down and vice versa.

This criterion is punishing our model for having too many parameters in comparison to the data, it is understandable as it may lead to overfitting. It also gives a nice measurable way of comparing models based on something more than just performance, the true goodness of a model with taking the dataset into account. We aim to minimize the criterion, as it provides the balance between the performance and the model complexity while looking at both things.

One of the reasons we decided to choose this criterion over for example the Akaike criterion is the stronger punishment for the complexity. It results in the final model being potentially simpler, so less training time, which can be good in our situation since we are looking for the best kernel and not the best model.

3.6 Implementation

3.6.1 Halving the interpolation points

This subsection of the thesis is dedicated to the implementation of the idea of halving the training points and seeing if it diminishes the performance significantly or not. This approach is a generalization of an idea presented in an earlier-mentioned paper. We have described the algorithmic approach behind it, now we can look at how this methodology works in real-world scenarios.

The implementation is based on the *sklearn.gaussian_process* library and assistance of standard libraries like: *numpy*, *pandas*, and *math* [33, 34, 35, 36]. In algorithm 1 we can see a pseudocode of our algorithm.

Algorithm 1 Algorithm based on halving training data.

```
Require: Checking if kernels are fitted to data $X$
Input: input features $X \in \mathcal{R}^{n \times m}$, target $y \in \mathcal{R}^n$
Output: Information about which kernel is fit according to this method
 1: kernels = list_of_all_kernels()
 2: scores_full, scores_half = []
 3: X_half, y_half = random_half(x), random_half(y) = []
 4: for kernel in kernels do
 5:     gpr_full = create_and_fit_model(X, y, kernel)
 6:     gpr_half = create_and_fit_model(X, y, kernel)
 7:     add_scores(scores_full, scores_half)
 8: end for
 9: for i in range(length(kernels) do
10:     if score_half[i].is_close(scores_full[i]) then
11:         print(kernels[i] is within the 80% confidence interval)
12:     else
13:         print(kernels[i] is NOT within the 80% confidence interval)
14:     end if
15: end for
```

We first create a list of all possible kernels from the package, that we will later use to iterate over. After that, we create placeholders for the results. Then iterating over all the kernels we fit the models with all the data and half the data. Testing and saving the results are done later. Once we go through all the kernels we can test if the closeness of the result is there and show which kernels are deemed suitable by this algorithm.

3.6.2 Comparing kernel evaluations of principal components to target function

This subsection of the thesis is dedicated to the implementation of the comparison of the principal components transformed by the kernel functions to the target variable to find a better kernel. We have described the algorithmic approach behind it, now we can look at how this methodology works in real-world scenarios.

We will have to implement our version of the kernel functions that are suitable for our idea. After that, we will be able to search. In algorithm 2 we can see a pseudocode of our algorithm.

Algorithm 2 Algorithm based on principal components compared to target.

```
Require: Looking for the kernels that best transform the PCs to target $X$
Input: input features $X \in \mathcal{R}^{n \times m}$, target $y \in \mathcal{R}^n$
Output: results for each kernel and principal component
 1: function_dict = create_dictionary(make_kernel_functions())
 2: target_column = 'force_x'
 3: results =
 4: for function_name, function in function_dict.items() do
 5:     for column in X.columns do
 6:         if column != target_column then
 7:             answer = evalaute_kernel_function(column)
 8:             rmse = compare(answer, target)
 9:             results[(function_name, column)] = rmse
10:         end if
11:     end for
12: end for
13: results = transform(pd.DataFrame(results))
14: list_of_best_kernels = []
15: for column in results.columns do
16:     best_kernel = where_min(results[column])
17:     list_of_best_kernels.append(best_kernel)
18: end for
19: list_of_best_kernels = [kernel[1] for kernel in list_of_best_kernels]
20: return results, list_of_best_kernels
```

3.6.3 Greedy approach with information criteria

This subsection of the thesis is dedicated to the practical implementation of the greedy Bayesian Information Criterion search for the optimal kernel. In the previous sections, we have discussed the theoretical foundations of this approach and its potential advantages in the context of machine learning and data analysis. Now, it's time to demonstrate how this methodology can be put into practice in real-world scenarios.

To achieve this goal, we will delve into the technical details of the search process and provide a comprehensive overview of the implementation process. We will showcase the steps involved in executing the algorithm, including the tools and technologies used. In algorithm 3 we can see a pseudocode of our algorithm.

Algorithm 3 Algorithm based on greedy information criteria search.

```
Require: Looking for the kernel with the best information criteria $X$
Input: input features $X \in \mathcal{R}^{n \times m}$, target $y \in \mathcal{R}^n$
Output: model with the best kernel according to information criteria
 1: kernels = list_of_all_kernels()
 2: number_of_kernels = length(kernels)
 3: gpr = create_and_fit_model(X, y, kernels)
 4: bic = calculate_bic(gpr)
 5: while TRUE do
 6:     bics = []
 7:     for i in range(number_of_kernels) do
 8:         new_kernels = kernels[:i] + kernels[i+1:]
 9:         new_kernel = sum(new_kernels)
10:         gpr = create_and_fit_model(X, y, new_kernel)
11:         bic = calculate_bic(gpr)
12:         bics.append(bic)
13:     end for
14:     min_idx = where_minimum(bics)
15:     if bics[min_idx] < bic then
16:         kernels = kernels.remove(min_idx)
17:         number_of_kernels -= 1
18:         bic = bics[min_idx]
19:     else
20:         break
21:     end if
22: end while
23: final_kernels = kernels
24: final_kernel = sum(final_kernels)
25: final_gpr = create_and_fit_model(X, y, final_kernel)
26: return final_gpr
```

3.7 Experiments

3.7.1 Halving the interpolation points

In this section, we will look into the experiments done to test the halving of the training dataset. The compared elements will be performed based on the root mean squared error, mean absolute error, and time of training. We will compare the performance of three models:

1. Base model - model with a standard kernel and no hyperparameter tuning,
2. Found model - model with the kernel that has been found by our algorithm and no hyperparameter tuning,
3. Final model - model with the kernel that has been found by our algorithm and tuned hyperparameters.

Terramechanics for Real-time Application

As we remember from earlier the data was divided into three subsets: train, validation, and test. The first one was used for training and searching purposes. The second one was used for the hyperparameter tuning. The last one was used only once at the end to verify the performance and calculate the statistics. Results are shown in table 3.3.

Table 3.3: Experiment result for halving train data approach for TerRA data.

	RMSE	MAE	Training time [s]
Basic model	2.181	1.168	76.5
Found model	2.292e-04	1.736e-03	98.7
Final model	1.691e-05	7.281e-04	125.4

Soil Contact Model

In this case, the data was also divided into three subsets: train, validation, and test too. The first one was used for training and searching purposes. The second one was used for the hyperparameter tuning. The last one was used only once at the end to verify the performance and calculate the statistics. Results are shown in table 3.4.

Table 3.4: Experiment result for halving train data approach for SCM data.

	RMSE	MAE	Training time [s]
Basic model	47.63	24.46	172.1
Found model	1.846	1.196	215.8
Final model	1.491	1.037	279.5

3.7.2 Comparing kernel evaluations of principal components to target function

In this section, we will look into the experiments done to test the comparison approach. We will once more follow all the steps as in the last cases. We will also compare the performance of the three models as before. Those three are:

1. Base model - model with a standard kernel and no hyperparameter tuning,
2. Found model - model with the kernel that has been found by our algorithm and no hyperparameter tuning,
3. Final model - model with the kernel that has been found by our algorithm and tuned hyperparameters.

Terramechanics for Real-time Application

The setup in terms of data division and the purpose of each set is the same as in section 3.7.1. Results are shown in table 3.5.

Table 3.5: Experiment result for comparison approach for TerRA data.

	RMSE	MAE	Training time [s]
Basic model	2.685	1.426	78.3
Found model	7.849e-04	5.901e-04	101.7
Final model	2.414e-05	1.8511e-05	121.9

Soil Contact Model

The setup in terms of data division and the purpose of each set is the same as in section 3.7.1. Results are shown in table 3.6.

Table 3.6: Experiment result for comparison approach for SCM data.

	RMSE	MAE	Training time [s]
Basic model	51.85	11.39	154.5
Found model	2.009	1.378	196.8
Final model	1.958	1.341	250.1

3.7.3 Greedy approach with information criteria

In this section, we will look into the experiments done to test the greedy search algorithm implementation. We will once more follow all the steps as in the last cases. We will also compare the performance of the three models as before. Those three are:

1. Base model - model with a standard kernel and no hyperparameter tuning,
2. Found model - model with the kernel that has been found by our algorithm and no hyperparameter tuning,
3. Final model - model with the kernel that has been found by our algorithm and tuned hyperparameters.

Terramechanics for Real-time Application

The setup in terms of data division and the purpose of each set is the same as in section 3.7.1. Results are shown in table 3.7.

Table 3.7: Experiment result for greedy search for TerRA data.

	RMSE	MAE	Training time [s]
Basic model	3.124	1.614	154.5
Found model	1.203e-05	8.865e-05	196.8
Final model	1.101e-06	8.071e-06	250.1

Soil Contact Model

The setup in terms of data division and the purpose of each set is the same as in section 3.7.1. Results are shown in table 3.8.

Table 3.8: Experiment result for greedy search for SCM data.

	RMSE	MAE	Training time [s]
Basic model	42.13	25.03	185.9
Found model	1.256	1.069	231.5
Final model	0.972	0.841	286.1

3.8 Comparison of the results

3.8.1 Explicit results of algorithms

In this section we will present what each algorithm returns, so it will be a final kernel. The fact that the methods have different builds, sometimes it will be a single kernel, and sometimes a kernel composed of a sum of basic kernels. This comparison will be divided into the two datasets that we have, as those were two different sets of experiments.

Terramechanics for Real-time Application

In table 3.9 we can see the results given by all the methods. They are in the form of a parameter that is given into the Python class and interpretation.

Table 3.9: Kernels found by the methods for the TerRA data.

Method	Result	Interpretation
Halving train data	DotProduct() + WhiteKernel()	Dot product kernel
Comparing PC's	DotProduct() + WhiteKernel()	Dot product kernel
Greedy	RBF() + DotProduct() + WhiteKernel()	Dot product and RBF kernel

We can see that the two methods returned a kernel that is a dot product. We know that the dot product kernel measures the similarity between the two input vectors by calculating their dot product of them. The resulting value tells us how similar the two input vectors are to each other. When we use the dot product kernel in Gaussian process regression, we end up with the resulting covariance matrix as typically diagonal one, meaning that the correlation between different input dimensions is not taken into account that much. This can make the dot product kernel useful in cases of high-dimensional input spaces where the correlations between dimensions are weak or not even there. Which is something we observed a little in our 3.4. While it is a simple kernel that only models linear correlations between input variables, it can be combined with other kernels to capture more complex relationships in the data. It is something worth exploring more.

As we can see from table 3.9 the greedy search found a different solution than the other two methods. The reason behind this is the fact that it is constructed in other ways than the remaining two. The RBF kernel is often a very popular choice in Gaussian process regression as it covers smooth and continuous functions. The combination of the two kernels means that the resulting combination will allow our model to approximate functions that are smooth and non-smooth with different levels of noise.

Soil Contact Model

In table 3.10 we can see the results given by all the methods. They are in the form of a parameter that is given into the Python class and interpretation.

Table 3.10: Kernels found by the methods for the SCM data.

Method	Result	Interpretation
Halving train data	DotProduct() + WhiteKernel()	Dot product kernel
Comparing PC's	Matern() + WhiteKernel()	Metern kernel
Greedy	Matern() + RBF() + WhiteKernel()	Matern and RBF kernel

In this case, we can see that we have three different kernels found. As in the case of the last time, the result of the greedy search is more complex than others. Once again we can see that the result of one algorithm is the dot product. The interpretation of this is the same as before.

A new interesting kernel that was found is the Matern kernel. It gives representation to a big family of a lot of different functions that are defined by two parameters, those are smoothness and length scale. The first one we manipulate if we want to play with the ability to differentiate functions and the second one is responsible for what we call decay. We can increase the smoothness of the function by increasing the first parameter. The second is for the correlation between points, if it is big the correlation will not be. The Matern kernel is generally a good choice for working with spatial data. We may say that it is close to the topic that we are dealing with, so we should not be too surprised for it to come up in the search.

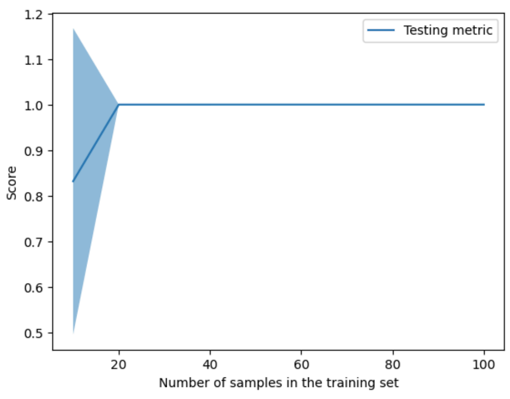
The RBF kernel, which is also often named the Gaussian kernel, is widely used in this field for its smoothness and ability that it has to capture complex patterns. It assigns higher similarity or in other words correlation to data points that are closer to each other and gets lower exponentially as the distance increases just as in Gaussian distribution. On the one hand this kernel is dependent on only a single length scale parameter, which determines the rate of decay, similar to the Matern kernel. On the other hand, the Matern kernel, as mentioned earlier, is a flexible kernel that can capture both smooth and non-smooth variations in the data. It allows for differentiability of high or low orders based on the value of the smoothness parameter. By combining the two kernels, we can use the smoothness and flexibility of both functions at the same time. This new combination kernel can detect complex patterns in the data while also allowing for adaptability to varying length scales. The RBF component contributes its ability to model intricate local relationships, while the Matern component adds the flexibility to capture broader variations and account for different degrees of smoothness. The resulting RBF + Matern kernel provides a powerful tool for analyzing data that exhibits both local and global structures. Its combination of smoothness, adaptability, and flexibility allows it to effectively model a wider range of data.

3.8.2 Impact of train set size on performance

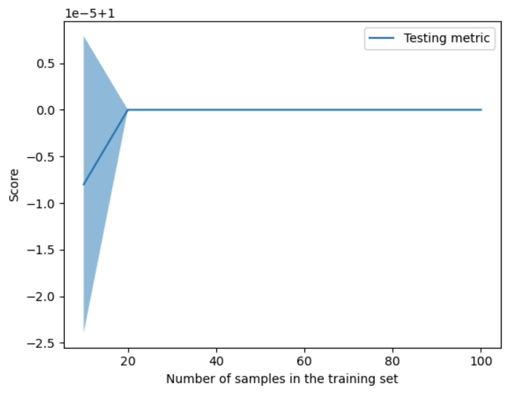
In this section, we will look at how the training sample size affects the performance of our final models. It will help us in many ways understand the importance of data and show if there is a possibility of using fewer data points and achieving the same results.

Terramechanics for Real-time Application

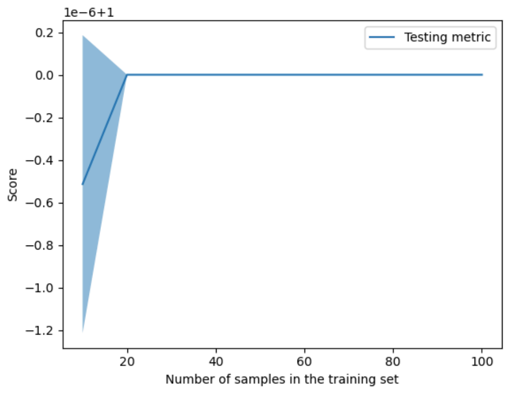
We can see in figure 3.12 that all three models achieved by all three methods are achieving the maximal performance just from a training sample of size 20. It means that in the case of this data, we do not need that much training data to get the best results achievable. The score that is specified in those plots is what part of the performance can be recreated using the given size of a training dataset.



(a) Plot for comparing PCs method.



(b) Plot for halving training data method.

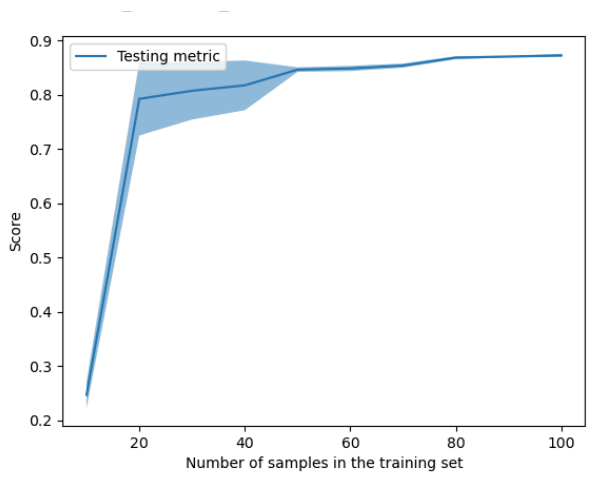


(c) Plot for greedy search method.

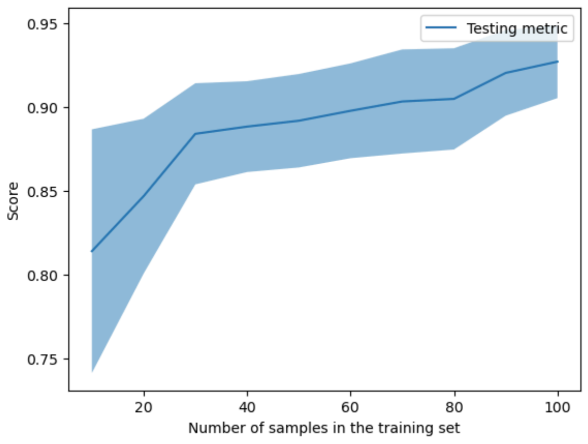
Figure 3.12: Plots of score vs. training data size for TerRA data.

Soil Contact Model

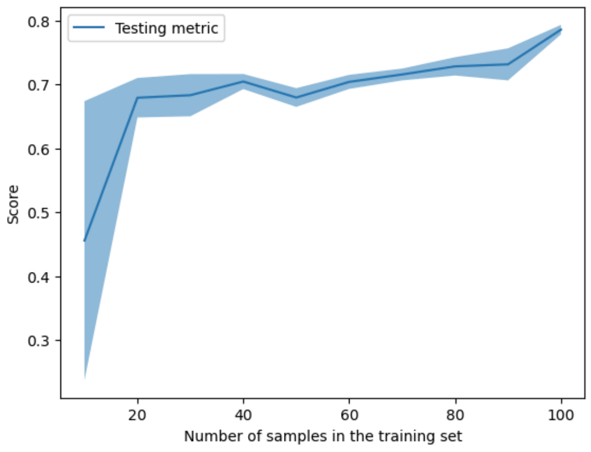
In figure 3.13 we can see that the situation is different than in the case of the figure 3.12. As little as 20 training samples are not enough to get the full performance of the models. It varies from model to model, but non get to full score even with 100 training samples. The score is the same thing as before.



(a) Plot for comparing PCs method.



(b) Plot for halving training data method.



(c) Plot for greedy search method.

Figure 3.13: Plots of score vs. training data size for SCM data.

We know that the model obtained by the greedy search is the most complex one, it is backed up by the fact that the score from the plot c in 3.13 is the lowest. We also observe that there is no plateau as in the case of figure 3.12 and terra data. It proves that the more complex data obtained from a more accurate model needs more computation power than a model to be reasonably predicted.

4 Conclusions

In this project, we have employed a wide range of techniques to enhance the performance of our final model and, more importantly, to identify the most effective method for determining the optimal kernel. Our approach involved several key components, including exploratory data analysis (EDA), principal components analysis (PCA), Gaussian process regressions, and more. Firstly, EDA enabled us to gain valuable insights into the dataset, uncovering patterns, trends, and potential outliers. This process helped us understand the underlying structure of the data, identify important variables, and make informed decisions during model development. Helped us select the correct subset of features for each dataset.

Using principal components analysis to try and reduce the dimensionality of the data even more or try more techniques than we did. Just like PCA worked and t-SNE did not, there may be a technique that will work even better. This step may be useful in cases when we decide we still have too many dimensions, but we suspect that some of them are irrelevant to the target variable prediction, like it was with our data. In our case, it helped reduce the number of dimensions by a lot. By reducing the dimensionality of the data we eliminated irrelevant or redundant features and focused on the more informative ones. This can lead to more efficient and accurate analysis, particularly in cases where the original dataset has a high number of dimensions. It will also tackle the problem of the need for a lot of points to fill a high-dimensional space.

We have compared the results of all three implemented methods in terms of root mean squared error, mean absolute error, and training time. Based on that, we can make assumptions about which one of them worked the best. We tried to perform the experiments in a manner that diminished the influence of randomness on the results. In terms of performance the method that resulted in the lowest error both RMSE and MAE, was the greedy information criterion search. In terms of training and searching time, all three methods had similar results. The visible difference can be caused by the nature of the results of each algorithm, as the greedy search returns a combination of kernels, that can be more than one kernel. This flexibility provides an obvious advantage over the other methods. Despite similar training and searching times across all three methods, it has performance advantages over the remaining methods. Another important observation is that the selection of a relevant kernel is far more important than fitting the hyperparameters of a kernel. We can see that by comparing the difference in the results of all three models that were used in the experiments for each method.

Future work

Many steps can be done further in this work. Firstly in our opinion, the most important step is to incorporate more data into the deliberations. It will allow for more testing of the methods and seeing if they are resulting in relevant conclusions. Secondly maybe using more data can result in improving the performance.

Another thing that can be done is to improve the existing methods in a manner of broadening the search area. For this to be done new kernels would have to be implemented in the *sklearn.gaussian_process.kernels* package or another package would have to be used. If the answer to the shortage of kernels is a usage of another package, for example, *GPy*, then the implementation would have to change but the algorithmic idea behind the methods can stay the same.

One more thing that can be done is to widen the searched subspace of the greedy search algorithm by allowing the exclusion of subsets of more than one at a time. In the present version, the search starts from the full model and kernels are taken away, an option for starting it from the null model can be implemented to have it checked both ways. One last change that can be made is a change of the information criterion from Bayesian information criterion to Akaike information criterion, it takes into account different parameters that may be more important.

List of Figures

List of Tables

Bibliography

[1] N. Drake. *Why we explore Mars and what decades of missions have revealed*. 2020. URL: https://www.nationalgeographic.com/science/article/mars-exploration-article (visited on 05/22/2022).

[2] N. Drake. *What We Learned from the Perseverance Rovers First Year on Mars*. 2020. URL: https://www.scientificamerican.com/article/what-we-learned-from-the-perseverance-rovers-first-year-on-mars/# (visited on 05/22/2022).

[3] M. Guo, A. Manzoni, M. Amendt, P. Conti, and J. S. Hesthaven. "Multi-fidelity regression using artificial neural networks: Efficient approximation of parameter-dependent output quantities". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022), p. 114378. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114378. URL: https://www.sciencedirect.com/science/article/pii/S0045782521006411.

[4] M. Contreras, C.-Y. Peng, D. Wang, and J.-S. Chen. "Determining Wheel-Soil Interaction Loads using a Meshfree Finite Element Approach Assisting Future Missions with Rover Wheel Design". In: Aug. 2012. ISBN: 978-1-62410-183-0. DOI: 10.2514/6.2012-4562.

[5] S. Barthelmes. "TerRA: Terramechanics for Real-time Application". In: *5th Joint International Conference on Multibody System Dynamics*. June 2018. URL: https://elib.dlr.de/121815/.

[6] R. Arvidson, J. Bell, P. Bellutta, N. Cabrol, J. Catalano, J. Cohen, L. Crumpler, D. Des Marais, T. Estlin, W. Farrand, R. Gellert, J. Grant, R. Greenberger, E. Guinness, K. Herkenhoff, J. Herman, K. Iagnemma, J. Johnson, G. Klingelhoefer, and A. 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* 115 (Sept. 2010). DOI: 10.1029/2010JE003633.

[7] R. E. Arvidson, J. W. Ashley, J. F. Bell III, M. Chojnacki, J. Cohen, T. E. Economou, W. H. Farrand, R. Fergason, I. Fleischer, P. Geissler, R. Gellert, M. P. Golombek, J. P. Grotzinger, E. A. Guinness, R. M. Haberle, K. E. Herkenhoff, J. A. Herman, K. D. Iagnemma, B. L. Jolliff, J. R. Johnson, G. Klingelhöfer, A. H. Knoll, A. T. Knudson, R. Li, S. M. McLennan, D. W. Mittlefehldt, R. V. Morris, T. J. Parker, M. S. Rice, C. Schröder, L. A. Soderblom, S. W. Squyres, R. J. Sullivan, and M. J. Wolff. "Opportunity Mars Rover mission: Overview and selected results from Purgatory ripple to traverses to Endeavour crater". In: *Journal of Geophysical Research: Planets* 116.E7 (2011). DOI: https://doi.org/10.1029/2010JE003746. eprint: https://agupubs.onlinelibrary.wiley.

com/doi/pdf/10.1029/2010JE003746. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2010JE003746.

[8] 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. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21647.

[9] M. Maimone, Y. Cheng, and L. Matthies. "Two years of Visual Odometry on the Mars Exploration Rovers". In: *Journal of Field Robotics* 24.3 (2007), pp. 169–186. DOI: https://doi.org/10.1002/rob.20184. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20184. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20184.

[10] J. Y. Wong. *Theory of Ground Vehicles*. John Wiley & Sons, Inc., 2008.

[11] 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*. Oct. 2018. URL: https://elib.dlr.de/121794/.

[12] A. Tasora, D. Mangoni, and D. Negrut. "An Overview of the Chrono Soil Contact Model (SCM) Implementation". In: *Digital Dynamics Lab* (Aug. 2018).

[13] C. F. Gauss. "Theoria motus corporum coelestum". In: *Werke* (1809).

[14] S. TUTORIALS. *Normal Distribution Quick Introduction*. URL: https://www.spss-tutorials.com/normal-distribution/ (visited on 01/25/2023).

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[16] T. E. o. E. Britannica. *inference*. URL: https://www.britannica.com/science/inference-statistics (visited on 01/25/2023).

[17] B. A. Olshausen. "Bayesian probability theory". In: *The Redwood Center for Theoretical Neuroscienc* (Mar. 2004).

[18] S. M. Ross. *STOCHASTIC PROCESSES*. John Wiley & Sons, Inc., 1996.

[19] L. Breuer. *Lecture notes in Introduction to Stochastic Processes*. 2014. URL: https://www.kent.ac.uk/smsas/personal/lb209/files/sp14.pdf (visited on 05/22/2022).

[20] N. Durrande. *Gaussian Proccess Summer School - Kernel Design*. 2014. URL: http://gpss.cc/gpss17/slides/KernelDesign.pdf (visited on 05/22/2022).

[21] R. Alake. *An Introduction To Gradient Descent and Backpropagation In Machine Learning Algorithms*. URL: https://towardsdatascience.com/an-introduction-to-gradient-descent-and-backpropagation-in-machine-learning-algorithms-a14727be70e9 (visited on 01/29/2023).

[22] S. Seitz. *Gradient-based explanations for Gaussian Process regression and classification models*. 2022. DOI: 10.48550/ARXIV.2205.12797. URL: https://arxiv.org/abs/2205.12797.

[23] I.-C. Yeh. "Modeling of Strength of High-Performance Concrete Using Artificial Neural Networks. Cement and Concrete research, 28(12), 1797-1808". In: *Cement and Concrete Research* 28 (Dec. 1998), pp. 1797–1808. DOI: 10.1016/S0008-8846(98)00165-3.

[24] D. Duvenaud. *Automatic Model Construction with Gaussian Processes*. https://github.com/duvenaud/phd-thesis. 2014.

[25] A. B. Abdessalem, N. Dervilis, D. J. Wagg, and K. Worden. "Automatic Kernel Selection for Gaussian Processes Regression with Approximate Bayesian Computation and Sequential Monte Carlo". In: *Frontiers in Built Environment* 3 (2017). ISSN: 2297-3362. DOI: 10.3389/fbuil.2017.00052. URL: https://www.frontiersin.org/articles/10.3389/fbuil.2017.00052.

[26] A. Makiewicz and W. Ratajczak. "Principal components analysis (PCA)". In: *Computers & Geosciences* 19.3 (1993), pp. 303–342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.

[27] M. Ringnér. "What is principal component analysis?" In: *Nature Biotechnology* 26 (Apr. 2008), pp. 303–4. DOI: 10.1038/nbt0308-303.

[28] S. M. Holand. *PRINCIPAL COMPONENTS ANALYSIS (PCA)*. 2019. URL: http://strata.uga.edu/8370/handouts/pcaTutorial.pdf (visited on 05/22/2022).

[29] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[30] 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. URL: https://elib.dlr.de/190039/.

[31] H. Owhadi and G. R. Yoo. "Kernel Flows: From learning kernels from data into the abyss". In: *Journal of Computational Physics* 389 (July 2019), pp. 22–47. DOI: 10.1016/j.jcp.2019.03.040. URL: https://doi.org/10.1016%5C%2Fj.jcp.2019.03.040.

[32] G. Schwarz. "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2 (1978), pp. 461–464. ISSN: 00905364. URL: http://www.jstor.org/stable/2958889 (visited on 04/20/2023).

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[34] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. "Array programming with NumPy". In: *Nature* 585 (2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`.

[35] W. McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56.

[36] G. Van Rossum. *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.