Karlsruher Institut für Technologie

Fakultät für Informatik
Institut für Anthropomatik und Robotik
Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS)
Prof. Dr.-Ing. Uwe D. Hanebeck

Intelligente ISAS Sensor-Aktor-Systeme

Bachelor Thesis

Improved Speed Estimation of BLDC Motors Using Gaussian Processes

Mariana Petrova

31. October 2019

Referent: Prof. Dr.-Ing. Uwe D. Hanebeck

Betreuer: Dipl.-Phys. Jana Mayer M.Sc. Ajit Basarur

Abstract

For high precision control of BLDC motors, the precise angular position and rotational speed of the rotors are required. In previous work, a new concept utilizing a regression model to predict the magnetic field of the rotor was developed and combined with a state-space estimation model. While the estimation of the angular position showed good results, the accuracy of the speed estimation, which is crucial for precise control of the BLDC motors, was not sufficient. Furthermore, due to the task requiring a low prediction time, the inclusion of more training data or considering an increased model complexity to improve the estimation is not straightforward.

The aim of this thesis is to evaluate different approaches for state estimation of BLDC motors. Thereby, a special focus is put on improving the speed estimation. Different variations of linear regression and Gaussian process models were built and combined with an extended Kalman filter. The developed models address the main flaws of previous works in that they consider the periodicity of the angular position and the problem of different input scales. Additionally, an emphasis was put on better utilization of the training data.

As a result, some proposed models are able to work with bigger training sets and therefore incorporate more information, while displaying comparable computational costs. The combination of those aspects allows us to identify several models showing a better state estimation performance than previous works.

Eidesstattliche Erklärung

Hiermit erkläre ich, die vorliegende Bachelor Thesis selbstständig angefertigt zu haben. Die verwendeten Quellen sind im Text gekennzeichnet und im Literaturverzeichnis aufgeführt.

Karlsruhe, 31. October 2019

Mariana Petrova

Contents

List of Figures

List of Tables

Notation

Conventions

x	Scalar
$\underline{x}$	Column vector
$\mathbf{A}$	Matrix
$(.)_k$	Quantity at time step k.
$\mathbb{R}$	Set of real numbers
$\sim$	Distribution operator E.g., $\boldsymbol{x} \sim \mathcal{U}$ means $\boldsymbol{x}$ is distributed according to $\mathcal{U}$.

Probabilistic Regression

$\underline{w}$	Weight vector, parametrizing the linear regression
$k(x, x')$	Kernel function
$\underline{\vartheta}$	Vector, containing all hyperparameters of the kernel
$\mathbf{K}_{NN}$	Covariance matrix of all training inputs
$\mathbf{K}_{*N}$	A vector, constructed by the execution of for a testing point will all training inputs
$\mathbf{K}_{PP}$	Covariance matrix of all pseudo-inputs
$\underline{\bar{x}}$	Pseudo input instance
$\bar{f}$	Pseudo output instance

State Estimation

$\underline{\hat{x}}_k^p$	*A priori* estimation at time step k
$\underline{\hat{x}}_k^e$	*A posteriori* estimation at time step k
$\mathbf{P}_k^p$	*A priori* estimation-error covariance at time step k
$\mathbf{P}_k^e$	*A posteriori* estimation-error covariance at time step k
$\mathbf{A}_k$	System matrix at time step k
$\mathbf{H}_k$	Measurement matrix at time step k for linear systems, *Jacobi*-matrix of the measurement function for non-linear systems
$\underline{h}_k(\underline{x}_k)$	Measurement function at time step k
$\underline{\varepsilon}_k$	System noise
$\underline{\xi}_k$	Measurement noise
$\mathbf{Q}_k$	System noise covariance
$\mathbf{R}_k$	Measurement noise covariance

Abbreviations

GP	Gaussian Process
SPGP	Sparse Gaussian Process using pseudo-inputs
SE	Squared Exponential kernel
SE-ARD	Squared Exponential kernel with Automatic Relevance Determination
PER-ARD	Custom Periodic kernel with Automatic Relevance Determination
KF	Kalman Filter
EKF	Extended Kalman Filter
RMSE	Root Mean Square Error

CHAPTER 1

Introduction

A very important part of the high precision control of BLDC motors is the precise knowledge of its current state, namely the angular position and velocity of the rotor. The permanent magnets inside the BLDC electric motors produce a magnetic field that can be detected at the backside of the motor. From the measurements of this magnetic field, several models that estimate the angular position and the rotational speed were built [1]. The developed models utilize B-Splines and Gaussian processes as regression methods. Both regression models were used in combination with the extended Kalman filter and the unscented Kalman filter to achieve precise rotor state estimation. However, these methods display several drawbacks, which will be addressed in this thesis.

Firstly, the used Gaussian process models do not scale well with the training data. This is an unfavourable characteristic of the model since a great amount of data can be generated by the hardware setup. Secondly, the characteristic of the angular position, namely its periodicity on 360°, is not addressed by the applied method. Therefore physical information is not utilized. Thirdly, the scale of the input data is not properly considered which may cause numerical problems.

In this thesis, we address the aforementioned concerns through several modifications of the regression approach. We first identify three "base" regression methods - linear regression using basis functions, Gaussian processes and sparse Gaussian processes using pseudo-inputs [2]. Subsequently, we build multiple different variations of each "base" model and evaluate their regression performance as well as their estimation performance when combined with an extended Kalman filter. Finally, we identify several models performing better than the previous work [1] in both position and speed estimation, while having similar computational costs.

The remainder of this work is organized as follows, chapter 2 gives an introduction to the hardware background of BLDC motors, as well as state-of-the-art motor feedback solutions. In chapter 3 we present the previously build models. Chap-

ter 4 gives theoretical background on the state estimation task, as well as the most commonly used estimation approach- the Kalman filter. The following chapter 5 presents the concept of probabilistic regression. There we describe (Bayesian) linear regression and Gaussian processes, as well as a sparse modification of the Gaussian processes, based on [2]. In chapter 6 we present our data and how it is processed. Afterwards, we describe the different regression models we construct and how they are integrated into the extended Kalman filter. We then propose an alternative approach for speed estimation, which uses the median filter based on the results of the extended Kalman filter. In chapter 7 we evaluate our regression models and analyze the overall performance of the state estimation using these regression models. We conclude with chapter 8, where we also point out directions for further research.

CHAPTER 2

Hardware background

2.1 Overview of BLDC Motors

Brushless direct current (BLDC) motors are synchronous electric motors that are rapidly gaining popularity in recent years. As the name suggests, these motors do not use brushes for mechanical communication, but are electronically controlled instead. The two main parts of a BLDC motor are the rotor and the stator. The rotor consists of a permanent magnet where the number of North (N) and South (S) pole pairs can vary from two to eight. With electromagnets positioned on the stator, the control of BLDC motors happens via a computer that charges up the electromagnets as the shaft with the permanent magnets turns. An overview of a typical BLDC motor can be seen in Figure 2.1.

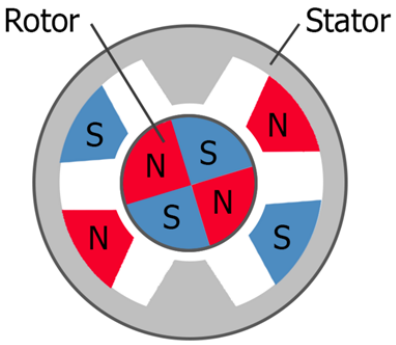


Figure 2.1: BLDC Motor overview

Reliable feedback of the angular position and rotational speed of the rotor is important for the performance of the total motion control system. Generally, any sensing system that can deliver information about the current rotor state can be used to electronically control the BLDC motor.

In the following, we will present an overview of the state-of-art sensor feedback systems for motor control.

2.2 State of the Art

2.2.1 Inbuilt Hall Sensors

One of the most widely employed sensor feedback systems uses the inbuilt Hall-sensors to measure the magnetic field, created inside the motor. Each Hall sensor produces either a High-signal whenever a south pole passes by, or a Low signal for a north pole [3]. The current rotor position is determined based on the combination of the sensor signals. The Hall sensors feedback system has the key disadvantage of low feedback accuracy feedback with a resolution of 60°. This drawback does not affect the control performance significantly by high rotational speeds but proves to have crucial negative effects for low-speed commutation. A feedback system using the Hall sensors is suitable in cases when high accuracy for low-speed rotations is not required, but the overall setup cost is budget sensitive.

2.2.2 Magnetic Resolver

Similarly to the inbuilt Hall sensors, the magnetic resolver measures the magnetic field of the permanent magnets on the rotor. However, in the case of the magnetic resolver, the magnetic field is measured outside the motor by four quadratic patterned Hall sensors. In [4] it is derived, how the signals of the different Hall sensors can be used to calculate the current rotation of the rotor.

2.2.3 Optical Encoder

Another possible way to provide feedback to the BLDC motor is by attaching an optical encoder unit. The optical encoder identifies the angular position using a light source, light detectors, and an optical grating. The light from the light source either passes the code disc or is blocked. The angular position is then determined by observing two identical bright-dark-patterns on the disc, which are placed with a known shift from one another. This system is advantageous in scenarios when very high precision dynamic positioning is required and the application is not cost-sensitive. Nevertheless, mounting an optical encoder to the shaft of the motor is not always a feasible solution e.g. in cases of space and size limitations.

2.2.4 Speed Measurement Through Stray Magnetic Field Sensing

As mentioned above, the rotation of the permanent magnets inside the rotor creates a magnetic field. In [5] a feedback system that uses measurements of the external

stray magnetic field in order to determine the current rotational speed of the rotor is presented. As the rotor changes position, the stray leakage field rotates at the same frequency and phase angle. The dependency between the rotational speed and the magnetic field can then be described as

$$\omega = \frac{60 f_{st}}{p} RPM,$$

where f_{st} is the rotating frequency of the leakage field in Hz and p defines the number of magnetic field pairs. In the paper, a tunnelling magnetoresistive (TMR) sensor is chosen to measure the field due to its high sensitivity, large operating range, and easy installation. The sensor is then mounted outside the motor yoke.

CHAPTER 3

Previous work

As an accurate feedback system is essential for the precise control of BLDC motors, methods for precise measurement and/or estimation of the rotor state are in demand. In [1] models for estimating the angular position and the angular velocity of BLDC motors were built. All of the models rely on measuring the magnetic field generated by the permanent magnets to acquire information about the rotor state. This estimation approach presents an alternative to the direct position measurement, which could be either inaccurate in the case of Hall-sensors or cost-inefficient in the case of optical encoders.

Next, we will present the hardware setup constructed in [1], followed by an overview of the estimation models.

3.1 Hardware Setup Configuration

An essential component of the hardware setup is the magnetic sensor that measures the magnetic flux density. The most important requirements by choosing such a sensor are high measuring accuracy, high sampling rate and the ability of the sensor to measure the field in several directions in space. In [1] two different sensors were selected and tested: the ASD2613-R TinyShield 3-axis compass and the Honeywell 2-axis magnetoresistive sensor HMC1052L-TR [6]. Since the sampling rate of the former sensor is too small for state estimation at higher rotational speeds of the motor, the latter sensor was chosen for the final models. As the magnetic sensor measures the magnetic field at a stationary position, its location needs to be chosen carefully. In [1] multiple placements of the magnetic sensor are considered, whereby a higher signal-to-noise ratio is prioritized. In the final setup, the sensor is placed on the backside of the motor, with 30mm axial distance and 12mm radial distance. Additionally, a high accuracy reference sensor is needed to measure the ground truth position of the rotor. The TMCS-2 optical incremental encoder is chosen for

this purpose. The two sensors are connected to a microcontroller board - Arduino Duo, that transmits the measurements to a computer. A complete diagram of the hardware setup used for the experiments is given in Figure 3.1.

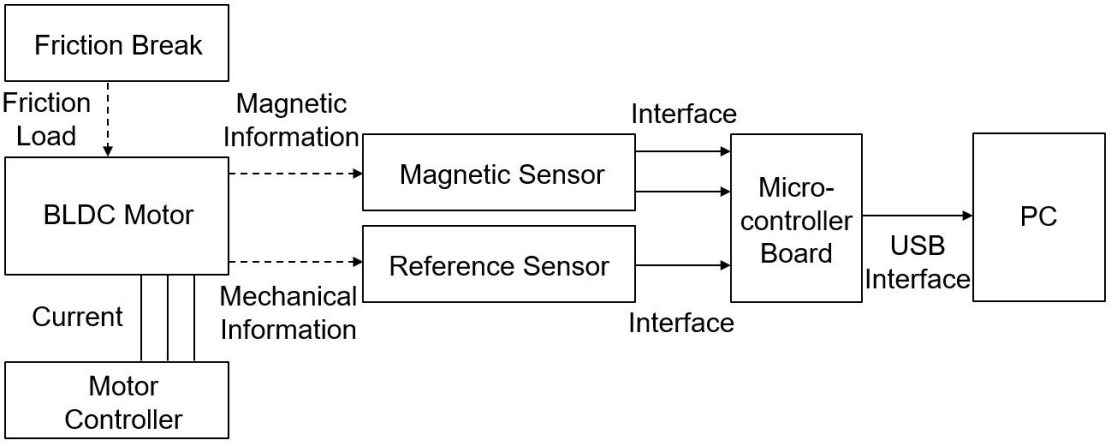


Figure 3.1: Hardware Setup [1].

3.2 State Estimation Setup

Thus far, we described the hardware setup, build in [1]. Following, one wants to create a model that uses the measurements of the magnetic field to estimate the state of the rotor. An overview of the entire estimation model is shown in Figure 3.2.

The experimental setup is used to generate data containing both the measurements of the magnetic field, as well as the true state of the rotor.

Firstly, we would need to identify the so-called measurement function, mapping the rotor state to the magnetic field measurements. Therefore, a part of the generated data is used for this purpose. Preliminary tests had shown that the observed dependence between the rotor state and the magnetic field is non-linear. To find the underlying non-linear relationship between them, [1] introduces Gaussian process and B-Splines as two possible approaches.

To estimate the current rotor state (angular position and rotational speed), a recursive estimation approach, namely the Kalman filter, is introduced. It first calculates a prediction of the current state based on a previous estimate. Then this prediction is used in conjunction with a current measurement (from the measurement equation) to estimate the true current state. More about state estimation and the Kalman filter, in particular, will be given in Section 4.

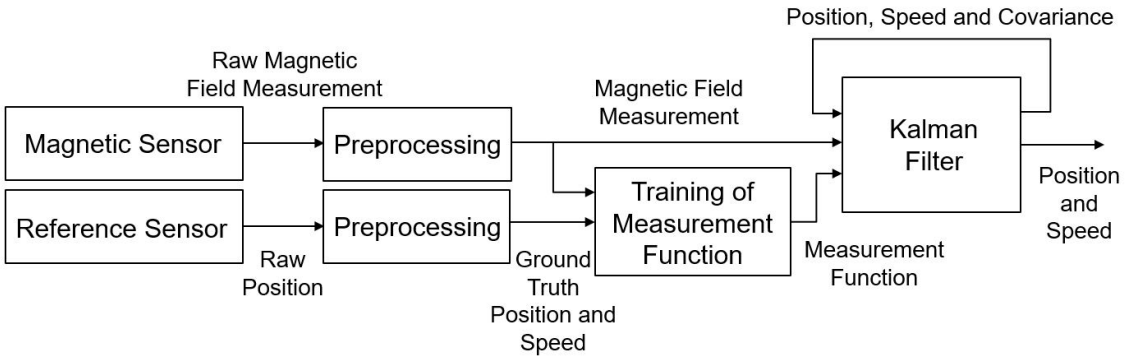


Figure 3.2: Block Diagram of the Position Estimation Approach [1].

CHAPTER 4

State Estimation

State estimation is the process of determining the current state of a system from measurements. For that purpose, the estimator aims to adjust a certain model such that the model results are closer to observed values. Therefore, such approaches can be applied to any problem, where system properties are not directly measurable.

In most sensor applications, a system can only be evaluated at certain discrete time points t_k, where ΔT is the time interval between t_k and t_{k-1}. Define the state vector $\underline{x}_k$, containing the parameters of interest. In the context of the thesis, $\underline{x}_k$ contains the angular position and the rotational speed of the rotor. Additionally, $\underline{y}_k$ contains the observed measurements of the system at a time step k. Generally, a linear time-discrete system can be modelled by the state transition equation, also known as the system equation

$$\underline{x}_k = \mathbf{A}_{k-1}\underline{x}_{k-1} + \mathbf{B}_{k-1}\underline{u}_{k-1} + \underline{\varepsilon}_{k-1}. \qquad \textbf{Linear system equation.}$$

The matrix $\mathbf{A}$ is called the system matrix, while $\mathbf{B}$ is the input matrix. The vector $\underline{u}_k$ contains any control inputs and $\underline{\varepsilon}_k$ is the system noise. The process noise $\underline{\varepsilon}_k$ is commonly assumed to be drawn from a zero mean distribution; $\underline{\varepsilon}_k \sim \mathcal{N}(\underline{0}, \mathbf{Q}_k)$.

Furthermore, the measurement system observes the system state at the discrete time point k

$$\underline{y}_k = \mathbf{H}_k\underline{x}_k + \underline{\xi}_k, \qquad \textbf{Linear measurement equation.}$$

where $\mathbf{H}_k$ is the output matrix and $\underline{\xi}_k$ is the measurement noise, commonly with the distribution $\underline{\xi}_k \sim \mathcal{N}(\underline{0}, \mathbf{R}_k)$.

So far, we have assumed that the system of interest has linear properties. In order to model a non-linear system, one has to first transform the above-mentioned system and measurement equations. The state vector in a typical nonlinear, discrete system can be described as

$$\underline{x}_k = \underline{f}_{k-1}(\underline{x}_{k-1}, \underline{u}_{k-1}, \underline{\varepsilon}_{k-1}), \qquad \textbf{Non-linear system equation.}$$

where $\underline{u}_{k-1}$ and $\underline{\varepsilon}_{k-1}$ characterize the same elements as in the linear system equation. Furthermore, the measurement equation with noise $\underline{\xi}_k$ will have the type

$$\underline{y}_k = \underline{h}_k(\underline{x}_k, \underline{\xi}_k), \qquad \textbf{Non-linear measurement equation.}$$

In the scope of this work, one wants to estimate the system state, consisting of the angular position and rotational speed of the rotor by observing the magnetic field measurements. One of the most popular state estimation approaches is the Kalman filter. In the following, we will present an overview of the Kalman filter, which addresses the problem of estimation in linear systems. Consequently, we will present two modifications of the filter, namely the extended Kalman filter and the unscented Kalman filter, used in [1].

4.1 Kalman Filter

The Kalman filter (KF) [7] is one of the most popular solutions to many tracking tasks, especially in the area of navigation. The filter is a recursive estimation approach that aims to estimate the state of a process by minimizing the mean of the squared error [8].

The overall goal would be to determine an optimal state estimate $\hat{\underline{x}}^e_{k+1}$, given a set of observations $\{\underline{y}_1, .., \underline{y}_{k+1}\}$. The KF could be divided into two parts - the first one being the "Prediction", followed by the "Update". In the former part, the KF calculates a prediction of the current state $\hat{\underline{x}}^p_k$ based on a previous estimate $\hat{\underline{x}}^e_{k-1}$. Then the following "Update" step uses this prediction in conjunction with a current measurement $\underline{y}_k$ to estimate the true current state.

Firstly, one derives the state prediction $\hat{\underline{x}}^p_k$ from the system equation by calculating

$$\hat{\underline{x}}^p_k = \mathbf{A}_{k-1}\hat{\underline{x}}^e_{k-1} + \mathbf{B}_{k-1}\hat{\underline{u}}_{k-1},$$

where $\hat{\underline{x}}^e_{k-1}$ is the estimation of the state for the previous time step.

Secondly, one uses the measurement equation to determine how good the state prediction fits with the current measurement $\underline{y}_k$. Based on that, a correction on the predicted state is applied. Such correction represents a weighting between the prediction and the measurement and is determined by the so-called Kalman gain $\mathbf{G}_k$. The gain is calculated based on the predicted state uncertainty, as well as the measurement uncertainty.

Since the KF is a recursive estimation approach, one first needs to define the initial system estimate $\hat{\underline{x}}^e_0$ with the covariance $\mathbf{C}^e_0$. The prior distribution for the system

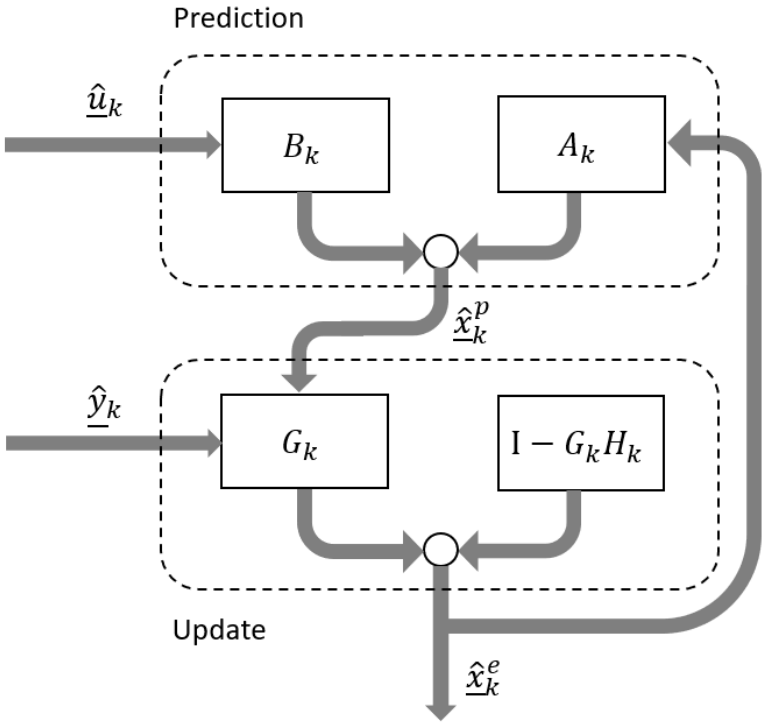


Figure 4.1: Kalman filter, divided into prediction step and update step.

state $\hat{\underline{x}}_0$ is defined with the help of the estimation error given the initial $\hat{\underline{x}}_0$

$$p(\hat{\underline{x}_0}) = \mathcal{N}(\mathbb{E}\underline{x}, \mathbf{P}_0) \quad \text{with} \quad \mathbf{P}_0 = \mathbb{E}(\underline{x} - \hat{\underline{x}}_0)(\underline{x} - \hat{\underline{x}}_0)^T.$$

A prediction step is then taken to calculate $\hat{\underline{x}}_1^p$ and $\mathbf{C}_1^e$. By including the measurement values $\underline{y}_1$ one can update the prediction, resulting in the estimates $\hat{\underline{x}}_1^e$ and $\mathbf{C}_1^e$. From then on, these values are recursively used for the new prediction. A summary of the KF can be seen in Figure 4.1. For simplicity, so far we have omitted the details surrounding the calculation of the filter gain $\mathbf{G}_k$ using predicted state uncertainty and the measurement uncertainty. The full calculations of this procedure can be found in Figure 4.2.

To summarize, one can write the estimate calculations as [9]

$$\hat{\underline{x}}_{k+1}^e = \underbrace{\mathbf{A}_k \hat{\underline{x}}_k^e + \mathbf{B}_k \hat{\underline{u}}_k}_{\text{Output of the system model}} + \underbrace{\mathbf{A}_k \mathbf{G}_k \underbrace{(\hat{\underline{y}}_k - \mathbf{H}_k \hat{\underline{x}}_k^e)}_{\text{Estimation error}}}_{\text{Correction}}.$$

4.2 Extended Kalman Filter

Previously, we introduced the Kalman filter state estimation approach, which is restricted to linear systems models. This model limitation occurs due to the linearity in the measurement equation. Since the estimation task in this work involves a non-linear relationship between the system state and the measurements, namely the magnetic field strength, a modification of the KF is required. For this purpose, we introduce the most popular approach for state estimation of non-linear systems - the Extended Kalman Filter (EKF) ([10], chapter 13).

The EKF is an approximative filter since its approach is to estimate the state of non-linear systems by linearizing it at the state estimate $\underline{\hat{x}}^e_{k-1}$. The first step of this process is the derivation of the following matrices from the state equation:

$$\mathbf{A}_{k-1} = \left.\frac{\partial \underline{f}_{k-1}}{\partial \underline{x}_k}\right|_{\underline{\hat{x}}^e_{k-1}}, \qquad \mathbf{C}_{k-1} = \left.\frac{\partial \underline{f}_{k-1}}{\partial \underline{\varepsilon}_k}\right|_{\underline{\hat{x}}^e_{k-1}}$$

In order to linearalize the measurement equation, one needs to also define

$$\mathbf{H}_{k-1} = \left.\frac{\partial \underline{h}_k}{\partial \underline{x}_k}\right|_{\underline{\hat{x}}^p_k}, \qquad \mathbf{D}_{k-1} = \left.\frac{\partial \underline{h}_k}{\partial \underline{\varepsilon}}\right|_{\underline{\hat{x}}^p_k}$$

The equation of the EKF are summarized in Figure 4.3.

4.3 Unscented Kalman Filter

While the EKF is a standard solution for state estimation of non-linear systems, the approach depends on a linear approximation at the estimate points. In [11] the Unscented Kalman Filter (UKF) is introduced as an alternative approach of state estimation of non-linear systems.

The UKF uses a sampling technique, called unscented transformation (UT) to pick a minimal set of points, also known as "sigma points". Each point is then transformed through the non-linear function. Finally, a Gaussian is computed from the transformed points. Unlike in EKF, estimating the current state with UKF does not require calculating the partial derivatives around the state estimate in order to linearize the system. For certain systems, the computation of the Jacobian matrix is very expensive, and therefore the UKF presents an additional benefit.

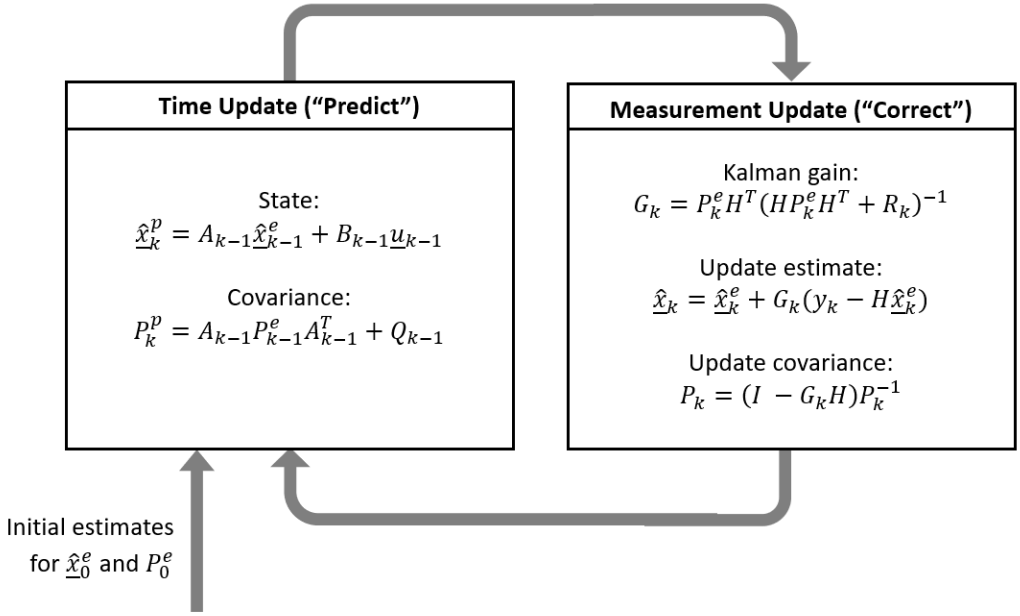


Figure 4.2: Kalman filter calculations.

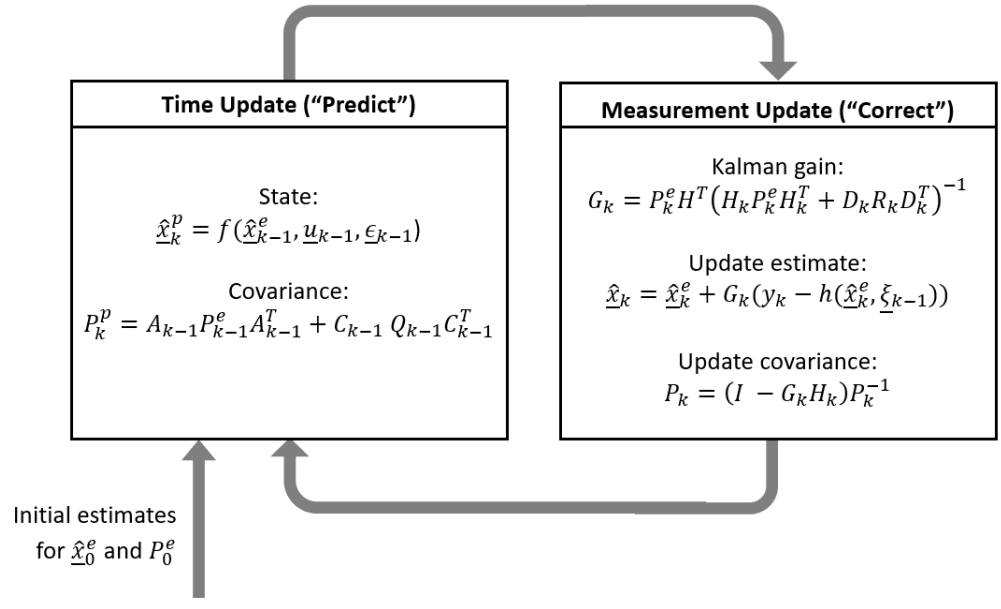


Figure 4.3: Extended Kalman filter calculations.

CHAPTER 5

Probabilistic Regression

The aim of regression models is to estimate a real-valued function f from a set of given data points $\mathcal{D}$. The function thereby maps instances $\underline{x} \in \mathbb{R}^D$ from the input space to a real-valued output

$$f : \mathbb{R}^D \longrightarrow \mathbb{R}.$$

In the cases, when the function f is parameterized, we will notate it as f_w, where w indicates the parameters. To estimate the function f_w (and the parameters w) via a Bayesian formalism, the rules of probability are used

$$p(a) = \int p(a, b)\ db \qquad \textbf{Sum Rule}$$

$$p(a \mid b) = \frac{p(a, b)}{p(b)} \qquad \textbf{Product Rule}$$

where a and b are random variables. In order to apply these rules to a regression problem, one has to clarify what is modeled as random, uncertain or unknown, and which (prior-)distribution assumptions are made. In the Bayesian formalism, the data $\mathcal{D}$ is assumed to be fixed and the parameters w of the parameterized function f_w are unknown and follow a prior-distribution $p(\underline{w})$. Inference about the distribution of f_w is then made via the Sum Rule

$$p(f_w) = \int p(f_w, \underline{w}) d\underline{w} = \int p(f_w \mid \underline{w}) \cdot p(\underline{w})\ d\underline{w},$$

where $p(f_w|\underline{w})$ denotes the modelled relationship between the function f_w and its parameters $\underline{w}$ (e.g. see Section 5.1). In the presence of data $\mathcal{D}$ (realisations of f_w) the distribution $p(f_w)$ of the output values of the function is updated to the posterior distribution $p(f_w|\mathcal{D})$.

$$p(f_w \mid \mathcal{D}) = \int p(f_w \mid \underline{w}, \mathcal{D}) \cdot p(\underline{w} \mid \mathcal{D})\ d\underline{w} = \int p(f_w \mid \underline{w}) \cdot p(\underline{w} \mid \mathcal{D})\ d\underline{w},$$

The dependence of $p(f_w \mid w, D)$ on $\mathcal{D}$ is usually omitted. In such cases $p(f_w \mid \underline{w}, \mathcal{D}) = p(f_w \mid \underline{w})$, as the model $p(f_w \mid \underline{w})$ is assumed to not depend on

previous observations (the realisations of f_w are i.i.d.). The posterior distribution of $\underline{w}$, i.e. $p(\underline{w} \mid \mathcal{D})$ can be calculated via Bayes theorem which can be derived from the Product Rule

$$p(\underline{w} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \underline{w}) \cdot p(\underline{w})}{p(\mathcal{D})} \qquad \textbf{Bayes Theorem.}$$

Even though this framework can be consistently applied independently of different modelling assumptions, the posterior distribution $p(f_w \mid \mathcal{D})$ is often intractable and can only be assessed via approximation methods [12]. For some special cases however, namely (Bayesian) linear regression and Gaussian processes, $p(f_w \mid \mathcal{D})$ has a closed-form solution.

5.1 Bayesian Linear Regression

In (Bayesian) linear regression, a random variable y is modelled as a function of an input $\underline{x}$ and a modeled parametric relationship $f_w(\underline{x})$ as

$$y = f_w(\underline{x}) + \epsilon \qquad \text{with} \qquad f_w(\underline{x}) = \underline{\phi}(\underline{x})^T \underline{w},$$

where $\underline{x} \in \mathbb{R}^D$ is a single D-dimensional input with y being the observed value. We have assumed, that the observed value y differs from the function value $f_w(\underline{x})$ with some random noise ϵ. The function $\underline{\phi}(\underline{x})$ thereby indicates some transformation of an input $\underline{x}$ to a K-dimensional output space

$$\underline{\phi} : \mathbb{R}^D \longrightarrow \mathbb{R}^K.$$

A key property of such a model is that the produced function $f_w(\underline{x})$ is linear with respect to $\underline{w}$, even though the function $f_w(\underline{x})$ may be non-linear in $\underline{x}$.

The vector $\underline{\phi} = (\underline{\psi}_1, ..., \underline{\psi}_M)$ contains M transformations $\underline{\psi}_j(\underline{x})$ with $j = 1, .., M$, also known as basis functions [12]. The vector of parameters $\underline{w} = (\underline{w}_1^T, ..., \underline{w}_M^T)^T$ is constructed as a concatination of the parameters vectors $\underline{w}_j$, corresponding to the basis function $\underline{\psi}_j$.

The space of functions the model can express can be varied by choosing different basis functions $\underline{\psi}_j$ of the vector $\underline{x}$, where frequent choices are [12]

- **Linear** $(M = 1)$:
  $\underline{\phi}(\underline{x}) = (\underline{\psi}_1(\underline{x})) = \underline{x}$
- **Affine** $(M = 2)$:
  $\underline{\phi}(\underline{x}) = (\underline{\psi}_1(\underline{x}), \underline{\psi}_2(\underline{x})) = (\underline{x}^T, 1)^T$

- **Polynomial of degree p** ($M = p$) :
  $\underline{\phi}(\underline{x}) = (\underline{\psi}_1(\underline{x}), ..., \underline{\psi}_{p-2}(\underline{x}), \underline{\psi}_{p-1}(\underline{x}), \underline{\psi}_p(\underline{x})) = ((\underline{x}^p)^T, \cdots, (\underline{x}^2)^T, \underline{x}^T, 1)^T,$

where in the case of the polynomial transformations, the power indicates an element-wise operation.

Define $\mathbf{X} = (\underline{x}_1^T, .., \underline{x}_N^T)$ be a matrix of size N x D, constructed by row-wise stacking of the training data input vectors. The corresponding outputs is then given in $\underline{y} = (y_1, .., y_N)$. To evaluate the training input variables $\mathbf{X}$ and model their output prediction $\underline{y}$, one needs the joint distribution of the function values of all training points, represented as the elements of the vector $\underline{y}$. Define $\mathbf{\Phi}$ for the training data

$$\mathbf{\Phi} = \begin{pmatrix} \underline{\psi}_1(\underline{x}_1)^T & \dots & \underline{\psi}_m(\underline{x}_1)^T \\ \vdots & \ddots & \vdots \\ \underline{\psi}_1(\underline{x}_n)^T & \dots & \underline{\psi}_m(\underline{x}_n)^T \end{pmatrix},$$
$$\text{such that} \quad \underline{y} = f_w(\mathbf{X}) + \underline{\varepsilon} \quad \text{with} \quad f_w(\mathbf{X}) = \mathbf{\Phi}\underline{w}.$$

As the aim of regression is to infer the probability distribution of the real valued output y (and the parameters $\underline{w}$), it is usually modeled as normally distributed $\underline{y} \sim \mathcal{N}(f_w(\mathbf{X}), \sigma_y^2 \cdot \mathbf{I})$. The parameters $\underline{w}$ are also unknown and modelled as a normally distributed random variable $\underline{w} \sim \mathcal{N}(0, \sigma_w^2 \cdot \mathbf{I})$.
In the presence of data $\mathcal{D} = \{(x_i, y_i) \mid i = 1, .., N\}$, which consists of tuples of input output relationships, the distribution of the parameters $p(\underline{w} \mid \mathcal{D})$ can be updated via Bayes Theorem as described earlier. From this, the distribution of y can then be inferred. Even though $p(\underline{w} \mid \mathcal{D})$ can be obtained in closed-form for the given special case, sometimes, a point-estimate of $\underline{\hat{w}}$ is preferred over the distribution $p(\underline{w} \mid \mathcal{D})$. Such an approximation is reasonable, if one specific value of $\underline{w}$ contributes the majority of the probability mass of $p(\underline{w} \mid \mathcal{D})$, and can be calculated by maximizing $p(\underline{w} \mid \mathcal{D})$

$$\begin{aligned}
\underline{\hat{w}} &= \arg\max_{\underline{w}} \quad p(\underline{w} \mid \mathcal{D}) \\
&= \arg\max_{\underline{w}} \quad \frac{p(\mathcal{D} \mid \underline{w}) \cdot p(\underline{w})}{p(\mathcal{D})} \\
&= \arg\max_{\underline{w}} \quad p(\mathcal{D} \mid \underline{w}) \cdot p(\underline{w}) \\
&= \arg\max_{\underline{w}} \quad \log p(\underline{y} \mid \mathbf{X}, \underline{w}) + \log p(\underline{w}) \\
&= \arg\min_{\underline{w}} \quad \frac{1}{2}(\underline{y} - \mathbf{\Phi}\underline{w})^T \Sigma_y^{-1} (\underline{y} - \mathbf{\Phi}\underline{w}) + \frac{1}{2} w^T \Sigma_w^{-1} w.
\end{aligned}$$

Due the the modelling assumptions of $\underline{y} \sim \mathcal{N}\left(f_w(\mathbf{X}), \sigma_y^2 \cdot \mathbf{I}\right)$ (with y_i being i.i.d.) and $\underline{w} \sim \mathcal{N}(0, \sigma_w^2 \cdot \mathbf{I})$, where $\sigma_y^2 \cdot \mathbf{I} = \Sigma_y$ and $\sigma_w^2 \cdot \mathbf{I} = \Sigma_w$, the estimation is reduced

to

$$\hat{\underline{w}} = \arg\min_{\underline{w}} \quad \frac{1}{2\sigma_y^2} \left\| \underline{y} - \mathbf{\Phi}\underline{w} \right\|_2^2 + \frac{1}{2\sigma_w^2} \|\underline{w}\|_2^2.$$

Since the resulting $\hat{\underline{w}}$ maximizes the a-posterior distribution of $p(\underline{w} \mid \mathcal{D})$ it is known as *Maximum A-Posteriori Estimate* (MAP) [13]. The terms $\frac{1}{\sigma_w^2}$ and $\frac{1}{\sigma_y^2}$ are merely weighting factors for both summands and are usually summarized in a single parameter $\lambda \in \mathbb{R}^+$ in front of $\|\underline{w}\|_2^2$ which leads to a formulation of the estimate equivalent to Ridge-Regression [12], Tikhinov Regularisation [14] or penalized Maximum-Likelihood [15].

The solution for the the MAP estimate, can be derived analytically in closed form by setting the gradient with respect to $\underline{w}$ to $\underline{0}$

$$\begin{aligned}
\underline{0} &= \nabla_{\underline{w}} \left(\frac{1}{2} \left\| \underline{y} - \mathbf{\Phi}\underline{w} \right\|_2^2 + \frac{\lambda}{2} \|\underline{w}\|_2^2 \right) \\
&= \mathbf{\Phi}^T \left(\underline{y} - \mathbf{\Phi}\underline{w} \right) + \lambda \cdot \underline{w} \\
&= \mathbf{\Phi}^T \underline{y} - \mathbf{\Phi}^T \mathbf{\Phi}\underline{w} + \lambda \cdot \underline{w} \\
&= \mathbf{\Phi}^T \underline{y} - (\mathbf{\Phi}^T \mathbf{\Phi} - \lambda \cdot \mathbf{I})\underline{w} \\
\hat{\underline{w}} &= (\mathbf{\Phi}^T \mathbf{\Phi} + \lambda \cdot \mathbf{I})^{-1} \mathbf{\Phi}^T \underline{y}.
\end{aligned}$$

The analytical solution for the MAP estimate derived this way, can be seen as a variant of the normal equation ([12], chapter 3), where the $\lambda \cdot \mathbf{I}$ added to the diagonal of $\boldsymbol{\Phi^T \Phi}$ helps in case of bad conditioning.

5.2 Gaussian Processes

In the case of linear regression, a distribution of functions is implicitly given through modelling a distribution over the parameters $\underline{w}$ of the function f_w. Gaussian processes (GP) take a different approach here, where, instead of modelling a distribution over parameters of a parametric model, GPs directly define a prior distribution over functions $p(f(\underline{x}))$ [12].

Even though such a formulation may seem unrelated to linear regression at first glance, both concepts are actually closely linked. Note that, in case of Bayesian linear regression with $\underline{w} \sim \mathcal{N}(0, \sigma_w^2 \cdot \boldsymbol{I})$, we can assess the mean and covariance of

an output $f_w(\underline{x})$ of the parametric model by

$$\begin{aligned}
\mathbb{E}_w[f_w(\mathbf{X})] &= \mathbb{E}_w[\mathbf{\Phi}\underline{w}] = \mathbf{\Phi}\mathbb{E}_w(\underline{w}) = 0 \\
\mathbb{C}[f_w(\mathbf{X}), f_w(\mathbf{X})] &= \mathbb{E}_w[(f_w(\mathbf{X}) - \mathbb{E}_w[f_w(\mathbf{X})])(f_w(\mathbf{X}) - \mathbb{E}_w[f_w(\mathbf{X})])^T] \\
&= \mathbb{E}_w[f_w(\mathbf{X}) f_w(\mathbf{X})^T] \\
&= \mathbb{E}_w[\mathbf{\Phi}\underline{w}\ \underline{w}^T\mathbf{\Phi}^T] \\
&= \mathbf{\Phi}\mathbb{E}_w[\underline{w}\ \underline{w}^T]\mathbf{\Phi}^T \\
&= \mathbf{\Phi}(\sigma_w^2\mathbf{I})\mathbf{\Phi}^T \\
&= \sigma_w^2\mathbf{\Phi}\mathbf{\Phi}^T
\end{aligned}$$

One can see that the relationship between two function outputs $f(\mathbf{X})$ is given by the scaled linear combination $\mathbf{\Phi}\mathbf{\Phi}^T$ of the transformed training examples. Instead of generating $\mathbf{\Phi}$ through a choice of explicit transformations $\underline{\phi}$, one can also define a so-called covariance function $k(\underline{x}, \underline{x}')$ that builds the covariance matrix $k(\mathbf{X}, \mathbf{X}) = \mathbf{K}_{NN} = \mathbf{\Phi}\mathbf{\Phi}^T$ directly [12]. Through choice of $k(\underline{x}, \underline{x}')$, the choice of the transformation $\underline{\phi}(\underline{x})$ is then implicitly taken.

This means that we can either define a distribution of f_w through a parametric form $\mathbf{\Phi}\underline{w}$ which makes f_w normally distributed as a linear combination of normally distributed random variables (with a certain mean and covariance), or we can explicitly define f to be normally distributed by choice of a mean function $\underline{m}(\underline{x})$ and a covariance function $k(\underline{x}, \underline{x}')$ directly. Due to common convention, we will refer to the covariance function k as "kernel" from now on. This terminology was previously avoided due to its less illuminating nature.

By introducing the kernel, one can circumvent the explicit definition of $\underline{\phi}(\underline{x})$. This, in other words, allows us to avoid the explicit choice of M basis functions and implicitly use a possibly infinite-dimensional function space [12].

Having defined a prior distribution over $p(f)$, as mentioned before, we can calculate the prior distribution of our regression output y through the application of the Sum Rule

$$p(y) = \int p(y \mid f) \cdot p(f) df.$$

As f is a function, define $\underline{f} = (f(\underline{x}_1), ..., f(\underline{x}_N))$ being the execution of f for the training points. The distribution $p(\underline{y} \mid \underline{f})$ can be modelled straightforwardly through a normal distribution (as done in linear regression) with $p(\underline{y} \mid \underline{f}) = \mathcal{N}(\underline{f}, \sigma_y^2 \cdot \mathbf{I})$, where $p(\underline{f}) = \mathcal{N}(\underline{0}, \mathbf{K}_{NN})$ and $p(\underline{y}) = \mathcal{N}(\underline{0}, \mathbf{K}_{NN} + \sigma_y^2 \cdot \mathbf{I})$.

The joint multivariate distribution of $\underline{f}$ and f_* is given as

$$p(\underline{f}, f_*) = \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\mathbf{K}_{NN} & \mathbf{K}_{N*}\\ \mathbf{K}_{*N} & \mathbf{K}_{**}\end{pmatrix}\right),$$

where $f_* = f(\underline{x}_*)$ with a test point $\underline{x}_*$. Furthermore, $k(\mathbf{X}, \underline{x}_*) = \mathbf{K}_{N*}$ is a matrix, build by the kernel. $\mathbf{K}_{NN}, \mathbf{K}_{*N}$ and $\mathbf{K}_{**}$ are build analogically.

After including the model noise $\sigma_y^2 \cdot \mathbf{I}$, we obtain the joint distribution of training and testing inputs

$$p(\underline{y}, y_*) = \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\mathbf{K}_{NN} + \sigma_y^2 \cdot \mathbf{I} & \mathbf{K}_{N*}\\ \mathbf{K}_{*N} & \mathbf{K}_{**} + \sigma_y^2\end{pmatrix}\right).$$

As a result, the posterior distribution of the test output y_* given $\mathcal{D}$ is calculated by conditioning on the observed training data. Since $p(y_* \mid \mathcal{D}) = p(y_* \mid \underline{y})$ one derives

$$p(y_* \mid \mathcal{D}) = \mathcal{N}(\mathbf{K}_{*,N}(\mathbf{K}_{NN} + \sigma_y^2 \cdot \mathbf{I})^{-1}\underline{y}, \quad \mathbf{K}_{**} + \sigma_y^2 \cdot \mathbf{I} - \mathbf{K}_{*N}(\mathbf{K}_{NN} + \sigma_y^2 \cdot \mathbf{I})^{-1}\mathbf{K}_{N*}).$$

The detailed calculations of the posterior were omitted, but the underlying theory can be found in [12], chapter 2.3.1.

5.2.1 Covariance Functions

A covariance function (also called a kernel) is a positive definite function of two input vectors $\underline{x}$ and $\underline{x}'$ from the input space. It encodes all assumptions about the form of the modelled function. The kernel defines the similarity of two values of a function calculated at two different locations in the input space.

$$k(\underline{x}, \underline{x}') = \mathbb{C}[f(\underline{x}), f(\underline{x}')].$$

The properties of the regression model are controlled by the hyperparameters. In the following, the collection of any hyperparameters of a covariance function will be referred to as the vector $\underline{\vartheta}$.

The Squared Exponential (SE) Kernel

The Squared Exponential (SE) kernel is one of the most commonly used covariance functions. The definition of the kernel is

$$k(\underline{x}, \underline{x}') = \sigma \cdot exp\left(-\frac{(\underline{x} - \underline{x}')^2}{2l^2}\right).$$

where σ is the signal standard deviation, used to scale the entire kernel. The definition of how near $\underline{x}$ and $\underline{x}'$ are is controlled by the length scale l. When the two vectors are near, their covariance is high, which enforces smoothness.

Automatic Relevance Determination

In the previously described SE kernel, all input dimensions have the same length scale l. This, however, creates some model restrictions when modelling a multi-dimensional input space. A flexible way to model multi-input functions is by multiplying kernels defined individually for every input, each with its length scale parameter. An example of a kernel, constructed as a product of SE kernels is the SE-ARD kernel [16], where

$$k(\underline{x}, \underline{x}') = \sigma \cdot exp\left(-\frac{1}{2}\sum_{d=1}^{n}\frac{(x_d - x_d')^2}{l_d^2}\right).$$

The abbreviation ARD stands for automatic relevance determination. The ARD kernel learns an individual length scale hyperparameter l_d for each input dimension, which implicitly defines how relevant the corresponding dimension is.

Periodic Covariance Function

The periodic kernel, derived in [17] allows one to model functions with periodical structure. In this case, each input from a one dimensional input space $x \in \mathcal{X}$ is mapped to a two-dimensional space, where $u(x) = (cos(x), sin(x))$.
Since

$$(cos(\underline{x}) - cos(\underline{x}'))^2 + (sin(\underline{x}) + sin(\underline{x}'))^2 = 4sin^2\left(\frac{\underline{x} - \underline{x}'}{2}\right),$$

the periodic kernel is then defined as a modification of the SE kernel.

$$k(\underline{x}, \underline{x}') = \sigma \cdot exp\left(-\frac{2\,sin^2\left(\frac{\underline{x} - \underline{x}'}{2}\right)}{l^2}\right)$$

where σ and l are again hyperparameters.

Combination rules

A problem arises when one wants to consider a structure that is not properly expressed by any of the common kernels. However, one can create new covariance

functions by combining or modifying already existing ones [15], chapter 4. The two main rules of combining properties are kernel addition and multiplication.

Adding kernels - The addition of two kernels could roughly be viewed as an OR operation. The resulting kernel will have a high value if either of the summand kernels has a high value.

Multiplying kernels - The multiplication of two kernels creates a valid kernel ([15], Chapter 4) and is a commonly used, especially when the two kernels are defined on different inputs of the function. The resulting kernel will have high values only if both of the base kernels have high values.

5.3 Sparse Gaussian Processes Using Pseudo-Inputs

Previously, we introduced GP as a regression approach. To summarize, in order to make predictions for test data, one needs to calculate the posterior distribution. Computing the inverse $[\mathbf{K}_{NN} + \sigma_y^2 \cdot \mathbf{I}]^{-1}$ costs $\mathcal{O}(N^3)$, where N is the number of training points. This makes inference prohibitively slow when working with bigger training sets. The computational problem, caused by the non-parametric nature of the GP, was addressed in [2]. The GP approximation the authors undertake involves finding an active set of M point locations in the input space that are used to parametrize the kernel.

In Section 5.2 we derived the posterior distribution of a test output y

$$\mu_* = \mathbf{K}_{*,N}(\mathbf{K}_{NN} + \sigma_y \cdot \mathbf{I})^{-1}\underline{y}$$
$$\sigma_*^2 = \mathbf{K}_{**} + \sigma_y^2 \cdot \mathbf{I} - \mathbf{K}_{*N}(\mathbf{K}_{NN} + \sigma_y \cdot \mathbf{I})^{-1}\mathbf{K}_{N*}.$$

Assuming the hyperparameters $\underline{\vartheta}$ of the kernel are already calculated, the mean μ_* and covariance σ_* can be considered as functions effectively parametrized by the training data. The intuition behind [2] is to replace the real data set $\mathcal{D}$ with a pseudo data set $\bar{\mathcal{D}} = \{(\underline{\bar{x}}_j, \bar{f}_j) \mid j = 1, .., P\}$.

Note that, $\underline{\bar{x}}_j$ are not real locations and their outputs are not real observations, hence it does not make sense to include noise on them. They are therefore equivalent to the latent values of the function $\bar{f}_j = f(\underline{\bar{x}}_j)$. However, the actual observed and predicted values of the regression model will be assumed noisy.

The single test point likelihood will then be

$$p(y_* \mid x_*, \bar{\mathcal{D}}) = p(y_* \mid x_*, \bar{\mathbf{X}}, \underline{\bar{f}}) = \mathcal{N}(\mathbf{K}_{*P}\mathbf{K}_{PP}^{-1}\underline{\bar{f}}, \quad \mathbf{K}_{**} + \sigma_y^2 - \mathbf{K}_{*P}\mathbf{K}_{PP}^{-1}\mathbf{K}_{P*}).$$

The likelihood of the entire training data is then

$$p(\underline{y} \mid \underline{\bar{f}}) \overset{i.i.d.}{=} \prod_{i=1}^{N} p(y_i \mid \underline{\bar{f}}) = \mathcal{N}(\mathbf{K}_{NP}\mathbf{K}_{PP}^{-1}\underline{\bar{f}}, \quad \text{diag}(\mathbf{K}_{NN} - \mathbf{Q}) + \sigma_y^2 \cdot \mathbf{I})$$
$$\text{with} \quad \mathbf{Q} = \mathbf{K}_{NP}\mathbf{K}_{PP}^{-1}\mathbf{K}_{PN}.$$

Next, one needs to define a prior $p(\bar{f})$ on the pseudo outputs. Since they are expected to be distributed similarly to the original outputs,

$$p(\underline{\bar{f}}) = p(0, \mathbf{K}_{PP})$$

would be a reasonable prior.

To find the marginal likelihood

$$\begin{aligned} p(\underline{y}) &= \int p(\underline{y} \mid \underline{\bar{f}}) \cdot p(\underline{\bar{f}}) d\underline{\bar{f}} \\ &= \mathcal{N}(\underline{0}, \mathbf{Q} + \text{diag}(\mathbf{K}_{NN} - \mathbf{Q}) + \sigma_y^2 \cdot \mathbf{I}). \end{aligned}$$

Analog to the standard GP, we derive the predictive distribution $p(y_* \mid \underline{x}_*, \mathcal{D}, \underline{\bar{x}}) = p(y_* \mid \underline{y})$

$$\begin{aligned} p(y_* \mid \underline{x}_*, \mathcal{D}, \underline{\bar{x}}) &= \mathcal{N}(\mu_*, \sigma_*^2) \\ \mu_* &= \mathbf{K}_{*P}(\mathbf{K}_{PP} + \mathbf{K}_{PN}\mathbf{D}\mathbf{K}_{NP})^{-1}\mathbf{K}_{PN}\mathbf{D}\underline{y} \\ \sigma_*^2 &= \mathbf{K}_{**} + \sigma_y^2 - \mathbf{K}_{*P}(\mathbf{K}_{PP}^{-1} - (\mathbf{K}_{PP} + \mathbf{K}_{PN}\mathbf{D}\mathbf{K}_{NP})^{-1})\mathbf{K}_{P*} \\ &\text{with} \quad \mathbf{D} = (\text{diag}(\mathbf{K}_{NN} - \mathbf{Q}) + \sigma_y^2 \cdot \mathbf{I})^{-1}. \end{aligned}$$

So far, we assumed that the locations of the pseudo-points $\bar{\mathbf{X}}$, as well as their outputs $\underline{\bar{f}}$ are already calculated. Finding the pseudo-points $\bar{\mathcal{D}}$ and the hyperparameters $\underline{\vartheta}$ of the kernel is done by maximizing the marginal likelihood $p(\underline{y} \mid \mathbf{X}, \bar{\mathbf{X}}, \underline{\vartheta})$ with respect to $\bar{\mathbf{X}}$ and $\underline{\vartheta}$ by using gradient ascent [2], [18].

CHAPTER 6

Methodology

In Section 3 we gave an overview of the models build in [1] that challenge the task of BLDC motors state estimation. As we previously stated, the goal of this thesis is to improve the rotor state estimation, while keeping the hardware setup, described in Section 3. As we take the same general structure of the EKF, we will concentrate on improving the regression model.

In the following, we will describe how the data sets are generated and pre-processed. Subsequently, we will explain the regression models of choice. Lastly, we will describe how these models are integrated in the estimation model - the EKF.

6.1 Data Description

As described in Section 3, the angular position values of the set are obtained by the reference sensor, while the output values are the result of the measurements of the magnetic sensor. Initially, the input data consists of the angular position of the rotor, while the output contains measurements of the magnetic flux density in both x- and y-axial direction.

6.1.1 Sensor Data Pre-processing

As the measurement noise of the optical encoder is neglectably small, one can accurately derive the angular velocity of the rotor given the measured angular position. The initial sensor measurement values for the angular position θ^{sensor} are $\theta^{sensor} \in [0°, 360°)$. In order to create a dependency between the angular position and angular velocity, one needs to first convert θ^{sensor} to radians, and then calculate the speed in radians per second (RPS)

$$\theta_k = \frac{\pi \cdot \theta_k^{sensor}}{180°}.$$

Assuming that all measurements are taken in a constant time interval ΔT, the ground truth angular velocity in time step k can be derived from the angular position by calculating

$$\omega_k = \frac{\theta_k - \theta_{k-1}}{\Delta T}.$$

Furthermore, B_k^x and B_k^y will denote the sensor measurements of magnetic flux density at a time step k in x-axial direction and y-axial direction respectively.

Consequently, the ranges of the values are

$$\theta_i \in [0\,rad,\ 2\pi\,rad), \quad \omega_i \in [-250\,RPS,\ 250RPS], \quad B_i^x, B_i^y \in [0,\ 4096].$$

6.1.2 Data Sets

In this thesis, two different sets of data, namely data set A, and data set B, are used. The first data set (A) is used to train the regression models and is therefore partitioned into training, validation and test set. The second data set (B) is used exclusively for testing. In data set A, which was also used in [1], the motor is run at different speeds while the directions are alternated. This way, 53590 data points are collected. The profile of the change in speed over time can be seen in Figure 6.1.

The second data set (B), which will be used exclusively for testing, consists of 33489 measurements. For the generation of this data set, the motor was set up to run with increasing speed in one direction until the maximum speed is reached. Then, the motor was slowed down and the same gradual increase of speed in the other direction was performed. The profile of the change in speed over time is visualized in Figure 6.2.

6.1.3 Train and Test Set Selection

To get a good representation of various speeds for our training, validation and testing sets (which are generated from data set A), we have to decide on a scheme to assign data points to the respective partitions. For this, the points for each partition are selected equidistant and non-overlapping. Also, the investigated models work with a different number of training points. These points are again chosen equidistant in time (as subset of the training set). An exemplary illustration of the training-subset-section scheme is illustrated in Figure 6.3. Note that, not all models use the entire training set.

Going forward with the description of the models, we encounter several parameters that need to be set manually. The value of those parameters will be chosen based

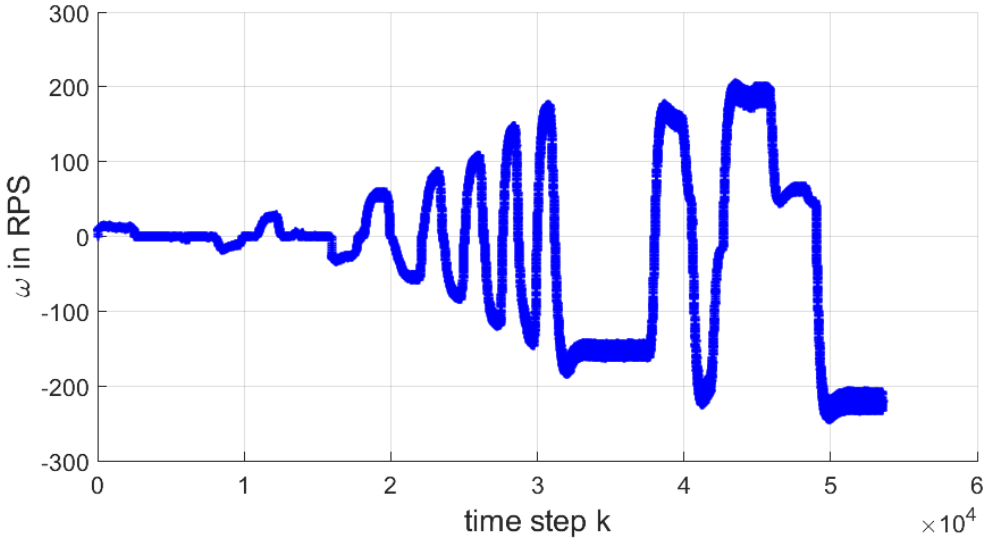


Figure 6.1: Change in ω on data set A.

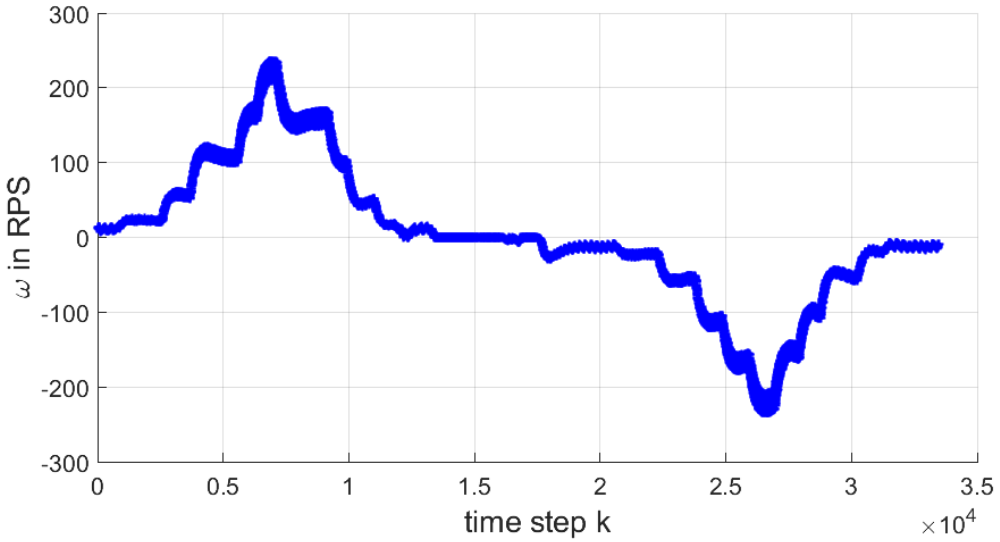


Figure 6.2: Change in ω on data set B.

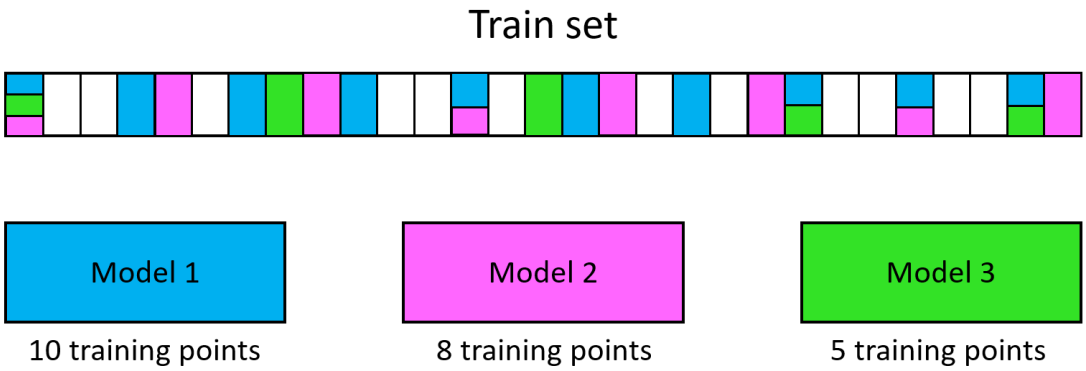


Figure 6.3: The way different models can select equidistant training points from the train partition of the data set. Note that, not the entire partition is used by all models.

on the validation set performance. Finally, the trained models are evaluated on data set B for which we report both their prediction (i.e. magnetic field) and their estimation (i.e. angle and speed) performance. Furthermore, as data set B is used entirely for testing, it is not split but used as is. In Figure 6.4, it is shown how the two data sets are utilized. Finally, the best performing models on the data set B are also evaluated on the test set partition of data set A.

From now on, we will not talk about the entire set of measurements, generated at a specific time step k, but only about the instances in the training set. Consider separate training sets $\mathcal{D}^x$ and $\mathcal{D}^y$, for the x-axial and y-axial direction of the magnetic field respectively

$$\mathcal{D}^x = \{(\underline{x}_i, B_i^x) \mid i = 1, .., N\}, \quad \mathcal{D}^y = \{(\underline{x}_i, B_i^y) \mid i = 1, .., N\}, \text{ with } \quad \underline{x}_i = \begin{pmatrix} \theta_i \\ \omega_i \end{pmatrix}.$$

6.1.4 Rescaling the Input Space

Kernel regression models, such as GPs, are prone to numerical problems as they can produce ill-conditioned covariance matrices $\mathbf{K}_{NN}$. As a reminder, the covariance matrices are the realization of the kernel function if we plug in all our training inputs. One way this ill-conditioning can happen is if the different input dimensions of the data are on different scales. To circumvent this, we rescale the angular velocity values to match the scale of the angular position through

$$\omega_i^{scaled} = \frac{\omega_i - min(\underline{\omega})}{max(\underline{\omega}) - min(\underline{\omega})} \cdot 2\pi \quad \text{, for all } i \in 1, ..., N,$$

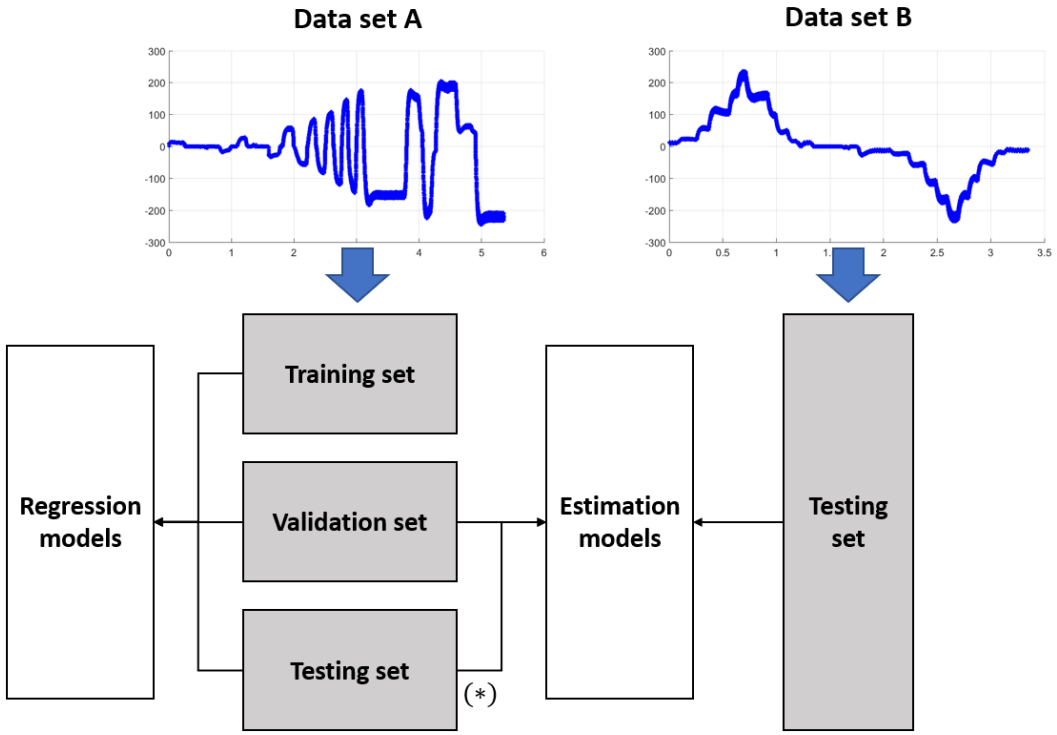


Figure 6.4: Usage of the two data sets A and B. The (*) indicates, that the testing set from A is used to evaluate only the models, showing best performance on data set B.

where $\underline{\omega} = (\omega_1, ..., \omega_N)$. Furthermore, we shift the output values B_i^x and B_i^y for $i = 1, ..., N$

$$B_i^{x,shift} = B_i^x - mean(\underline{B}^x)$$
$$B_i^{y,shift} = B_i^x - mean(\underline{B}^y),$$

so that the output has zero mean. This is done to match the zero mean prior of the Gaussian Process. Through these steps, we construct two additional variants of the training sets

$$\mathcal{D}^{x,scaled} = \{(\underline{x}_i^{scaled}, B_i^{x,shift}) \mid i = 1, .., N\},$$
$$\mathcal{D}^{y,scaled} = \{(\underline{x}_i^{scaled}, B_i^{y,shift}) \mid i = 1, .., N\}, \quad \text{with} \quad \underline{x}_i^{scaled} = \begin{pmatrix} \theta_i \\ \omega_i^{scaled} \end{pmatrix}.$$

6.2 Constructed Regression Models

The main part of this thesis is the construction of multiple regression models with different properties. The models are based on the three "base" models we discussed in Section 5- GPs, SPGPs and linear regression with basis functions. The first point

of difference between the models, derived from the same "base" method, comes with the choice of the kernel. The second cause of difference comes with the choice of the data set, used for training - $\mathcal{D}^x$ and $\mathcal{D}^y$ for the original or $\mathcal{D}^{x,scaled}$ and $\mathcal{D}^{y,scaled}$ for scaled input data. As a result of the different combinations, ten different regression models for each axial-direction of the magnetic field built in total.

6.2.1 Gaussian Processes

Previously, we introduced the theory behind the GPs for regression problems. We build four different model configurations, based on this theory. Each of the configurations is used separately to predict the magnetic field in each axial direction. All of the models are trained with 800 instances of the training set. Two of the models use the original data $\mathcal{D}^x$ and $\mathcal{D}^y$, while the other two use the rescaled one. Furthermore, two different kernels were used - the squared exponential with automatic relevance determination, and a custom periodic kernel. The details about the two covariance functions will be presented later in this chapter. While the kernels and the probability calculations are manually coded, the hyperparameters of the kernel are optimized using the Gaussian Processes for Machine Learning (GPML) Toolbox [19].

To spare the costly computation of the matrix inversion in the mean function, we precompute $(\mathbf{K}_{NN} + \sigma_y^2 \cdot \mathbf{I})^{-1}\underline{y}$, which results in an N-dimensional vector. After training the GP, we can directly use this vector when making predictions for testing inputs, which brings the time complexity down to $\mathcal{O}(N)$ per test case.

6.2.2 Linear Regression with Radial Basis Functions

The next group of regression models is based on the linear regression model introduced in Section 5.1. To remind, we can select M so-called basis functions and build the regression model as a linear combination of these basis functions. In this work, we use $M = 800$ basis functions defined through a function of 800 basis points. We select these points from the predefined training set, such that they are equally distributed. Define $\mathbf{X}_F = (\underline{x}_{F,1}^T, ..., \underline{x}_{F,M}^T)$ as a matrix of size $M \times D$, constructed by row-wise stacking of the inputs of the basis points. Furthermore, define the matrix $\mathbf{X} = (\underline{x}_1^T, ..., \underline{x}_N^T)$ with the rest of the points in the training set, where N is their number. The corresponding output of the training set would be $\underline{y} = (y_1, ..., y_N)^T$.

Therefore, a single basis function will be defined as

$$\phi_i(\underline{x}) = k(\underline{x}_{F,i}, \underline{x}).$$

From these basis function we can then create the matrix

$$\mathbf{\Phi} = \begin{pmatrix} k(\underline{x}_{F,1}, \underline{x}_1) & \dots & k(\underline{x}_{F,M}, \underline{x}_1) \\ \vdots & \ddots & \vdots \\ k(\underline{x}_{F,1}, \underline{x}_N) & \dots & k(\underline{x}_{F,M}, \underline{x}_N) \end{pmatrix},$$

such that $\underline{y} = f_w(\mathbf{X}) + \underline{\varepsilon}$ with $f_w(\mathbf{X}) = \mathbf{\Phi}\underline{w}$.

So far, we have defined the matrix $\mathbf{\Phi}$, but still have no information about $\underline{w}$. In equation 5.1 we showed how the maximum a-posteriori estimate of the parameters $\underline{\hat{w}}$ can be calculated. The value of the regularization constant λ is selected by evaluating multiple values on the validation set and choosing the one for which the model performed best. After calculating the parameters $\underline{\hat{w}}$ (for the selected λ) and already having defined the matrix $\mathbf{\Phi}$, we have all needed components to make predictions with the linear regression model. The output of a single test instance is then calculated as

$$y = \mathbf{K}_{*F}^T \underline{\hat{w}}, \quad \text{where} \quad \mathbf{K}_{*F} = \begin{pmatrix} k(\underline{x}, \underline{x}_{F,1}) \\ \vdots \\ k(\underline{x}, \underline{x}_{F,1}) \end{pmatrix}.$$

To develop a better intuition about the linear regression approach with basis functions created by the SE in particular, we illustrate an example with five basis functions. On Figure 6.5 one can see how five basis functions looks like before being weighted. After calculating the maximum a-posteriori estimate $\underline{\hat{w}}$, the linear combination of the basis functions looks like the magenta line in Figure 6.6.

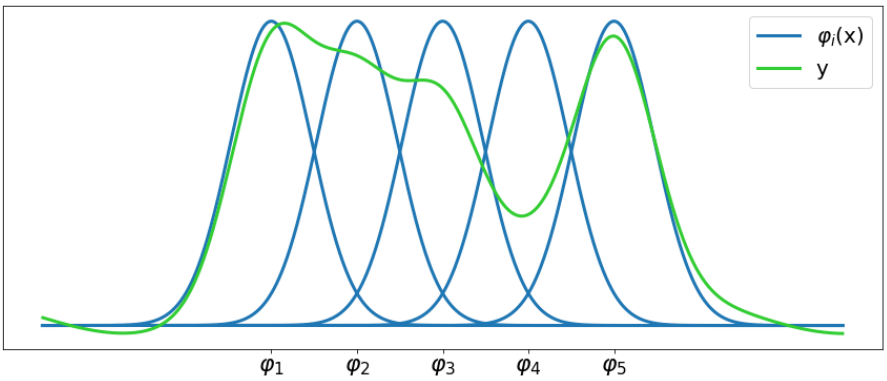


Figure 6.5: Unweighted SE basis functions ϕ_i and $\underline{y}$ that is to be approximated.

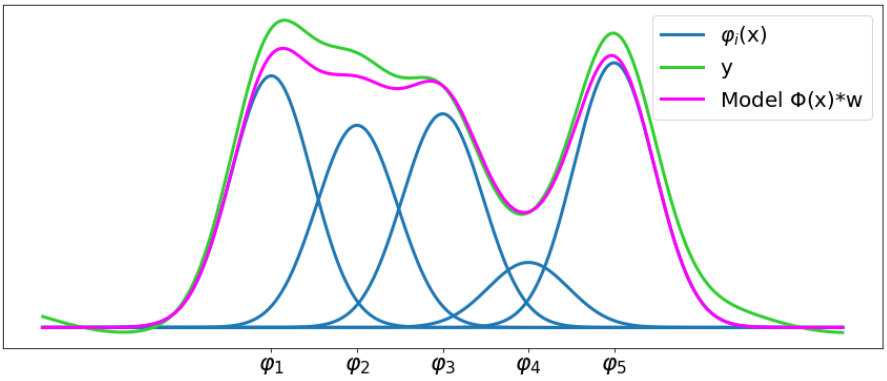


Figure 6.6: Model prediction with weighted SE basis functions ϕ_i trying to approximate $\underline{y}$.

6.2.3 Sparse Gaussian Process Using Pseudo-Inputs

The third group of models are based on the paper "Sparse Gaussian Processes using Pseudo-Inputs" [2] we described in Section 5.3. The models use 2500 training points and one much smaller set of pseudo-points of size $P = 300$. The most computationally intensive part of calculating the predictive mean (equation 5.3) for a testing point is the calculation of $(\mathbf{K}_{PP} + \mathbf{K}_{PN}\mathbf{D}\mathbf{K}_{NP})^{-1}\mathbf{K}_{PN}\mathbf{D}\underline{y}$, which again, can be precomputed. Once the precomputation has is done, the mean prediction per test case lies in $\mathcal{O}(P)$, where the variance prediction costs $\mathcal{O}(P^2)$ [2].

In Figure 7.2a and 7.2b, one can see the SPGP predictive mean function of the magnetic field in x-axial and y-axial direction respectively. The black crosses are the locations of the pseudo-points, which are optimized together with the hyperparameters of the kernel.

6.3 Kernels

In this thesis, we use two different kernels to define our regression models. Both of the kernels consider different length scales for each of the two input dimensions- the angular position θ and the rotational speed ω.

As said earlier, each kernel has hyperparameters to define its structure. For all models, we use the parameter vector $\underline{\vartheta}$ to summarize the hyperparameters of the kernels.

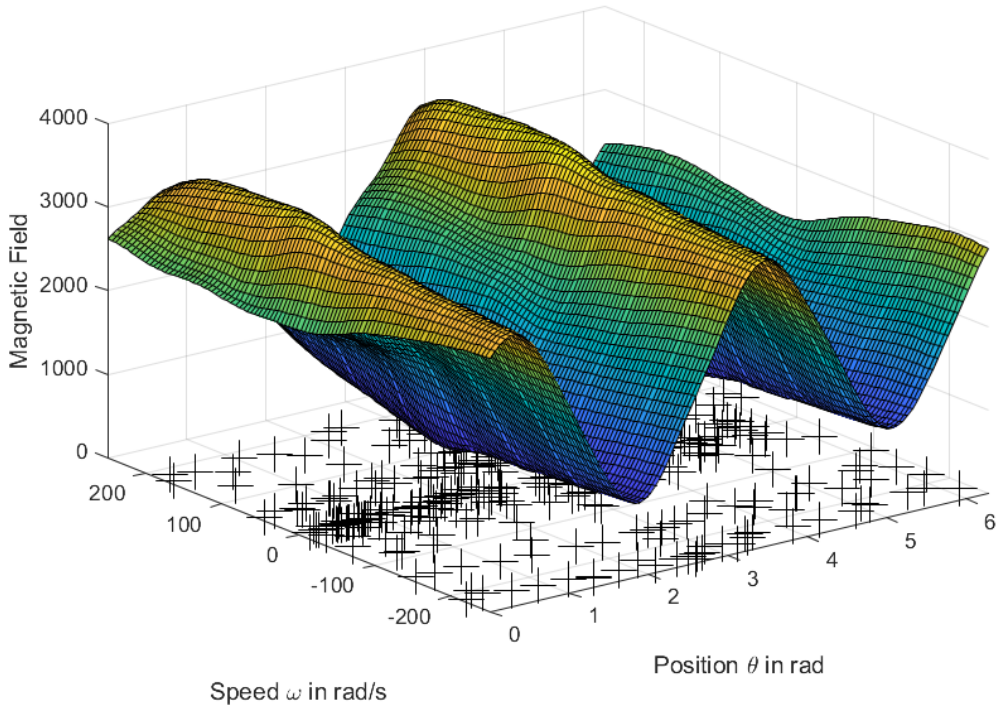
$$\underline{\vartheta} = \begin{pmatrix} l_\theta \\ l_\omega \\ \sigma_s \\ \sigma_y \end{pmatrix}.$$

Here, the l_θ and l_ω are the lengthscales for each input dimension. The hyperparameter σ_s is the signal standard deviation, while σ_y is the noise standard deviation. While there exist several approaches to determine the hyperparameters, we will go with minimizing the negative log marginal likelihood with respect to the hyperparameters as described in [15], chapter 5.

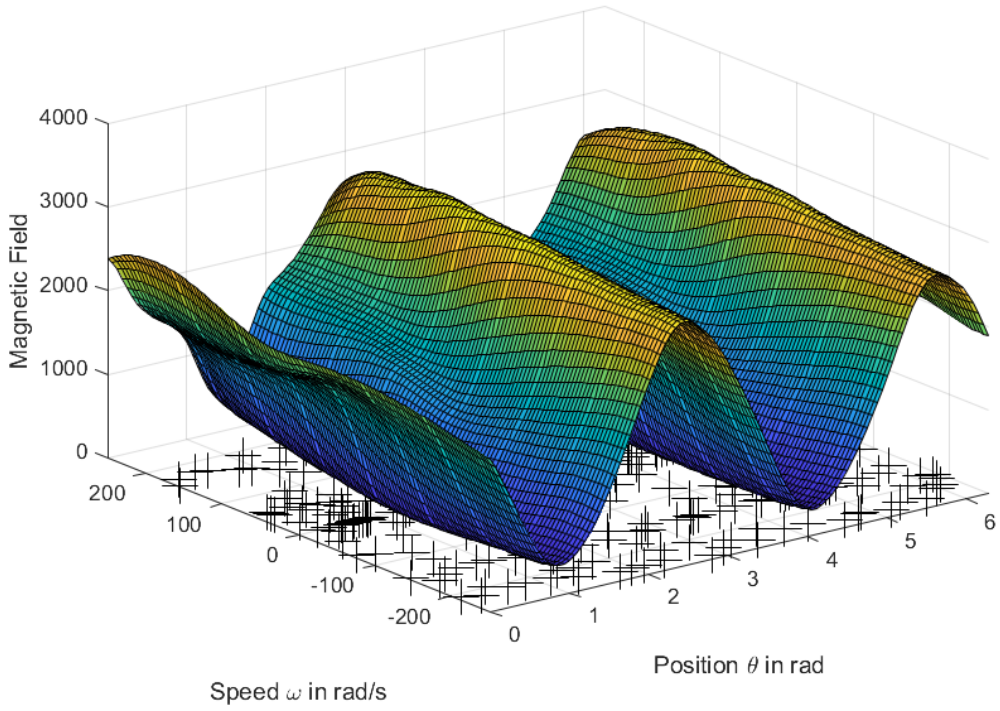
6.3.1 Squared Exponential Automatic Relevance Determination (SE-ARD)

Earlier we introduced the Squared Exponential Automatic Relevance Determination Kernel (SE-ARD). The kernel is defined as

$$k(\underline{x}, \underline{x}') = \sigma_s \cdot exp\left(-\frac{1}{2}\sum_{d=1}^{n}\frac{(\underline{x}_d - \underline{x}'_d)^2}{l_d^2}\right),$$



(a) Magnetic field in x-axial direction.



(b) Magnetic field in y-axial direction.

Figure 6.7: Predictive mean of the SPGP model.

where $\underline{x} = \begin{pmatrix} \theta \\ \omega \end{pmatrix}$, and $\underline{x}' = \begin{pmatrix} \theta' \\ \omega' \end{pmatrix}$.

Since the EKF needs the derivative of the kernel function with respect to each input dimension, we will introduce it here. The details of why the derivative of the kernel is required, as well as how it is used, will be explained later in Section 6.4.
The SE-ARD derivative with respect to the angular position of the tested instance $\underline{x}'$ is

$$\frac{\partial k(\underline{x}, \underline{x}')}{\partial \theta'} = \frac{(\theta - \theta')\sigma_s^2}{l_\theta^2} \cdot exp\left(-\frac{(\theta - \theta')^2}{2l_\theta^2} - \frac{(\omega - \omega')^2}{2l_\omega^2}\right).$$

The derivative with respect to the rotational speed of the tested instance is

$$\frac{\partial k(\underline{x}, \underline{x}')}{\partial \omega'} = \frac{(\omega - \omega')\sigma_s^2}{l_\omega^2} \cdot exp\left(-\frac{(\theta - \theta')^2}{2l_\theta^2} - \frac{(\omega - \omega')^2}{2l_\omega^2}\right).$$

6.3.2 Custom Periodic Kernel (PER-ARD)

While no direct modelling assumptions aside from smoothness can be made about the dependence of the speed, the physical dependency of the magnetic field on the angular position is periodic. Therefore, we build a custom kernel utilizing the combination rules described in Section 5.2.1. Our custom kernel models both outputs, i.e. position and speed by a separate kernel, where SE is used for the speed, and a periodic kernel is used for the angular position. We then combine both of these kernels by multiplying them

$$k(\underline{x}, \underline{x}') = \sigma_s \cdot exp\left(-\frac{2sin^2\left(\frac{\theta - \theta'}{2}\right)}{l_\theta^2}\right) \cdot exp\left(-\frac{(\omega - \omega')^2}{2l_\omega^2}\right),$$

where $\underline{x} = \begin{pmatrix} \theta \\ \omega \end{pmatrix}$, and $\underline{x}' = \begin{pmatrix} \theta' \\ \omega' \end{pmatrix}$.

Analogically to the SE-ARD kernel, we will calculate the derivatives of the periodic kernel here, while their usage will be described at a later point.

The two derivatives with respect to each input dimensions are

$$\frac{\partial k(\underline{x},\underline{x}')}{\partial \theta'} = \frac{2\sigma_s^2(\theta-\theta')}{l_\theta^2|\theta-\theta'|}\cdot exp\left(-\frac{2sin^2\left(\frac{|\theta-\theta'|}{2}\right)}{l_\theta^2}-\frac{(\omega-\omega')^2}{2l_\omega^2}\right) \cdot cos\left(\frac{|\theta-\theta'|}{2}\right)\cdot sin\left(\frac{|\theta-\theta'|}{2}\right)$$

$$\frac{\partial k(\underline{x},\underline{x}')}{\partial \omega'} = \frac{2\sigma_s^2(\omega-\omega')}{l_\omega^2}\cdot exp\left(-\frac{2sin^2\left(\frac{|\theta-\theta'|}{2}\right)}{l_\theta^2}-\frac{(\omega-\omega')^2}{2l_\omega^2}\right).$$

6.4 The Extended Kalman Filter

In Section 4 we gave the theoretical background to the Extended Kalman filter for state estimation. In this section, we describe the specific configuration of choice for the filter. In this thesis we use the Nonlinear Estimation Toolbox [20] to implement the EKF in Matlab.

6.4.1 System Model

Firstly, we need to define a dependancy between the current state prediction and the previous state estimation

$$\underline{x}_k = \boldsymbol{A}_{k-1}\underline{x}_{k-1} + \boldsymbol{B}_{k-1}\underline{u}_{k-1} + \underline{\varepsilon}_{k-1}.$$

As the target of the estimation are the angular position θ and rotational speed ω, we can define the system state at the time step k as

$$\underline{x}_k = \begin{pmatrix}\theta_k \\ \omega_k\end{pmatrix}.$$

One can assume that the current angular position θ_k changes only based on the estimated rotational speed at time step $k-1$ and ΔT. Therefore,
$\theta_k^p = \theta_k^e + \Delta T \cdot \omega_{k-1}^e$. Furthermore, for a small time interval between measurements ΔT, the difference between ω_k and ω_{k-1} is neglectably small. Hence, we assume that the two values for the speed are equal and expect the small correction to

be applied by the measurement equation of the EKF. To summarize, the system equation describing our constant velocity model will be

$$\underline{x}_k^p = \begin{pmatrix} 1 & \Delta T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{k-1}^e \\ \omega_{k-1}^e \end{pmatrix} + \underline{\varepsilon}_{k-1}.$$

As $\underline{\varepsilon}_k \sim \mathcal{N}(\underline{0}, \mathbf{Q}_k)$, we need to define $\mathbf{Q}_k$ to specify the noise. To do that, we optimize the noise covariance matrix by testing our estimation setup with different values. The testing is done on the specifically assigned validation set.

6.4.2 Measurement Function

Next, we need to identify the measurement equation of the system. As noted before, the dependancy between the rotor state (angular position and rotational speed) and the magnetic field output is not linear. We define the measurement output vector $\underline{y}_k$ at the time step k as

$$\underline{y}_k = \begin{pmatrix} B_k^x \\ B_k^y \end{pmatrix} = \underline{h}(\underline{x}_k) + \underline{\xi}_k.$$

The measurement $\underline{h}(\underline{x}_k)$ is determined by the regression model. Since we consider two regression models f^x and f^y, one for each magnetic field axis, we define the function

$$\underline{h}(\underline{x}_k) = \begin{pmatrix} f^x(\underline{x}_k) \\ f^y(\underline{x}_k) \end{pmatrix}.$$

As introduced, GPs and SPGPs provide a probabilistic prediction of the output at specific input location. However, to define $\underline{h}(\underline{x}_k)$ we need to choose specific values for $f^x(\underline{x})$ and $f^y(\underline{x})$. We select $f^x(\underline{x}_k)$ and $f^y(\underline{x}_k)$ as the predictive mean value of the distribution $p(B_k^x \mid \underline{x}_k, \mathcal{D}^x)$ and $p(B_k^y \mid \underline{x}_k, \mathcal{D}^y)$ respectively.

Furthermore, we need to define the measurement noise $\underline{\xi}_k$. We already assumed zero mean noise which results in $\underline{\xi}_k \sim \mathcal{N}(\underline{0}, \mathbf{R}_k)$. To define the matrix $\mathbf{R}_k$, we need to measure the noise in each axial direction. The measurement noise represents the output noise of the regression models. It is set to be the standard deviation of the predicted output values from the ground truth values of the training data. This value is corresponding to the noise standard deviation hyperparameter of the kernel. Therefore, we define

$$\mathbf{R}_k = \begin{pmatrix} \sigma^x & 0 \\ 0 & \sigma^y \end{pmatrix},$$

where σ^x and σ^y are the noise standard deviation of the magnetic field in x-axial and y-axial direction respectively.

As a point of improvement, we considered using the variance of the predictive distribution for each time step to dynamically change the measurement noise. However, after calculating the GPs with SE-ARD variance for some simulated data, we did not observe a drastic change of the variance despite around the border values of θ and ω. On Figure 6.8 the predictive variance on the magnetic field in x-axial direction is displayed. While Figure 6.8a shows the variance with points on the full spectrum of values for θ and ω, Figure 6.8b restricts the border speed values to provide a better view. The magnetic field on the y-axial direction produces similar results.

Preliminary tests of the estimation model with dynamically changing measurement noise (based on the predictive mean) did not lead to improved results. Based on this, we decided to use a constant measurement noise in both directions. Furthermore, using constant measurement noise spares us the costly calculations ($\mathcal{O}(N^2)$ for a test instance) of the variance at each time step.

Furthermore, we need to also define the Jacobi-matrix $\mathbf{H}_k$, which is used for local linearization. The matrix is defined by the derivatives of each regression model with respect to each input dimension

$$\mathbf{H}_k = \begin{pmatrix} \frac{\partial f^x(\underline{x}_k)}{\partial \theta'} & \frac{\partial f^x(\underline{x}_k)}{\partial \omega'} \\ \\ \frac{\partial f^y(\underline{x}_k)}{\partial \theta'} & \frac{\partial f^y(\underline{x}_k)}{\partial \omega'} \end{pmatrix},$$

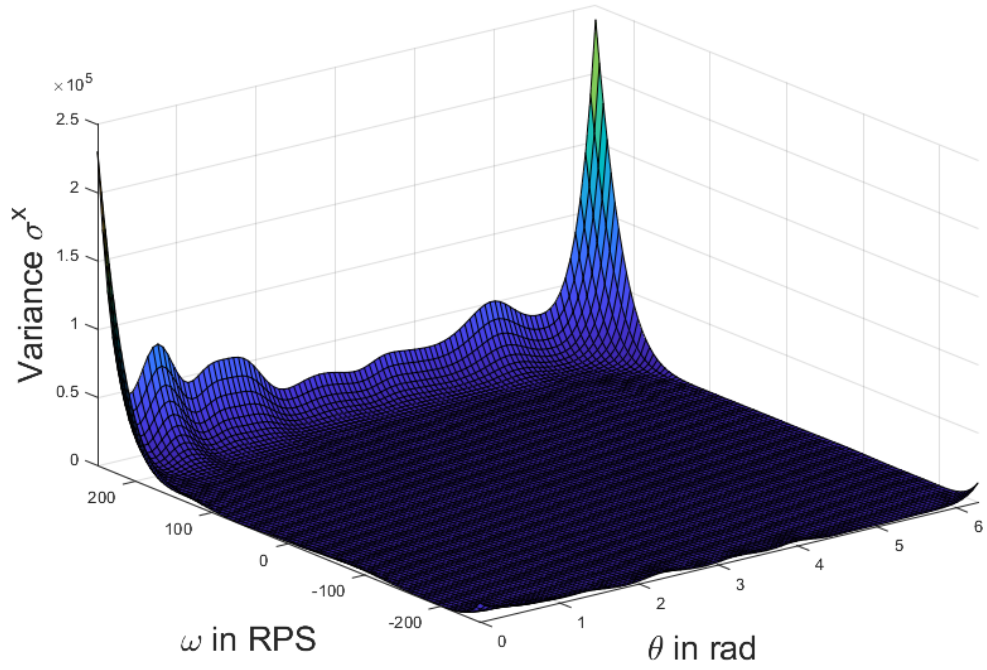
where $f^x(\underline{x}_k)$ is the predictive mean of the magnetic field in x-direction at the point $\underline{x}_k$ and f^y is the one in y-direction.

6.5 An Alternative Approach to Speed Estimation

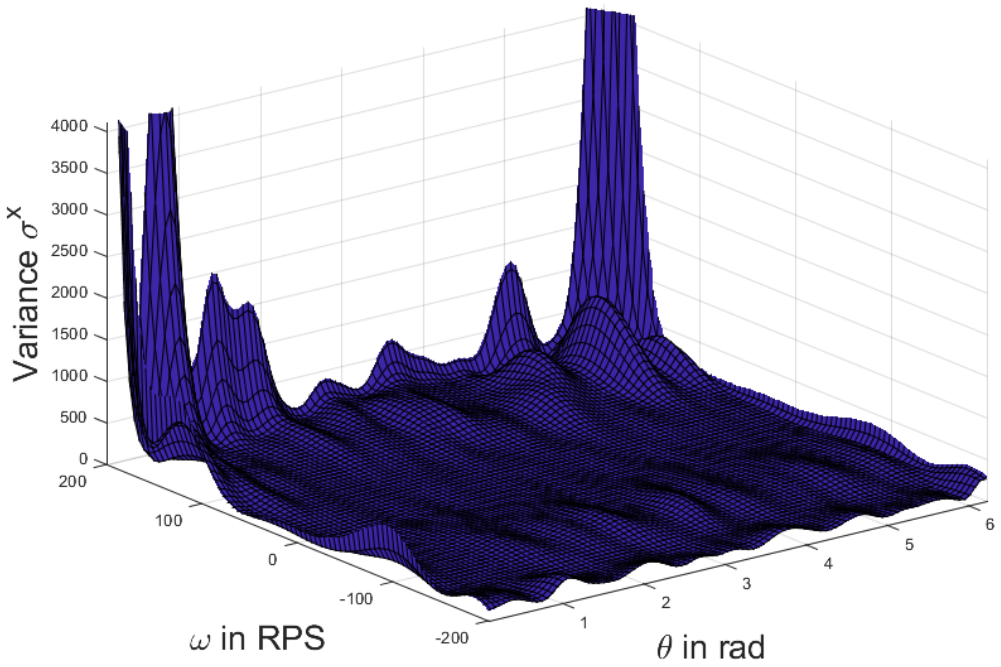
In [1], as well as in our models, we observed a much lower error on the angular position estimation than on the rotational speed. Therefore, we developed an additional approach to estimate the rotational speed by using the angular position estimates. Initially, we used the last and the second to last position estimate from the EKF and calculated the change in radians for the length of the time step ΔT as

$$\omega_k = \frac{\theta_k - \theta_{k-1}}{\Delta T}.$$

However, the data noise contribution, as well as the slightly unstable position estimate led to a very high error of the speed estimation.



(a) Variance on the full spectrum of values for the speed and the position.



(b) Variance when the border values for the speed are excluded.

Figure 6.8: Predictive variance on magnetic field in x-direction using the GPs with SE-ARD kernel.

To remove the position estimation noise and smoothen the results, we use the median filter. The reason we chose this filter is because of its robustness against outliers. The median filter is a filtering technique that computes an output sample by taking the median value of input samples under some window [21]. In our case we select this window to be of size 20 and calculate the speed (like in equation 6.5) between each chronological pair of the last 20 position estimations $(\theta^e_{k-19}, ..., \theta^e_k)$. This gives us 20 different values for the rotational speed, from which we need to choose one. The selection is done by first sorting the speed estimations and then choosing the median value as final speed estimation for the current time step k.

Given a median filter with size c (in our case $c = 20$), calculating the c number of rotational speed values is in $\mathcal{O}(c)$. Additionally, these values are sorted using the quick sort algorithm, which has best case $\mathcal{O}(c \cdot logc)$, average case of $\mathcal{O}(c \cdot logc)$ and worst case of $\mathcal{O}(c^2)$.

CHAPTER 7

Evaluation

In this chapter, we evaluate the accuracy of the models built on the test data (see Section 6.1.3). The evaluation is performed by calculating the prediction of the regression model (or the estimation model), and then comparing it to the ground truth value on the test data.

For this, the Root Mean Square Error (RMSE) metric is used

$$\mathbf{RMSE} = \sqrt{\frac{1}{N}\sum_{i}^{N}(prediction_i - truth_i)^2}$$

We first show the results for the regression models (on the magnetic field), and later evaluate the complete estimation model consisting of the regression model and the EKF.

7.1 Results of the Regression Models

The described regression models are evaluated using the RMSE metric for both, the x-axial and y-axial direction for the entire range of the rotational speeds. The corresponding RMSE values are shown under 'all speeds'. Furthermore, the models are evaluated at particular rotational speeds (in Rotation Per Minute - RPM) : ±100 RPM, ±200 RPM, ±500 RPM, ±1000 RPM, ±1500 RPM.

All described models, are tested on both, the original ($\mathcal{D}^x$, $\mathcal{D}^y$) and on the scaled data sets $\mathcal{D}^{x,scaled}$ and $\mathcal{D}^{y,scaled}$. The results with the SE-ARD kernel can be found in the tables 7.1, 7.2 and 7.3 for the linear regression, the GP and the SPGP models respectively.

One can see, that the linear regression models does not show significant difference in accuracy between the original and rescaled data. However, a small improvement in the regression for smaller rotational speeds is noticeable.

Linear regression, SE-ARD			
	Tested speed ω_{test}/RPM	**RMSE(B^x)**	**RMSE(B^y)**
original data	all speeds	20.3572	20.3436
	100	26.6053	27.7138
	-100	26.6358	27.5146
	200	27.3728	11.4616
	-200	32.7014	30.6205
	500	14.573	16.4511
	-500	18.789	19.0015
	1000	20.4266	19.2807
	-1000	17.2019	19.9658
	1500	13.2834	15.0703
	-1500	11.8905	12.3322
rescaled data	all speeds	20.4093	21.537
	100	24.9558	23.6075
	-100	24.4855	23.8256
	200	26.1694	10.7595
	-200	31.4702	30.2521
	500	13.2536	16.3365
	-500	16.2811	17.3904
	1000	21.1124	22.4258
	-1000	16.6563	21.2493
	1500	14.8652	16.6138
	-1500	14.7033	16.3391

Table 7.1: Error on the linear regression model, using SE-ARD.

GP, SE-ARD			
	Tested speed ω_{test}/RPM	**RMSE(B^x)**	**RMSE(B^y)**
original data	all speeds	24.0851	29.4115
	100	31.3455	29.5093
	-100	31.9589	34.0501
	200	31.2527	12.6483
	-200	37.043	36.1389
	500	16.1633	20.4691
	-500	19.1951	21.2171
	1000	30.3442	26.471
	-1000	18.9244	18.5268
	1500	25.6991	32.4093
	-1500	12.8694	13.2978
rescaled data	all speeds	22.738	23.3306
	100	30.7039	31.7805
	-100	27.3712	31.1648
	200	31.0714	13.5354
	-200	33.5892	33.3805
	500	16.9664	20.8776
	-500	28.4761	20.529
	1000	22.9061	21.9617
	-1000	19.0699	19.0411
	1500	15.5473	19.9976
	-1500	13.1341	12.9673

Table 7.2: Error on the GP regression model, using SE-ARD.

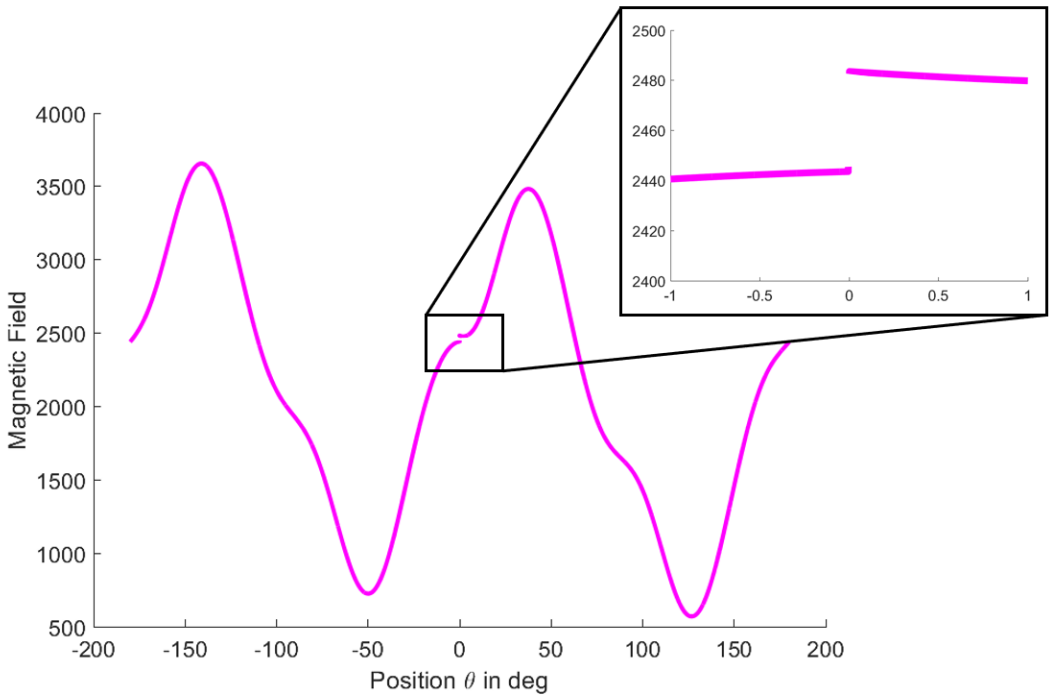
SPGP, SE-ARD			
	Tested speed ω_{test}/RPM	**RMSE(B^x)**	**RMSE(B^y)**
original data	all speeds	20.0253	20.8206
	100	25.1472	29.3937
	-100	24.5395	29.8419
	200	26.4798	12.2894
	-200	31.2337	32.2373
	500	14.507	16.7967
	-500	17.0268	18.4604
	1000	19.812	20.8038
	-1000	15.2159	17.3848
	1500	14.447	14.6305
	-1500	12.134	12.3024
rescaled data	all speeds	20.2326	21.058
	100	25.9554	29.0621
	-100	25.4629	29.4862
	200	26.8038	12.1431
	-200	31.6549	31.9225
	500	14.8311	16.8595
	-500	18.0652	18.3476
	1000	20.3878	20.5122
	-1000	15.5858	17.5701
	1500	14.5778	15.3909
	-1500	12.0724	12.3209

Table 7.3: Error on the SPGP regression model, using SE-ARD.

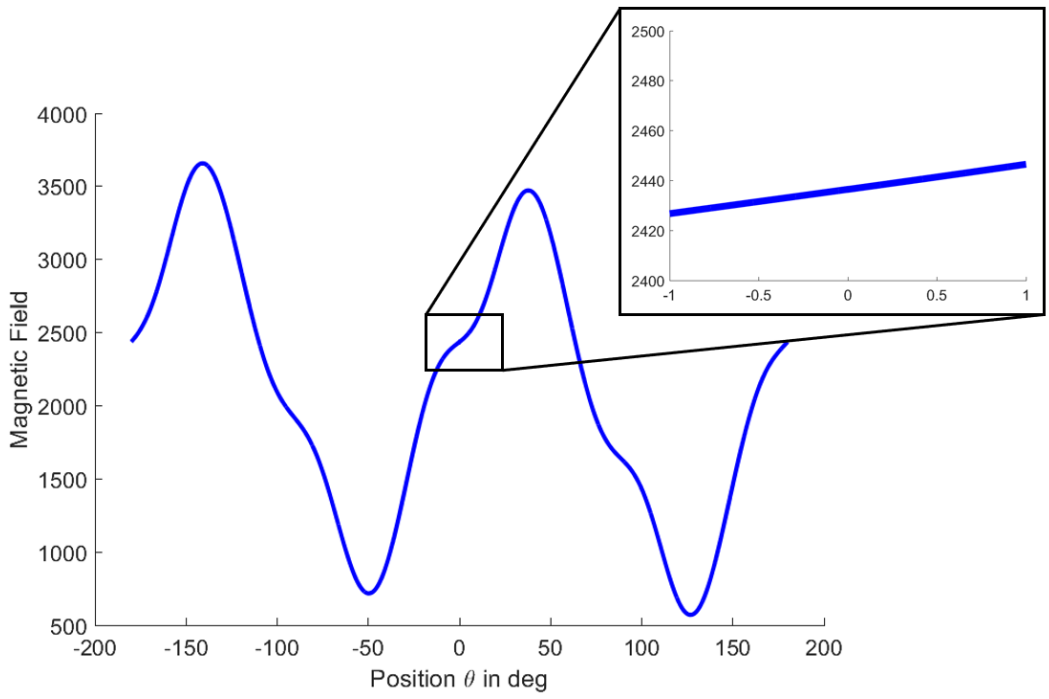
The rescaled data, on the other hand, shows improvement for the GP model. The overall RMSE of the magnetic field in x-direction RMSE(B^x) drops from 24.0851 to 22.738, while the one in y-direction changes from 29.4115 to 23.3306. The overall improvement seems to be mainly due to the smaller RMSE in the prediction for higher rotational speeds. Similar to linear regression models, the SPGP models also do not seem to show improvement with the change of the training data set.

A disadvantage of the SE-ARD kernel is that it does not consider the periodicity of the magnetic field with respect to the angular position. Therefore, the regression model is not smooth, but disconnects between 359° and 0°. This effect can be seen on Figure 7.1a, where the magnetic field (blue) in x-axial direction is displayed for a speed of 100 RPM. However, this flaw can be addressed through the kernel constructed in Section 6.3.2. Using this kernel, the periodicity of the angular position can be modelled correctly (see Figure 7.1b).

In Table 7.4 and 7.5, the evaluations of the models with the PER-ARD kernel are displayed. It can be seen that the GP benefits slightly from rescaling the training set, while the linear regression model shows worse results.



(a) GP with SE-ARD kernel.



(b) GP with PER-ARD kernel.

Figure 7.1: Predictive mean on the GP model in the x-axial direction with speed of 100 RPM. The displayed GP model was trained on the $\mathcal{D}^x$ data set.

Linear regression, PER-ARD			
	Tested speed ω_{test}/RPM	**RMSE(B^x)**	**RMSE(B^y)**
original data	all speeds	19.9605	19.4425
	100	25.7811	24.8649
	-100	25.5017	25.1204
	200	26.5468	10.6162
	-200	32.1711	29.5502
	500	14.385	16.0423
	-500	18.0409	16.9649
	1000	19.6105	19.6897
	-1000	15.9792	16.6833
	1500	13.2177	14.7046
	-1500	11.7613	12.3884
rescaled data	all speeds	19.9998	21.0704
	100	24.0646	22.9453
	-100	23.2886	22.8068
	200	25.2367	10.6025
	-200	31.3607	29.7382
	500	12.3886	15.6523
	-500	15.6701	16.5481
	1000	19.2053	21.7803
	-1000	15.4185	19.3591
	1500	14.7323	16.3967
	-1500	14.7827	16.4364

Table 7.4: Error on the linear regression model, using PER-ARD.

GP, PER-ARD			
	Tested speed ω_{test}/RPM	**RMSE(B^x)**	**RMSE(B^y)**
original data	all speeds	23.0706	23.8307
	100	29.6973	25.9605
	-100	31.3473	30.2951
	200	30.0944	11.5002
	-200	36.5908	33.6804
	500	15.7344	18.8311
	-500	18.3456	23.3978
	1000	31.3557	26.2573
	-1000	19.645	20.9683
	1500	21.28	20.9987
	-1500	12.5847	13.0525
rescaled data	all speeds	22.7687	23.25
	100	29.6912	25.937
	-100	31.3093	30.3125
	200	30.0704	11.4914
	-200	36.5897	33.6741
	500	15.725	18.843
	-500	18.3832	23.3529
	1000	30.1384	24.9513
	-1000	19.5277	20.9763
	1500	22.9405	21.6091
	-1500	12.6051	13.0391

Table 7.5: Error on the GP regression model, using PER-ARD.

7.2 Results for the State Estimation

So far, we have evaluated the regression models for the magnetic field estimation. In this section we analyse the performance of the state estimation models. This includes evaluating the accuracy of the EKF, as well as the EKF with the additional speed estimation. The different regression models will thereby be used as the measurement function. All evaluations are done on the full data set as well as on groups for different rotational speeds. Furthermore, the times required to calculate a single test prediction are compared.

From now on we will define the RMSE of the angular position as RMSE(θ) and rotational speed as RMSE(ω). As described before, the rotational speed can be estimated directly from the EKF, or post-processed from the angular position estimates, by using the median filter. Therefore, the tables of evaluation have two columns for the speed estimation.

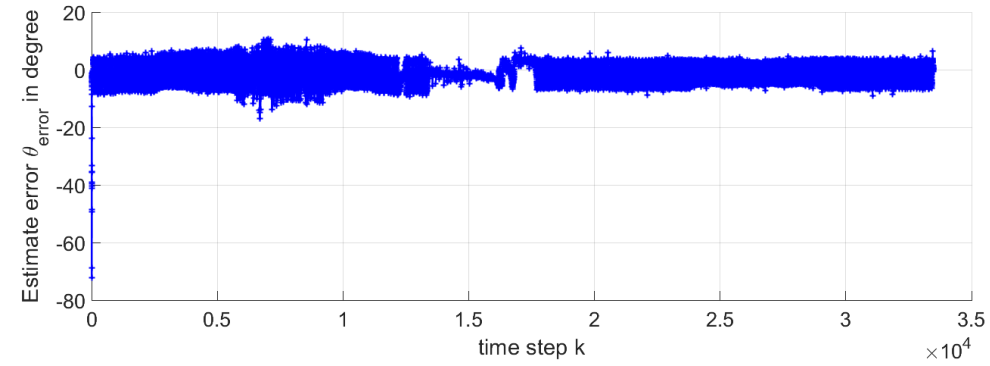
In the working process, we observed cases where the estimation model jumps on the estimation of 180° or 360° degrees. A possible reason for such bias can be that the regression using SE-ARD kernel do not produce a smooth output between 359° and 0°. Another possible cause could be if the model is tested on data, containing instances that are not equidistant in time, or taken with a low sampling rate. The reason for that would be the higher time step ΔT and therefore decaying dependancy of the current position given the last position and speed, as well as the similarity of the values of the magnetic field for angular position $\theta \in [0, 180°)$ and $\theta \in [180, 360°)$. The estimation model would then be especially volatile at higher rotational speeds.

However, electrical motors only consider position values in the range of [0°, 180°). Therefore, the error of the angular position estimate in such cases is calculated based on the angular difference with the ground truth on 180° instead of on 360°. In the evaluation tables, such cases is indicated with a (*).

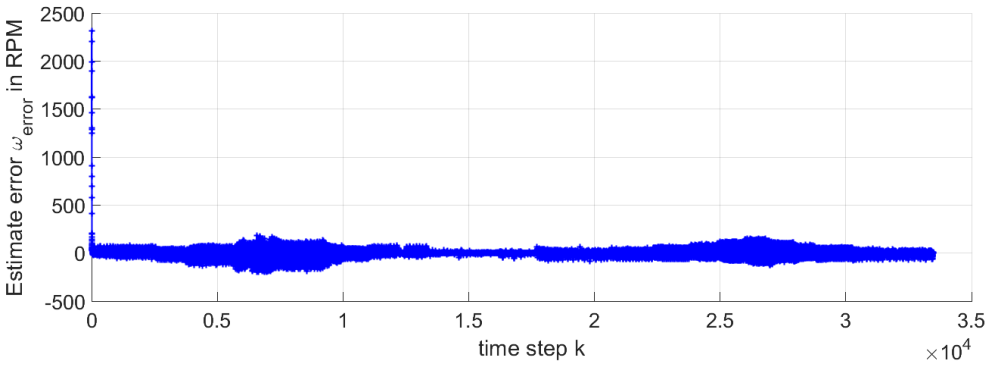
7.2.1 State Estimation with Regression Models with SE-ARD kernel

In Tables 7.6, 7.7 and 7.8 we can see the evaluation of all estimation models, using a regression model with the SE-ARD kernel. The different combinations of training data for the regression models, as well as the two options for speed estimation span 12 different estimation models in total.

The RMSE(θ) of the angular position is not significantly different when comparing the estimation models using the original and the rescaled data for training the regression model. For example, the error overall speeds when using the linear re-



(a) Error on the angular position estimation.



(b) Error on the rotational speed estimation.

Figure 7.2: Accuracy of the state estimation of the EKF, using linear regression with SE-ARD kernel.

gression model is RMSE(θ) = 3.1219 for the original data and RMSE(θ) = 3.1188 for the rescaled one. This error of the entire testing set does not include the first 50 testing values. The reason for that is the high error the EKF produces until it goes on track with the estimation. Visualisation on the state estimation of the EKF using linear regression with SE-ARD kernel can be seen in Figure 7.2.

If we direct our attention towards the error of the rotational speed RMSE(ω), we can observe bigger diversity of results. The speed estimation shows better performance on all models, when using the original data instead of the rescaled one. There is also a noticable difference in the results, depending on whether the speed was directly estimated by the EKF, or post-processed afterwards. The alternative approach to speed estimation seems to have a worse overall performance than the EKF when the original training data is used to build the regression models. However, it is noticeable that the performance of this approach on low rotational speeds is comparable, if not better at times, than the standard EKF. Therefore, the high overall error stems from a significantly smaller accuracy on the estimation of higher speeds.

When looking at the estimation models on the rescaled data, a huge improvement

of the speed estimation by using the alternative approach is visible. Even though the model with the median filter outperforms the basic EKF estimation, it still has higher RMSE than models, trained on the original data.

Finally, from the 12 estimation models that use SE-ARD kernel for the regression, the ones using the original data and the standard EKF show the best overall results.

7.2.2 State Estimation with Regression Models with PER-ARD kernel

The next group of estimation models use the PER-ARD kernel for constructing the regression models. Their detailed evaluation can be found in Tables 7.9 and 7.10. We will first analyse the performance on the angular position estimation and then review the speed estimation with the two speed estimation approaches.

Similar to the estimation models using SE-ARD, training linear regression with the rescaled data does not show a considerable change. However, one can notice a significant difference in the GP performance when changing the training set. As shown on Table 7.10, the overall error of the position RMSE(θ) drops by half when the rescaled training set is used.

As for the speed estimation, all models using only the EKF for estimation perform better on the original data than on the rescaled one. Furthermore, the speed estimation, enhanced by the median filter shows worse results when the original data is used for training. However, similarly to the models with SE-ARD, this alternative approach gives smaller RMSE(ω) on the speed when the rescaled data is used.

As it can derived from the tables, under all the models that use the PER-ARD kernel, the one showing the best speed estimation is the one using GPs with the original data and takes the predictions directly from the EKF. However, a significant drawback of this model seems to be the trade-off between good angular position and speed estimation. Unlike most of the other models, the GPs using $\mathcal{D}^x$ and $\mathcal{D}^y$ have double the position error. The second-best performing speed estimation model of this group is linear regression, using the original training data and direct EKF output without the median filter.

In conclusion, as we require both a good position and speed estimation, we consider the behaviour of the GP with PER-ARD on the original data unfavorable. Therefore, the prefered model with periodic kernel is the one using linear regression, with overall error in the angular position of RMSE(θ) = 3.0831, and in the speed of RMSE(ω) = 52.5253.

Linear regression, SE-ARD				
	Tested speed, ω_{test}/RPM	**RMSE(θ)/deg**	**RMSE(ω)/RPM**	**RMSE(ω)/RPM, calculated from previous positions**
original data	all speeds	3.1219	36.2002	54.3772
	100	3.4869	20.3897	15.1950
	-100	2.9095	19.6732	14.8198
	200	3.1138	21.3653	23.5046
	-200	3.0927	22.7065	20.4847
	500	2.5006	22.5987	35.2152
	-500	3.3974	28.9835	36.9025
	1000	2.8582	36.8392	56.5344
	-1000	2.8524	39.4783	62.1423
	1500	3.5472	54.2243	79.3536
	-1500	2.9244	53.2678	87.0477
rescaled data	all speeds	3.1188	114.7342	54.1630
	100	3.3761	30.14	15.1924
	-100	2.7796	23.3522	15.5806
	200	2.9245	46.8398	21.6608
	-200	3.0636	53.7893	20.0186
	500	2.7385	141.878	35.6236
	-500	3.1653	92.2313	34.8199
	1000	3.029	131.6414	53.7278
	-1000	2.955	159.4561	61.11311
	1500	3.7763	128.1984	77.5723
	-1500	2.8929	163.6278	85.6375

Table 7.6: Error on the state estimation model using the linear regression model with SE-ARD kernel on data set B.

GP, SE-ARD				
	Tested speed, ω_{test}/RPM	**RMSE(θ)/deg**	**RMSE(ω)/RPM**	**RMSE(ω)/RPM, calculated from previous positions**
original data	all speeds	3.3207	37.1094	52.9751
	100	3.5463	21.28	15.3225
	-100	3.0249	17.3466	13.9967
	200	3.1956	11.7355	22.0366
	-200	3.0366	16.9609	18.3847
	500	2.477	19.8413	32.6535
	-500	3.2706	23.537	30.8535
	1000	2.7623	33.9785	58.8090
	-1000	2.9279	42.5576	62.9820
	1500	4.5514	50.1125	75.6725
	-1500	2.8308	57.7483	75.9825
rescaled data	all speeds	3.0862	73.845	50.8665
	100	3.4636	22.1679	15.6073
	-100	2.9629	20.5727	16.7883
	200	3.0311	28.0375	19.8142
	-200	3.1957	34.4186	20.0386
	500	2.4549	81.1739	30.9767
	-500	3.444	53.7529	30.9497
	1000	3.0112	77.7788	52.6519
	-1000	2.9164	94.456	55.1143
	1500	2.8572	81.0658	61.2132
	-1500	2.9424	104.1294	86.4795

Table 7.7: Error on the state estimation model using the GP regression model with SE-ARD kernel on data set B.

SPGP, SE-ARD				
	Tested speed, ω_{test}/RPM	**RMSE(θ)/deg**	**RMSE(ω)/RPM**	**RMSE(ω)/RPM, calculated from previous positions**
original data	all speeds	2.773	36.2694	50.3445
	100	3.3208	19.127	15.1131
	-100	2.8569	18.58	15.5186
	200	2.9006	19.7593	23.5654
	-200	3.0173	21.3021	18.8087
	500	2.1953	21.0527	31.5452
	-500	3.1085	27.2406	35.4852
	1000	2.306	35.2732	52.7030
	-1000	2.5536	41.3773	60.5473
	1500	2.499	50.4356	69.9738
	-1500	2.4576	57.3548	77.3295
rescaled data	all speeds	2.7836	43.0006	42.6009
	100	3.4076	15.1149	16.9204
	-100	2.8691	18.7284	16.6091
	200	2.9487	11.6843	19.2379
	-200	2.9255	18.6661	19.5287
	500	2.1551	28.625	28.0710
	-500	2.9468	26.1075	36.8502
	1000	2.2431	39.1521	46.2572
	-1000	2.5637	49.7271	56.4043
	1500	2.4296	51.4058	49.6067
	-1500	2.4416	65.7764	61.8658

Table 7.8: Error on the state estimation model using the SPGP regression model with SE-ARD kernel on data set B.

Linear regression, PER-ARD				
	Tested speed, ω_{test}/RPM	**RMSE(θ)/deg**	**RMSE(ω)/RPM**	**RMSE(ω)/RPM, calculated from previous positions**
original data	all speeds	3.0831	52.5253	54.0362
	100	3.5789	98.9172	16.7319
	-100	2.9985	18.9374	16.3649
	200	3.0418	12.2103	24.3805
	-200	3.1408	19.2956	21.5406
	500	2.4067	30.6432	33.3664
	-500	3.2423	26.8993	36.9437
	1000	2.6962	40.1217	55.7926
	-1000	2.7515	50.7311	63.5946
	1500	3.3082	52.3184	78.4468
	-1500	2.8361	66.4689	83.9004
rescaled data	all speeds	3.0979	101.8811	56.0206
	100	3.3319	28.0665	15.6875
	-100	2.7606	22.2979	16.7428
	200	2.8861	41.1872	22.1032
	-200	3.0575	47.9421	20.4974
	500	2.6382	123.8555	36.9728
	-500	3.1254	80.472	37.0621
	1000	2.8383	115.7574	54.2087
	-1000	2.9185	138.945	65.9100
	1500	3.8519	113.9956	77.1367
	-1500	2.8705	144.6743	87.0702

Table 7.9: Error on the state estimation model using the linear regression model with PER-ARD kernel on data set B.

GP, PER-ARD				
	Tested speed, ω_{test}/RPM	**RMSE(θ)/deg**	**RMSE(ω)/RPM**	**RMSE(ω)/RPM, calculated from previous positions**
original data	all speeds	7.6937	39.5337	40.2053
	100	8.7427	28.8024	31.5822
	-100	9.5521	27.4853	30.1910
	200	3.5794	12.7425	14.8403
	-200	8.1427	32.9498	36.1480
	500	3.897	20.6941	20.9687
	-500	3.738	24.8161	25.8333
	1000	3.7795	35.2253	35.1609
	-1000	4.7694	44.4829	43.6638
	1500	4.0281	47.3751	46.6995
	-1500	4.5266	59.7772	58.9859
rescaled data	all speeds	3.4066	70.2517	55.5344
	100	3.4402	21.8498	15.2240
	-100	2.8641	20.3365	16.6019
	200	3.0491	26.2229	19.2910
	-200	3.0925	32.5974	20.5512
	500	2.6318	75.3846	39.7674
	-500	3.4712	50.2378	35.1630
	1000	3.7626	72.8072	60.7269
	-1000	3.0516	88.7589	59.0359
	1500	4.4826	77.0799	65.9718
	-1500	3.2509	99.0775	102.5381

Table 7.10: Error on the state estimation model using the GP regression model with PER-ARD kernel on on data set B.

Estimation models with EKF						
	GP			**Linear regression**		**SPGP**
	SE	SE-ARD	PER-ARD	SE-ARD	PER-ARD	SE-ARD
time in ms for a single prediction	0.39864	0.40315	0.52947	0.40358	0.53688	0.78932
Estimation models with EKF, the speed the is calculated using previous positions						
	GP			**Linear regression**		**SPGP**
	SE	SE-ARD	PER-ARD	SE-ARD	PER-ARD	SE-ARD
time in ms for a single prediction	-	0.56670	0.69345	0.57436	0.69144	0.9944

Table 7.11: Computational time for a single estimation.

7.2.3 Computational Costs

Another factor of the models we need to consider is the computational cost to perform a prediction for a single test instance. The factors influencing this calculation time are the number of training instances for the GP, the number of basis functions for linear regression and the number of pseudo-inputs for the SPGP. Furthermore, the estimation models that calculate the speed with the median filter require additional computational time.

In Table 7.11 the computational time of all estimation models is displayed. It is to be observed, that for a median filter of size 20, an additional time of around 0.166 ms is needed for the calculation of each prediction. Furthermore, the results show the higher complexity of the PER-ARD kernel, which results in a higher computational cost of around 1.2 ms compared to the SE-ARD.

Finally, the EKF models that use GP and linear regression with SE-ARD have comparable time performance to the previous work - GP with SE [1].

7.2.4 Summary and Comparison with the Previous Work

Thus far, we presented a detailed evaluation of all regression and estimation models we built. In order to compare our models to the estimation method using EKF and GP, described in [1], we trained the model with our training data.

From the previously evaluated estimation models, we select the best performing ones and compare them to the best performing model in [1]. All of the models are evaluated both data set A and B where the RMSE of the angular position RMSE(θ) and rotational speed RMSE(ω) are calculated. This comparison can be seen in Tables 7.12 and 7.13 for the data set A and B respectively.

The first conclusion one can draw from the evaluation is that all models using SE-ARD kernel perform better on the speed evaluation, and better or comparably good at the position estimation. Nevertheless, abnormalities can be observed in some cases, namely with the model using GP with SE-ARD kernel.

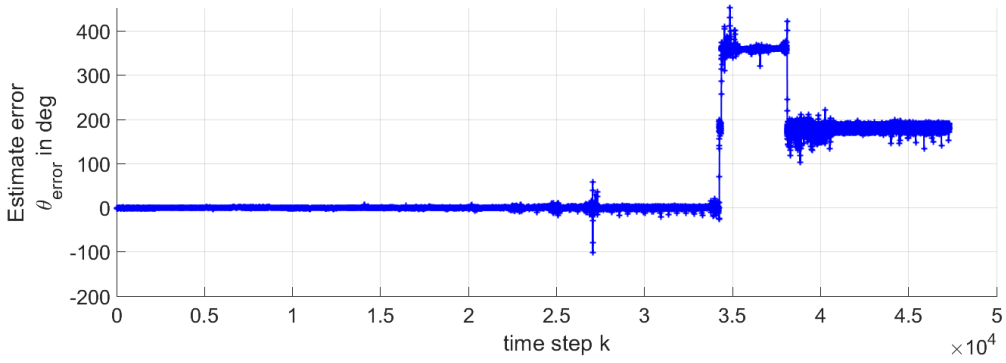
As we previously said, the prediction of the Kalman filter is sometimes shifted by 180°. Since the electrical setup only considers angles up to 180°, in the cases of such shits we consider the estimation $\theta \; mod \; 180°$. To illustrate the difference between the original position error and the one on 180°, we created Figure 7.3. Finally, the error of the wrapped to 180° estimation is comparable to the errors of the other estimation models.

	Tested speed in RPM	GP SE [1]	Linear regression SE-ARD	GP SE-ARD	SPGP SE-ARD	Linear regression PER-ARD
RMSE(θ)/deg	all speeds	3.8333	1.7780	3.5818*	2.3078	1.7582
RMSE(ω)/RPM		156.8229	143.6718	142.8802	143.0210	159.2587
RMSE(ω)/RPM tested on data set A	100	19.1855 ≈ 19%	21.1074 ≈ 21%	33.6303 ≈ 34%	21.3321 ≈ 21%	61.1583 ≈ 61%
	-100	19.0687 ≈ 19%	23.5678 ≈ 23%	48.6155 ≈ 49%	23.9872 ≈ 24%	93.6365 ≈ 94%
	200	35.5607 ≈ 18%	44.3521 ≈ 22%	76.7792 ≈ 38%	45.2284 ≈ 23%	143.9374 ≈ 72%
	-200	31.7461 ≈ 16%	41.4444 ≈ 21%	86.6955 ≈ 43%	42.4181 ≈ 21%	165.8885 ≈ 83%
	500	76.2431 ≈ 15%	72.5205 ≈ 14%	80.8137 ≈ 16%	71.9653 ≈ 14%	114.1803 ≈ 23%
	-500	76.6494 ≈ 15%	71.0236 ≈ 14%	79.7492 ≈ 16%	70.3204 ≈ 14%	117.3674 ≈ 23%
	1000	140.3101 ≈ 14%	121.7744 ≈ 12%	128.0981 ≈ 13%	119.9335 ≈ 12%	172.5581 ≈ 17%
	-1000	168.6133 ≈ 17%	150.5822 ≈ 15%	149.4238 ≈ 15%	150.6443 ≈ 15%	177.5078 ≈ 18%
	1500	231.4947 ≈ 15%	211.0442 ≈ 14%	198.1633 ≈ 13%	208.2188 ≈ 14%	197.2595 ≈ 13%
	-1500	196.7902 ≈ 13%	173.7358 ≈ 12%	162.4917 ≈ 11%	170.4043 ≈ 11%	162.0444 ≈ 11%

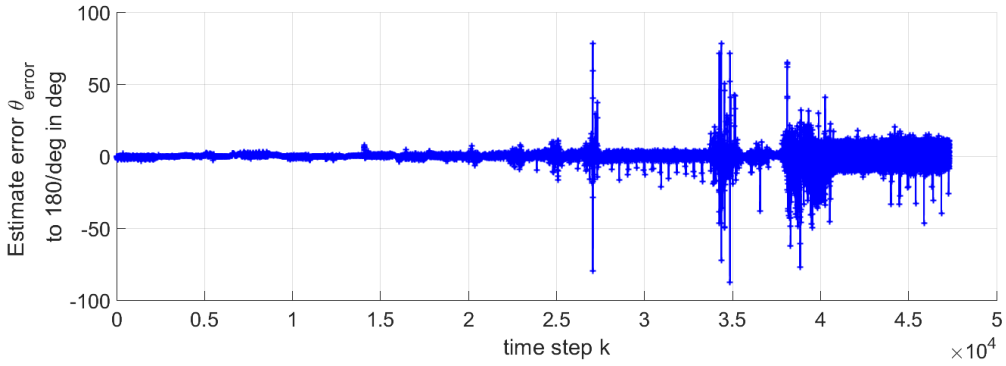
Table 7.12: Comparison between the estimation models using the EKF, without post-processing of the speed estimation, and regression models trained on data set A. The * indicates the results that produce different RMSEs of the angular position, depending on whether the ground truth and the estimation are wrapped on 180° or remain on the range of $[0°, 360°)$. In such cases, the RMSE on 180° is displayed.

	Tested speed in RPM	GP SE [1]	Linear Regression SE-ARD	GP SE-ARD	SPGP SE-ARD	Linear Regression PER-ARD
RMSE(ω)/deg	all speeds	3.1406	3.1219	3.3207	2.773	3.0831
RMSE(ω)/RPM		51.2665	36.2002	37.1094	36.2694	52.5253
RMSE(ω)/RPM tested on data set B	100	26.0839 $\approx 26\%$	20.3897 $\approx 20\%$	21.28 $\approx 21\%$	19.127 $\approx 19\%$	98.9172 $\approx 99\%$
	-100	30.5403 $\approx 31\%$	19.6732 $\approx 19\%$	17.3466 $\approx 17\%$	18.58 $\approx 19\%$	18.9374 $\approx 19\%$
	200	28.9891 $\approx 14\%$	21.3653 $\approx 11\%$	11.7355 $\approx 6\%$	19.7593 $\approx 10\%$	12.2103 $\approx 6\%$
	-200	42.3423 $\approx 21\%$	22.7065 $\approx 11\%$	16.9609 $\approx 8\%$	21.3021 $\approx 11\%$	19.2956 $\approx 9\%$
	500	39.6673 $\approx 8\%$	22.5987 $\approx 5\%$	19.8413 $\approx 4\%$	21.0527 $\approx 4\%$	30.6432 $\approx 6\%$
	-500	54.9531 $\approx 11\%$	28.9835 $\approx 6\%$	23.537 $\approx 5\%$	27.2406 $\approx 5\%$	26.8993 $\approx 5\%$
	1000	62.5437 $\approx 6\%$	36.8392 $\approx 4\%$	33.9785 $\approx 3\%$	35.2732 $\approx 4\%$	40.1217 $\approx 4\%$
	-1000	63.5969 $\approx 6\%$	39.4783 $\approx 4\%$	42.5576 $\approx 4\%$	41.3773 $\approx 4\%$	50.7311 $\approx 5\%$
	1500	69.7157 $\approx 5\%$	54.2243 $\approx 4\%$	50.1125 $\approx 3\%$	50.4356 $\approx 3\%$	52.3184 $\approx 3\%$
	-1500	76.035 $\approx 5\%$	53.2678 $\approx 4\%$	57.7483 $\approx 4\%$	57.3548 $\approx 4\%$	66.4689 $\approx 4\%$

Table 7.13: Comparison between the estimation models using the EKF, without post-processing of the speed estimation, and regression models trained on data set A.



(a) Error on the angular position estimation.



(b) Error on angular position estimation, wrapped on 180°.

Figure 7.3: Estimation error on the angular position of the GP model using SE-ARD.

CHAPTER 8

Conclusion and Future Work

In the field of robotics and control systems, having a reliable and accurate feedback system is essential for a high precision control. A desirable feedback system, in the case of BLDC motors, returns information on both the current angular position and rotational speed of the rotor.

In this work, we investigated several approaches to estimate the angular position and rotational speed. All methods combine magnetic field measurements taken from the backside of the motor with a state-space estimation model i.e. the Extended Kalman Filter. Ten regression models with different properties were evaluated as measurement functions for the Kalman filter. From them, twenty estimation models were evaluated.

Firstly, we addressed the idea of developing a model that scales better with the training data. For this purpose, we constructed linear regression models with basis functions, as well as models based on "Sparse Gaussian Processes using Pseudo-inputs" [2]. Both of these approaches can efficiently utilize big data sets by compressing its useful information in a precomputation step. Then, in the prediction step, only the compressed information is used.

We trained all models based on these two approaches with 2500 training points. This is triple the amount of training data the standard GP models work with. Moreover, the computational time for estimating a single test input with the linear regression stays similar to the one with the GP model.

Secondly, we identified a periodic kernel in order to properly model the periodical structure of the magnetic field with respect to the angular position. The results of both, the regression models alone, and the overall state estimation, did not improve by applying this change. However, due to the higher complexity of the kernel, a slight increase of computational time per estimation was observed.

Thirdly, as GP models are vulnerable to numerical errors when the regression inputs are on different scales, we also rescaled the training data. Rescaling the data showed

certain small improvements on the prediction of the regression models, but these were not transferred to improvement in the estimation.

Additionally, we introduced a post-processing step, that calculates the estimation speed from the previous positions using the median filter for smoothing. The usage of this method showed mixed results, but also increased the computational costs.

Finally, we identified that linear regression, the GP and the SPGP, in combination with the SE-ARD kernel all outperform the model from [1].

We identified several directions for future research. The first thing to be considered is the development of smarter basis points sampling technique for the linear regression. A second research direction, that can be of interest, is substituting the radial basis functions with trigonometric ones. This adaptation could lead to model improvements, given how the magnetic field changes with respect to the angular position. Another idea for future improvement of the estimation would be to eliminate the Kalman Filter in favour of a machine learning approach that considers previous observations i.e. models an internal state or context. An examples for such would be Long Short-Term Memory (LSTM) Neural Networks [22] or Time Delay Neural Networks (TDNN) [23]. Lastly, in contrast to the static data generation process employed in this work, an active learning framework [24] (e.g. uncertainty sampling) could be set up. Through this, the training points could be generated in a sequential manner in order to increase the model's performance.

Bibliography

[1] Fabian Sordon. *Position Estimation of BLDC Motors Using Gaussian Processes*. KIT, ISAS, 2019.

[2] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

[3] José Carlos Gamazo-Real, Ernesto Vázquez-Sánchez, and Jaime Gómez-Gil. Position and speed control of brushless dc motors using sensorless techniques and application trends. *sensors*, 10(7):6901–6947, 2010.

[4] Ekbert Hering, Gert Schönfelder, et al. Sensoren in wissenschaft und technik. *Auflage, Viewg+ Teubner Verlag, Wiesbaden*, 2012.

[5] Xuyang Liu, Chunhua Liu, and Philip WT Pong. Velocity measurement technique for permanent magnet synchronous motors through external stray magnetic field sensing. *IEEE Sensors Journal*, 18(10):4013–4021, 2018.

[6] Honeywell. 3-Axis Digital Compass IC HMC5883L, `https://github.com/TinyCircuits/TinyCircuits-TinyShield-Compass-ASD2613/raw/master/res/HMC5883L_3-Axis_Digital_Compass_IC_Datasheet.pdf`.

[7] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[8] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter, Lecture, Univ. North Carolina, Chapel Hill, NC, USA, `https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf`.

[9] Uwe D. Hanebeck. *Informationsverarbeitung in Sensornetzwerken, Lecture*. Karlsruhe Institute of Technology, 2019.

[10] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[11] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. IEEE, 2000.

[12] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[13] Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[14] CW Groetsch. The theory of tikhonov regularization for fredholm equations. *104p, Boston Pitman Publication*, 1984.

[15] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[16] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[17] David JC MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998.

[18] Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse gaussian process regression. Technical report, 2003.

[19] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (GPML) toolbox. *Journal of machine learning research*, 11(Nov):3011–3015, 2010.

[20] Jannik Steinbring. Nonlinear estimation toolbox. *https://bitbucket.org/nonlinearestimation/toolbox*, 2015.

[21] Moncef Gabbouj, Edward J Coyle, and Neal C Gallagher. An overview of median and stack filtering. *Circuits, Systems and Signal Processing*, 11(1):7–45, 1992.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[24] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.