Realtime Data-driven Learning and Model Predictive Control using Gaussian Processes for Dynamical Systems

J.F.J. Pollack

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Realtime Data-driven Learning and Model Predictive Control using Gaussian Processes for Dynamical Systems

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

J.F.J. Pollack

October 27, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Nowadays, machine learning (ML) methods rapidly evolve for their use in model-based control applications. Model-based control requires an accurate model description of the dynamical system to reassure the performance of the controller. Conventionally, this model description is retrieved from first-principles modelling which can be problematic if the system consists of high-order and/or time-varying dynamics. In these applications, ML methods may benefit because of their promising potential to model complicated system behaviour from data. In the field of ML methods for the modelling and control of dynamical systems, Gaussian processes (GPs) form an interesting opportunity. The ability of GPs to directly learn nonlinear system dynamics prevents huge costs and/or efforts when modelling complex systems. GP dynamical models are capable of accurately predicting the behaviour of dynamical systems while also measuring the confidence level of the prediction. For making predictions, GPs do not require huge amounts of data which benefits them over other ML methods.

The powerful model predictive control (MPC) and the fully data-driven dynamical modelling capabilities of GPs makes their combination an interesting candidate for sophisticated control systems. MPC has advantages over other control methods since the controller allows operational constraints that provide freedom in controller design or prevent the system to be steered in an infeasible direction, and, the controller is easily extendable to nonlinear and multivariable control. Next to this, an MPC controller is naturally modified to incorporate GP dynamical models (GP-MPC). The GP-MPC controller exploits the GP dynamical model for making predictions over the prediction horizon while it is also possible to incorporate the confidence of the predictions for increased robustness of the controller. Whereas GP-MPC is studied extensively as an augmented model for other modelling techniques, fully data-driven GP-MPC approaches are also deemed to be promising.

This thesis considers the use of GPs for learning and modelling dynamic systems for incorporation in a realtime MPC application. The dynamic system studied in this thesis is a double pendulum system in both a simulation and a real-world environment. The simulation provides initial insights into the problem and allows the rapid development of an algorithm. Eventually, the proposed realtime GP-MPC algorithm is tested on a physical laboratory-scale setup. The results show that the realtime data-driven controller is able to track a reference with high accuracy, while also being robust to disturbances. These promising results on GPs for realtime nonlinear control might be a step for GPs to be incorporated into future control systems.

Table of Contents

Acknowledgements

My last years of study were particularly unusual due to the COVID-19 pandemic. Nevertheless, the progress I made and the chances I got during this time will benefit me for the rest of my professional career. For the sake of ending my five years of study at the TU Delft, I have written this thesis which was not there if the people around me had stopped pushing, believing, and advising me. Therefore, I would like to dedicate this section to thank the people that have helped me finish this thesis.

Before thanking my daily supervisor Dr. ir. Sebastiaan Mulders, I would like to thank my leading supervisor Prof. dr. ir. Jan-Willem van Wingerden with whom I have started this research. After having listened to the subjects I like in our first meeting, he came up with the subject of this thesis which I am grateful for. Also, in the meetings that followed he provided me with new insights into the subject that improved this thesis, and, he trusted me to end it well. Furthermore, I am thankful that he connected me to my daily supervisor Sebastiaan Mulders.

Secondly, I want to say many thanks to my daily supervisor Sebastiaan Mulders who closely monitored and helped me while writing this thesis. Sebastiaan gave me insights into the problem, helped me code Gaussian processes (GPs), and showed me how to use them for control. He also reviewed every chapter I have written, pushed me to improve the thesis, trusted me in the course of the project, and, listened and helped me at all times.

At last, I want to thank my parents and brother, girlfriend, and friends for their (emotional) support. Especially, I want to thank my brother who acted as a sparring partner and who gave me advice. Also, I want to thank my girlfriend who helped and supported me throughout my thesis.

Delft, University of Technology — J.F.J. Pollack
October 27, 2022

Chapter 1

Introduction

1-1 Background

Modern engineering practices seek smart, efficient, and autonomous solutions for their engineering challenges. Examples found in the industry are abundant as one can think about the automation in the industry where robots are used for tasks like welding, packaging, and cutting [1], smart and efficient energy management systems for saving energy [2] or the desire to make autonomous vehicles in the automotive industry [3]. In realizing today's engineering solutions, control systems play an important role. For instance, state-of-the-art robotic manipulators use control strategies such as intelligent PID control, robust control and adaptive control [1], and, autonomous vehicles use controllers like model predictive control (MPC) or fuzzy control [3]. Most of the time the controllers require a model to reassure the performance of the system like stability or safety [4].

In the last decade, the classification and regression techniques from the rapidly emerging field of machine learning (ML) see a high pace of progression [5]. Traditionally, ML techniques are used for learning static mathematical relationships from data, while it is in control required to use dynamic equations for reassuring performance. Since ML has proven to be powerful in many applications, the control community gained increasing interest in the use of ML in control applications for developing the capability of control systems. More specifically, using ML for retrieving dynamic relationships may offer significant benefits in the design of control systems [4].

The traditional control theory covers controllers for both linear and nonlinear systems where examples of linear control methods are PID control, optimal control, and robust control, and, examples of nonlinear control methods include Lyapunov-based controllers, backstepping controllers, and feedback linearization [6]. However, these control methods are model-based which means that an accurate model description of the system is required for performance guarantees of the controller. The model description is often retrieved from physics-based or first-principles modelling which creates problems if the system to be controlled is complex, large-scaled, and/or exhibits time-varying dynamics. In other words, the (first-principles)

modelling process is sometimes either impossible due to high-order system dynamics, economically not feasible because of the time and efforts required for obtaining accurate models, and/or it is necessary to deploy costly experts for updating the model from time-to-time [4], [6]. Luckily, solutions are offered by modern systems where data is becoming more abundantly available. This information paves the way for ML to be incorporated in control [6].

A popular control method for data-driven applications is MPC [7], [8]. In classic MPC, the future state of the system is calculated by optimizing the control input to the system over a time interval known as the prediction horizon. The future states are predicted by a dynamic model of the system. The MPC controller offers advantages over other control methods since the optimization problem allows constraints that increases the design freedom of the controller or prevent the system to be steered in an infeasible direction. Other attractive properties of the controller include the flexibility in the design of the objective function, the compatibility with many modelling approaches, and its ability to expand the controller to multivariable and nonlinear systems [9]. The adaptive nature of MPC also explains its popularity in machine learning applications.

When incorporating data-driven methods in MPC, the previously assumed dynamic model is replaced by a data-driven one. The benefit of using data-driven models in control is their ability to directly incorporate nonlinear dynamics in the model [10]. Examples of data-driven methods that are used in conjunction with MPC are neural networks, random forests, deep learning, support vector machines, and Gaussian processes [7]. However, multiple of these methods require vast amounts of data before an accurate model description is obtained which makes them computationally demanding and data inefficient, e.g., neural networks and deep learning [7]. Also, some methods cannot be applied in receding horizon control without adaptation of the model because they lack a closed-loop expression, e.g., random forests [4], [11]. However, an ML method that is directly implementable in a data-driven MPC controller without requiring to see huge amounts of data, are Gaussian processes (GPs). Even if the size of Gaussian process (GP) models grows to unmanageable size because of the availability of big data, GPs offer approximate modelling frameworks that reduce their computational burden while keeping the GP model sufficiently accurate [12]. Another advantage of GPs over other ML methods is their ability to provide reasonable predictions about the system while also giving a measure of uncertainty. This measure of uncertainty can be used for improving the robustness of the control system [4]. Therefore, in this work, we will study the use of GPs in a data-driven MPC framework for controlling dynamical systems.

In recent years, GPs are increasingly used in MPC applications (GP-MPC). In the literature, GP-MPC is mostly used for modelling the error dynamics of a nominal model to improve the MPC control performance. The nominal model is constructed based on first-principles modelling. Examples of such applications are found in the field of autonomous racing [13], [14], robotics [15], flight control [16] and vehicle control [17]. Next to modelling the error dynamics, GPs in an MPC problem are also used for generating the reference trajectory for a medical robot [18] and to model system dynamics partially [19], [20]. Since GP-MPC is studied extensively as an augmented model for other modelling and control techniques, it is worth looking into fully data-driven GP-MPC approaches which are also known as black-box modelling.

Fully data-driven approaches for GP-MPC are used for the control of the energy in a building and to predict the power demand [4]. Other fully data-driven GP-MPC examples are in

the field of chemical processes [21]–[23] and the control of an unknown nonlinear system [24]. However, the applications of fully data-driven GP-MPC mainly remain in simulation environments. It is therefore the aim of this thesis to investigate the practical real-time feasibility of a fully data-driven GP-MPC scheme on a physical lab setup.

1-2 Objective & Research Questions

The aim of this thesis is to learn a nonlinear dynamical model by making use of GPs for the application in an MPC framework. A fully data-driven and black-box modeling approach is employed for deriving the GP dynamic model. The GP-MPC algorithm is intended to be applied to a double pendulum system in both a simulation and a physical setup. The simulation environment is used for the development of the GP-MPC algorithm and to obtain the first results on the performance. For the aim of the research, the following research goal is formulated:

Research goal: Develop an integrated real-time control application and fully data-driven solution for the (nonlinear) dynamic modelling and predictive control by employing GPs for a double pendulum system.

In fulfilling the research goal, this thesis is split up into several subquestions. The first subquestion involves the modelling of dynamical systems with GPs which requires the specification of a kernel function and model structure. GPs use kernel functions to predefine the behaviour of the GP model, while the physical interpretation of the GP is specified by the model structure. It is therefore valuable to look into the different kernel functions and model structures that are able to model the dynamics of a double pendulum system for application in an MPC framework which results in the first subquestion:

Subquestion I: Which combination of model structure and kernel function is suitable for capturing the dynamics of a double pendulum? And what are the limitations and possibilities of the GP dynamical models to be incorporated in a model predictive control framework?

The second subquestion deals with the challenge of computationally demanding GP dynamic models. In GP theory, methods exist for approximating the full GP model that allows modelling a computationally efficient GP while remaining sufficiently accurate. These GPs are known as approximate GPs. The use of approximate GPs can bring advantages for accelerating the required calculations. Therefore, the second subquestion is the following:

Subquestion II: How is an approximate GP used for accelerating the involved calculations for constructing dynamical models? And is the performance of the model affected by the approximation?

The third subquestion involves bridging the GP-MPC problem from simulation to real-time control. Bringing the problem to reality involves challenges with respect to computational

time and the effectivity of the GP-MPC algorithm. This is especially the case for systems that require a higher control bandwidth, such as a double pendulum system. Therefore, the third subquestion is formulated as:

Subquestion III: How is an approximate GP employed in an MPC framework, such that it is capable of real-time control of a double pendulum system taking into account the computational time and effectivity of the GP-MPC algorithm?

The last subquestion is about implementing the eventually proposed GP-MPC algorithm in the real-world. Simulating the GP-MPC algorithm might have discrepancies with respect to the reality where other challenges may show up. Therefore, the fourth subquestion is formulated as:

Subquestion IV: How is the resulting GP-MPC algorithm implemented in the physical double pendulum system? And what performance can be observed?

1-3 Thesis outline

This section presents the outline of this thesis. Each paragraph gives a brief summary of the contents of each chapter.

Chapter 2 provides a theoretical background that serves as the backbone of this thesis. The chapter starts with the preliminaries of Gaussian process regression (GPR) to gain useful insights. After discussing the preliminaries, the chapter explains the GPR theory on the identification of dynamical systems, and, the approximation techniques for GPR that speed up calculations. The chapter is concluded by discussing the use of dynamical GP models in an MPC framework.

Chapter 3 outlines the methodology of this thesis which elaborates on the development of an GP-MPC algorithm. This involves an explanation of the several development phases in subjects such as data acquisition, implementation strategies, system specifications, control strategies, and validation of results. Also, the details of the physical double pendulum system are discussed in this section.

Chapter 4 presents the results of the proposed GP-MPC algorithm in both the simulation and the physical setup. The simulation provides the first results of the research questions, including a specification of a kernel function and model structure that is able to describe the dynamics of the double pendulum system. Also, approximate GPs are employed for speeding up the calculations of the GP, and the GP-MPC algorithm is tested on performance and computation time. After finishing the simulation part, the algorithm is tested on a real-world double pendulum setup.

Chapter 5 concludes the thesis by answering the research questions. Also, the possibilities and limitations of the used method are explained, and recommendations are proposed for future research.

Chapter 2

Theoretical background

In this chapter, a theoretical background is provided to acquire the knowledge that is needed to answer the thesis questions. For answering the questions about dynamical modelling with Gaussian processes (GPs) it is required to discuss the general and dynamical theory of GPs. Also, it is interesting to look into the approximate GPs for answering the question about computationally demanding Gaussian process (GP) models. Finally, the chapter discusses dynamical GP models for use in a model predictive control (MPC) framework for answering the questions about control. To acquire this necessary knowledge, the chapter is split up into four sections of which each section discusses a different part of the theory.

Section 2-1 deals with the preliminaries of Gaussian process regression (GPR). This section is meant to provide the basic knowledge of GPs that is needed to understand the remainder of this thesis. The subjects discussed in this section involve the initialization of the GP by specifying the GP priors, the modelling of the GP with the priors, and making predictions of (un)seen data with the GP model.

Section 2-2 introduces the theory on GPR for learning dynamical models. GPs are normally used in regression problems to learn static input-output mappings. However, in control, there is an interest in modelling dynamical systems which requires looking into dynamical mappings. The modelling of dynamical systems with experimental data is also known as system identification [25]. For system identification, it is required to define a model structure [26]. Therefore, this section discusses the dynamical model structures that are available for system identification with GPs. It is also briefly mentioned how GP models are used if only stochastic data is available.

Section 2-3 discusses GP models for reducing the amount of data used for modelling while still yielding an accurate model. This is helpful for accelerating the involved calculations. The reduced data GPs are also known as the approximate GPs [27]. However, the theory for GP approximation methods is extensive and not equally important for this thesis. Therefore, the most relevant methods for this thesis are discussed in this section which includes the prior and posterior approximation methods.

Finally, Section 2-4 introduces GP dynamical models for the use in an MPC framework. The goal of this thesis is to perform control based on GP models. For this purpose, an MPC

controller is used since this controller is known to perform well in a stochastic environment [28]. The section shortly introduces MPC controllers in general, and their use in a (stochastic) GP framework.

2-1 Preliminaries of Gaussian Process Regression

The fundamental theory on GP regression is outlined in this section. In regression problems, GPs fit a function to data by assuming that it follows a normal distribution. So, as with any probability distribution, a GP is fully defined by a mean μ and a covariance Σ. This is also mentioned in [29] that provides the following definition of GPs:

Definition 1 (Gaussian process)**.** Let $\mathcal{X}$ be a nonempty set, $\Sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel and $\mu : \mathcal{X} \to \mathbb{R}$ be any real-valued function. Then a random function $\mathbf{f} : \mathcal{X} \to \mathbb{R}$ is said to be a Gaussian process (GP) with mean function μ and covariance kernel Σ, denoted by $\mathcal{GP}(\mu, \Sigma)$, if the following holds: For any finite set $X = (x_1, \ldots, x_n) \subset \mathcal{X}$ of any size $n \in \mathbb{N}$, the random vector

$$\mathbf{f}_X = (\mathbf{f}(x_1), \ldots, \mathbf{f}(x_n))^\top \in \mathbb{R}^n$$

follows the multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean vector $\boldsymbol{\mu} = (\mu(x_1), \ldots, \mu(x_n))^\top$ and covariance matrix $\boldsymbol{\Sigma} = (\Sigma(x_i, x_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$.

To perform regression with GPs a couple of prior assumptions are required. These prior assumptions are the selection of a prior mean function and a prior kernel function. This selection is based on the available information on the unknown function in the regression problem. However, it is most often the case that no a priori information is available on this function. In this case, a common practice is to use a mean function of zero and a kernel function that is known to be descriptive for many functions. The assumption of taking a prior mean function of zero is always valid since it is possible to re-scale the data such that the mean is zero [30]. However, the selection of the kernel function requires more knowledge. It is also the most crucial part in setting up the GP regression problem since the kernel function predefines the behaviour of the model [12]. The kernel functions and their behaviour are therefore discussed in more detail in Section 2-1-1.

After defining the priors of the GP regression problem it is used to be combined with the acquired regression data by using the following Bayesian inference:

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}, \tag{2-1}$$

where the inference is used to combine the prior believes of a function, the observed data in the likelihood, and the evidence to obtain a posterior probability distribution [26]. The evidence is also known as the marginal likelihood of the function which acts as a normalising factor.

In a GP framework, Bayesian inference is used in two phases [26]. The first phase is the modelling phase where the inference is used to learn a GP model from acquired data. This is done by learning the hyperparameters of an a priori selected kernel function. The hyperparameters are the free variables in a kernel function which are learned by either maximizing the log

marginal likelihood [31] or by using Markov chain Monte Carlo (MCMC) for approximating the posterior distribution numerically [32]. Secondly, the inference is also used in the prediction phase. The prediction phase exploits the knowledge that is acquired in the modelling phase to make predictions at (un)seen input locations.

The remainder of this section is structured as follows. Section 2-1-1 further discusses the selection of the kernel function. Furthermore, Section 2-1-2 explains Bayesian inference for the modelling phase, and Section 2-1-3 explains the learning of the correct hyperparameters for obtaining an accurate model description. Finally, Section 2-1-4 discusses Bayesian inference for the prediction phase.

2-1-1 Kernel Functions

The covariance function, which is also known as the kernel function, is one of the free design choices in GP modelling. The function is used for constructing the covariance matrix of the GP, which represents the correlations between datapoints. The kernel function is only valid if it produces a positive semi-definite matrix. Almost every kernel function consists of parameters that are predefined by the input data from the dataset and free variables (hyperparameters). The hyperparameters determine the behaviour of the model, e.g., fast/slow fluctuation, or large/small function space [33]. It is therefore important to find the correct hyperparameters since the behaviour of the model should match the unknown function.

The kernel is of fundamental choice when modelling with GPs as it predefines the behaviour of the model, e.g., linear, periodic, or quadratic [12]. The choice of the kernel is based on the assumption of what characteristics and patterns are expected in the data [31]. For example, if one wants to model a periodic function a logical choice of the kernel would be to choose a periodic kernel. Next to periodic kernel functions, there are numerous kinds of kernel functions, and a few of them are discussed later.

GP kernel functions are categorized in two main categories: stationary and non-stationary [26]. Stationary kernels are used for processes from which the joint probability distribution is assumed to be invariant when shifted in time or space. Examples of such kernel functions are the squared exponential (SE) kernel, Matérn class kernels, and the periodic kernel. Alternatively, if it is assumed that the joint probability distribution changes if another input point is evaluated, a non-stationary covariance function is chosen. The linear kernel, polynomial kernel, or neural network kernel are examples of non-stationary kernels.

Because of the fact that not every kernel function is of importance to discuss in this thesis, a selection is made. The selection is based on its usefulness in modelling later on in this thesis. First of all, the linear kernel is discussed since this kernel helps in understanding of the GP problems. Secondly, the SE kernel is introduced as this kernel function is able to model a great variety of nonlinear functions, and, it is one of the most used kernel functions. The last kernel function that is introduced is the periodic kernel function for its ability to model periodic behaviour of functions or systems.

Linear Kernel

The linear kernel is a nonstationary kernel and is used when the function of interest is assumed to be linear or a combination of multiple linear functions. It is nonstationary due to the fact

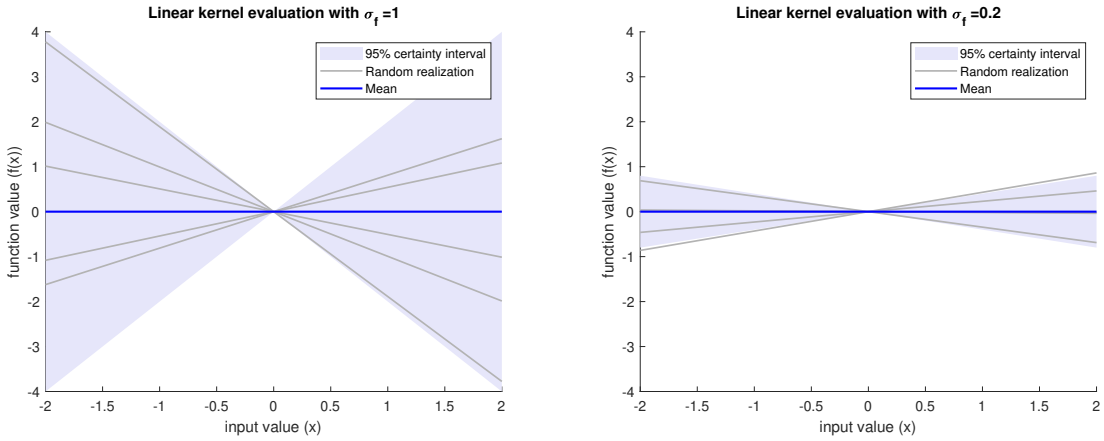


Figure 2-1: Visualization of the prior linear kernel. The grey lines in the plot are random realizations of functions that are in the function space of the kernel. Also, the 95% certainty interval is visualized which spans the function space which is most likely. The figures indicate that the function space changes if the variance hyperparameter is altered.

that it is not dependend on the relative distance between input points [33]. The kernel is defined by the following equation:

$$\Sigma_{\text{lin}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_{\text{f}}^2\left(\mathbf{x}_i \cdot \mathbf{x}_j\right), \tag{2-2}$$

where $\cdot$ indicates an element-wise multiplication and σ_f^2 the hyperparameter representing the variance of the function to be modelled, i.e., the function space.

The prior linear kernel is visualized in Figure 2-1. The figure retrieves that the function space becomes narrower if the hyperparameter σ_{f}^2 is adjusted to lower values.

The linear kernel is also suitable for using the automatic relevance determination (ARD) property [26]. The ARD property is useful if a regression vector $\mathbf{x}_i$ is provided to the GP for automatically adjusting the contribution of the regressor. This is useful for determining the relevance of the regressor for retrieving the unknown function. The ARD property is obtained by introducing a length scale hyperparameter l_i for each regressor. So, the amount of hyperparameters when using the ARD property in linear kernels is n_{x} if $\mathbf{x}_i \in \mathbb{R}^{n_{\text{x}}}$.

Now that there is known what the meaning of the ARD property is, it is interesting to see how this property appears in the kernel function. The linear kernel with ARD property is defined in the following:

$$\Sigma_{lin,ARD}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathbf{x}_i^T \mathbf{\Lambda}^{-1} \mathbf{x}_j, \tag{2-3}$$

with Λ is a diagonal vector of the lengthscale hyperparameters, i.e., $\Lambda = \text{diag}\left(\left[l_1^2, \ldots, l_{n_x}^2\right]\right)$.

Squared Exponential Kernel

The SE kernel is one of the most used kernels since it is able to model a great variety of nonlinear functions [26]. The kernel is categorized as a stationary kernel due to its dependency on the relative distance between two input points. The SE kernel is defined as follows:

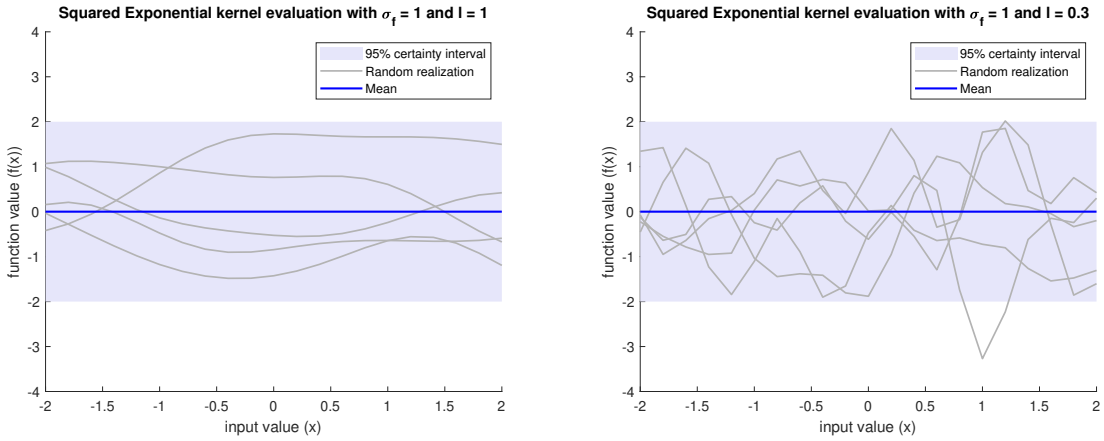


Figure 2-2: Visualization of the prior SE kernel. The grey lines in the plot are random realizations of functions that are in the function space of the kernel. Also, the 95% certainty interval is visualized which spans the function space which is most likely. The figures indicate that the smoothness of the function changes if the lengthscale hyperparameter is altered.

$$\Sigma_{\text{SE}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_{\text{f}}^2 \exp\left(-\frac{r^2}{2l^2}\right), \tag{2-4}$$

with $r = |\mathbf{x}_i - \mathbf{x}_j|$, $|\cdot|$ the norm operator, σ_{f}^2 the variance hyperparameter, and l^2 the lengthscale hyperparameter.

The SE kernel is visualized in Figure 2-2. The figure visualizes that adjusting the lengthscale hyperparameter influences the smoothness of the prior functions. The results of adjusting the variance hyperparameter hold the same as in the linear kernel case, i.e., the function space differs.

The SE kernel can also be adapted to incorporate the ARD property [26]. By using the property, the amount of hyperparameters that are used increases from 2 to $n_{\text{x}} + 1$, i.e., one lengthscale hyperparameter l_i^2 for each regressor and the variance hyperparameter σ_{f}^2. The ARD-SE kernel is defined as follows:

$$\Sigma_{\text{SE,ARD}}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sigma_{\text{f}}^2 \exp\left(-\frac{1}{2}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)^T \Lambda^{-1}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right), \tag{2-5}$$

with Λ again a diagonal vector of the lengthscale hyperparameters.

Periodic Kernel

The last kernel to evaluate is the periodic kernel. This kernel is used in problems that involve some kind of periodicity. It also belongs to the stationary kernels category. An example of a periodic kernel is as follows [26]:

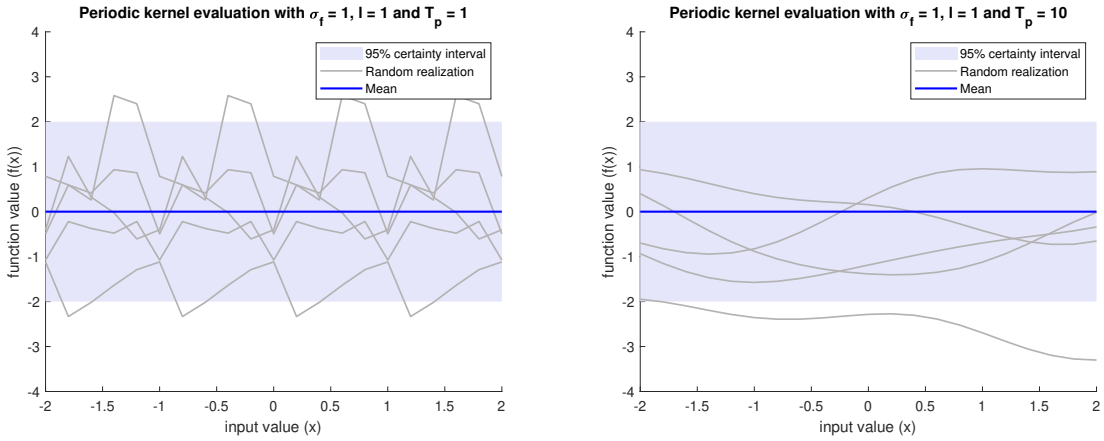


Figure 2-3: Visualization of the prior periodic kernel. The grey lines in the plot are random realizations of functions that are in the function space of the kernel. Also, the 95% certainty interval is visualized which spans the function space which is most likely. The figures indicate that periodicity of the function changes if the periodic hyperparameter is altered.

$$\Sigma_{\text{periodic}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_{\text{f}}^2 \exp\left(-\frac{2}{l^2}\sin^2\left(\frac{\pi}{T_p}r\right)\right), \tag{2-6}$$

with $r = |\mathbf{x}_i - \mathbf{x}_j|$, σ_{f}^2 the variance hyperparameter, l^2 the lengthscale hyperparameter, and T_{p} the periodicity.

The evaluation of the periodic kernel is visualized in Figure 2-3. It is obtained that adjusting the periodicity result in a change in the frequency of the prior functions. The same conclusion is drawn with respect to the kernels discussed before if the lengthscale and variance hyperparameter is altered.

Composite kernels

The kernels that are introduced before are kernels in the general form. To model more complex behaviour it is possible to make composite kernels that are created by doing a kernel operation [26]. A kernel operation is for example an addition of two kernels. The benefit of doing kernel operations is that the composite kernel consists of all the characteristics of the general kernels, e.g., a kernel that consists of both linear and periodic behaviour. Note that the kernel operation is only valid if it produces a positive semi-definite matrix.

A composite kernel that is often used is to add a diagonal matrix of noise variances σ_ϵ^2 to the kernel that is used for modelling the underlying function. This kernel operation adds one free hyperparameter to the GP model which is the noise variance σ_ϵ^2 [33]. The noise matrix is added in order to handle data that is corrupted with noise, or to overcome numerical issues when modelling with GPs. A visualization of an added noise matrix is provided in the following:

$$\Sigma_{\text{composite}} = \Sigma + \sigma_\epsilon^2 I, \tag{2-7}$$

where Σ can be any kernel matrix, and I the identity matrix of appropriate size.

Other examples of kernel operations that are known to be valid are [26]:

- **Sum of kernels:** $\Sigma_{\text{composite}} = \Sigma_1 + \Sigma_2$.
- **Product of kernels:** $\Sigma_{\text{composite}} = \Sigma_1 \times \Sigma_2$.
- **Scaling of the kernel:** $\Sigma_{\text{composite}} = \alpha\Sigma$, with α a scaling term.

2-1-2 Modelling with Gaussian processes

In this thesis, learning a GP model is referred to as fitting a GP function to acquired data [31]. This is done by finding the free variables $\boldsymbol{\theta}$ of the selected kernel function Σ. The free variables are called hyperparameters. Therefore, a dataset $\mathcal{D} \in \mathbb{R}^{N \times n_{\mathrm{x}}+1}$ is first constructed to gather the data of an unknown function f. A dataset consists of a collection of input points $\mathbf{x}_i \in \mathbb{R}^{n_{\mathrm{x}}}$ and output points $y_i \in \mathbb{R}$, i.e. $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}$. The outputs are obtained by providing the input points $\mathbf{x}_i$ to an unknown function $f(\mathbf{x}_i)$ of which outputs y_i are observed. This input/output relation is visualized in the following:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad f : \mathbb{R}^{n_{\mathrm{x}}} \to \mathbb{R}, \epsilon \sim \mathcal{N}\left(0, \sigma_{\epsilon}^2\right), \tag{2-8}$$

where i is the index number indicating the sample, n_{x} the dimension of the input vector, and ϵ_i the measurement noise that is assumed to be Gaussian white noise with zero mean and variance σ_{ϵ}^2. Note that the input points should be chosen carefully in order to capture all necessary information about the unknown function.

Secondly, prior assumptions on the regression problem are introduced. As mentioned before, this is done by specifying a mean and kernel function. In this case, the prior mean function is assumed to be zero due to previously mentioned reasons and the prior kernel function of the output measurements $\mathbf{y}$ is the following [26]:

$$\boldsymbol{\Sigma}_{\mathrm{y}} = \boldsymbol{\Sigma} + \sigma_{\epsilon}^2 \mathbf{I}, \tag{2-9}$$

which is a summed version of the kernel $\boldsymbol{\Sigma}$ of the function f and σ_{ϵ}^2 the variance of the Gaussian white noise. The kernel $\boldsymbol{\Sigma}$ can be any kernel function that is representative for the unknown function f. Combining these two priors, the following prior probability distribution is obtained: $\mathbf{y} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_y)$.

Lastly, the prior probability distribution is used in the Bayesian framework of Equation (2-1) to obtain the posterior distribution of the function f over the dataset $\mathcal{D}$ and the hyperparameters θ. In mathematical terms, the following inference is obtained [26]:

$$p(f \mid \mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \frac{p(\mathbf{y} \mid f, \mathbf{x}, \boldsymbol{\theta}) p(f \mid \boldsymbol{\theta})}{p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})}, \tag{2-10}$$

where $p(f \mid \mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ is the posterior, $(\mathbf{y} \mid f, \mathbf{x}, \boldsymbol{\theta})$ the likelihood, $p(f \mid \boldsymbol{\theta})$ the prior for the initial hyperparameters, and $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ the evidence. However, to obtain the inference of

Equation (2-10) it is required to evaluate complex expressions that might be analytically intractable [26]. Therefore, it is most of the time approximated. A method that is used most often to approximate the inference is to estimate the hyperparameters $\hat{\boldsymbol{\theta}}$ by maximizing the evidence/marginal likelihood of Equation (2-10). The mathematics and further explanation for estimating the hyperparameters is introduced in Section 2-1-3.

2-1-3 Kernel Hyperparameter Estimation

The kernel hyperparameters $\boldsymbol{\theta}$ are essential to extract a model from the data that matches the unknown function [30]. Since it is often unclear how the hyperparameters are valued, they are estimated using Bayesian inference. Bayesian inference is already used previously in Equation (2-10) for explaining the modelling framework. However, for obtaining the hyperparameters a deeper level of inference is required [26]. This level of inference exploits the marginal likelihood (denominator) of Equation (2-10) such that the hyperparameters are inferred over the acquired data. This means that the function f is marginalized out to obtain a likelihood of hyperparameters for the provided data. The deeper layer of inference of the marginal likelihood is:

$$p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y} \mid \mathbf{x})}, \tag{2-11}$$

where $p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$ is recognized as the marginal likelihood of Equation (2-10). However, in most cases, it is analytically intractable to obtain a solution to the inference due to the fact that the posterior distribution is calculated by solving several integrals. An example of an integral that might be analytically intractable is provided by the evidence of Equation (2-11) which looks as follows:

$$p(\mathbf{y} \mid \mathbf{x}) = \int p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{2-12}$$

Luckily, there are several methods that enable numerical approximation of this inference. One widely known technique is to derive an estimation of the hyperparameters by maximizing the marginal likelihood. In Equation (2-11) it is observed that the posterior distribution over the hyperparameters is proportional to the marginal likelihood, i.e., $p(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$. If there is assumed that the hyperparameters are uniformly distributed, it is possible to exploit this proportionality in a maximization problem. The result of the maximization problem is a right selection of the hyperparameters with maximized probability. This is done by first rewriting the evidence like a Gaussian probability distribution as follows:

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{y}^T \boldsymbol{\Sigma}^{-1} \mathbf{y}}. \tag{2-13}$$

Secondly, for numerical scaling purposes, the logarithm of the likelihood is used as an objective function $\ell(\boldsymbol{\theta})$ that is suitable for (non-convex) optimization [26]:

$$\ell(\boldsymbol{\theta}) = \ln p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = -\overbrace{\frac{1}{2}\ln(|\boldsymbol{\Sigma}|)}^{\text{complexity term}} - \overbrace{\frac{1}{2}\mathbf{y}^T \boldsymbol{\Sigma}^{-1} \mathbf{y}}^{\text{data-fit term}} - \overbrace{\frac{N}{2}\ln(2\pi)}^{\text{normalisation const.}}. \tag{2-14}$$

Equation (2-14) is also called the log marginal likelihood which provides an estimation of the hyperparameters θ by maximizing the function.

To maximize the log marginal likelihood, several optimization algorithms are available. If a gradient based optimization tool is used it is recommended to specify the gradient of the log marginal likelihood. The derivative with respect to each parameter is defined as the following [26]:

$$\nabla(\ell(\boldsymbol{\theta})) = \frac{\partial \ln p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})}{\partial \theta_i} = -\frac{1}{2} \operatorname{trace}\left(\boldsymbol{\Sigma}^{-1} \frac{\partial \boldsymbol{\Sigma}}{\partial \theta_i}\right) + \frac{1}{2} \mathbf{y}^T \boldsymbol{\Sigma}^{-1} \frac{\partial \boldsymbol{\Sigma}}{\partial \theta_i} \boldsymbol{\Sigma}^{-1} \mathbf{y}, \tag{2-15}$$

where θ_i is the i-th hyperparameter, and $\partial \boldsymbol{\Sigma} / \partial \theta_i$ the partial derivative of the kernel to the i-th hyperparameter.

2-1-4 Prediction with Gaussian Process models

So far, it is explained how GP models are obtained by fitting a function to data and how the hyperparameters are estimated. However, it is unknown how the model is used to do predictions of the function $f(\mathbf{x}_*)$ at (un)seen input points $\mathbf{x}_*$. Therefore, this section discusses how the acquired GP model is used in a prediction problem.

Next to using Bayesian inference for modelling a GP and the hyperparameter estimation, it is used a third time for doing predictions. The prediction problem uses Bayesian inference for calculating the posterior probability distribution of the function $f(\mathbf{x}_*)$. For obtaining the posterior, it is inferred over the dataset $\mathcal{D}$, the model kernel matrix with its hyperparameters $\boldsymbol{\Sigma}$, and the new (un)seen input $\mathbf{x}_*$ [26]. This is visualized as follows:

$$p\left(f_* \mid \mathcal{D}, \boldsymbol{\Sigma}, \mathbf{x}_*\right) = \frac{p\left(\left[\mathbf{y}^T, f_*\right]^T \mid \boldsymbol{\Sigma}, \mathbf{x}, \mathbf{x}_*\right)}{p\left(\mathbf{y} \mid \boldsymbol{\Sigma}, \mathbf{x}\right)}. \tag{2-16}$$

However, the representation of the predictive posterior in Equation (2-16) is not yet manageable. For making the posterior manageable, it is rewritten to normal (Gaussian) distributions which is valid since Gaussian processes are assumed to be normally distributed (see Definition 1). This is further explained in Appendix A-1 where the following result is obtained:

$$p\left(f_* \mid \mathcal{D}, \boldsymbol{\Sigma}, \mathbf{x}_*\right) = \frac{|\boldsymbol{\Sigma}|^{\frac{1}{2}}}{(2\pi)^{\frac{1}{2}} |\boldsymbol{\Sigma}_{N+1}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\left[\mathbf{y}^T, f_*\right] \boldsymbol{\Sigma}_{N+1}^{-1} \left[\mathbf{y}^T, f_*\right]^T - \mathbf{y}^T \boldsymbol{\Sigma}^{-1} \mathbf{y}\right)}, \tag{2-17}$$

where $\boldsymbol{\Sigma}_{N+1} \in \mathbb{R}^{N+1 \times N+1}$ is defined as the kernel function that incorporates the covariance of the prediction point $\boldsymbol{\Sigma}_{**}$, and the cross-covariance between the prediction point and training points $\boldsymbol{\Sigma}_*$. The matrix is visualized as follows:

$$\boldsymbol{\Sigma}_{N+1} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}_* \\ \boldsymbol{\Sigma}_*^T & \boldsymbol{\Sigma}_{**} \end{bmatrix}. \tag{2-18}$$

The last step for obtaining the posterior GP that is suitable for predictions is to calculate the posterior mean μ_{f_*} and covariance $\sigma^2_{f_*}$ at the prediction point $\boldsymbol{x}_*$. This is achieved by rewriting Equation (2-17) as follows:

$$\begin{aligned} E(f_*) = \mu_{f_*} &= \boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1} \mathbf{y} \\ \text{var}(f_*) = \sigma^2_{f_*} &= \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_*. \end{aligned} \tag{2-19}$$

The derivation of this solution is provided in Appendix A-1. The solution 2-19 fully defines a posterior Gaussian process for doing predictions.

2-2 Gaussian Processes for modelling dynamical systems

The main goal of this thesis is to investigate GPs for the aim of black-box learning of dynamical systems. The engineering practice of learning dynamical systems from experimental data is also known as system identification which is widely used in control applications. System identification hinges on an identification cycle which is an iterative process of data acquisition, modelling and validation, until a proper model has been extracted from the acquired data [25].

In this section, the theory of black-box system identification is combined with the theory of GPR. The section focuses on the black-box system identification of nonlinear systems since the linear system identification theory is a well understood research area, and the goal in this thesis is to construct a dynamical model of a nonlinear system. In black-box system identification it is most often assumed that the dynamical system behaves in a specific model structure [26]. Multiple model structures are known in the literature which are also used in a GP framework. The model structures for GPs are divided into two categories: predictor models and state space (SS) methods. Therefore, the predictor models for GPs are discussed in Section 2-2-1, and the state space methods in Section 2-2-4. Also, in system identification, it is common to rely on noisy measurement data. Therefore, Section 2-2-5 explains GP models that deal with noise in both the modelling and prediction phases.

2-2-1 Nonlinear predictor models for Gaussian processes

A renowned technique in system identification is the prediction error method (PEM). Traditionally, the PEM methods retrieve a model by parameterizing an input-output relation in a specific model structure. The model structures are either linear or nonlinear which depends on the complexity of the system that is to be identified [34].

The model structures of the PEM methods are also frequently used for machine learning applications, and more specifically, especially in a GP machine learning framework [26]. In contrast to the traditional system identification methods, GPs do not parameterize the input-output relation but use the GP regression theory to fit a dynamical function to the observed data. The structure of the dynamical function is defined by a regressor vector $\mathbf{x}(k)$. An example of a dynamical function is the following:

$$y(k) = f(\mathbf{x}(k)) + \epsilon(k), \tag{2-20}$$

Model structure	Mapping
Nonlinear finite impulse response(NFIR)	$\hat{y}(k) = f(u(k-1), u(k-2), \ldots, u(k-m)) + \epsilon$
Nonlinear autoregressive model with exogenous input (NARX)	$\hat{y}(k) = f(y(k-1), y(k-2), \ldots, y(k-n), u(k-1), u(k-2), \ldots, u(k-m)) + \epsilon$
Nonlinear output error (NOE)	$\hat{y}(k) = f(\hat{y}(k-1), \hat{y}(k-2), \ldots, \hat{y}(k-n), u(k-1), u(k-2), \ldots, u(k-m)) + \epsilon$
Nonlinear autoregressive moving average with exogenous inputs (NARMAX)	$\hat{y}(k) = f\left(y(k-1), \cdots, y\left(k-n\right), u(k-1), \cdots, u\left(k-m\right), \epsilon(k-1), \cdots, \epsilon\left(k-l\right)\right) + \epsilon(k)$
Nonlinear Box-Jenkins (NBJ)[1] [35]	$\hat{y}(k) = \frac{B(q)}{A(q)} f(u(k)) + \frac{D(q)}{C(q)} \epsilon(k)$

Table 2-1: Several nonlinear prediction error methods that are used in GP regression problems. The q operator indicates the time lag of the variable. All methods estimate the output function differently which is defined by a regression vector. For example the regression vector $\mathbf{x}(k)$ that is used in the NFIR method is $\mathbf{x}(k) = [u(k-1), u(k-2), \ldots, u(k-m)]^T$. It is also assumed that the estimates are corrupted with Gaussian white noise ϵ.

where k is the discrete timestep indication, y the observed value, $f(\cdot)$ a nonlinear mapping, $\mathbf{x}(k) \in \mathbb{R}^{n_\mathbf{x}}$ the regression vector, and $\epsilon(k)$ Gaussian white noise.

As in common identification theory, the structure of the regression vector $\mathbf{x}(k)$ can take several forms. The choice of the form is based on the properties of the dynamical system that is to be identified, i.e. some systems require more complex structures to model the dynamics. The different PEM model structures are presented in Table 2-1 and discussed in the following.

NFIR structure for Gaussian processes

The nonlinear finite imulse response (NFIR) structure is one of the simpler structures that is used for estimating a nonlinear dynamical function. The regression vector only considers a window of m past inputs [26]:

$$\mathbf{x}_{\text{NFIR}}(k) = \begin{bmatrix} u(k-1) & u(k-2) & \cdots & u(k-m) \end{bmatrix}^T, \tag{2-21}$$

where $u(k)$ is the input of the dynamical system at time k. Compared to the other model structures, the NFIR method does not incorporate feedback of the system [26]. It is therefore required to choose a large window of past inputs for accurately modelling the system dynamics. This makes the method computationally demanding for modelling [10].

NARX structure for Gaussian processes

A structure that is often used for the identification of nonlinear dynamical systems with GPs is the nonlinear autoregressive model with exogenous input (NARX) [36]. The NARX

[1] To the best of the author's knowledge, the method has never been implemented in a GP framework. Therefore it is not discussed in detail.

regression vector considers a window of past inputs and outputs to predict the current output. This regression vector is visualized as:

$$\mathbf{x}_{\text{NARX}}(k) = \begin{bmatrix} y(k-1) & \cdots & y(k-n) & u(k-1) & \cdots & u(k-m) \end{bmatrix}^T, \tag{2-22}$$

where n indicates the considered amount of past outputs $y(k)$ and m the considered amount of past inputs $u(k)$ to the system. Compared to the other methods in Table 2-1 this method suffers from unrealistic noise assumptions which lead to a bias in the model. Therefore, the NARX structure is prone to error accumulation if it is used for simulation where the output of the model is used in a feedback loop [10].

NOE structure for Gaussian processes

The nonlinear output error (NOE) structure estimates the function by making use of a window of past inputs and past estimates. The regression vector of this structure is the following [37]:

$$\mathbf{x}_{\text{NOE}}(k) = \begin{bmatrix} \hat{y}(k-1) & \cdots & \hat{y}(k-n) & u(k-1) & \cdots & u(k-m) \end{bmatrix}^T, \tag{2-23}$$

where $\hat{y}(k)$ is the predicted output of the system. Since the NOE structure uses the predicted output, the model prevents error accumulation in simulation since the simulation error is minimized [10].

NARMAX structure for Gaussian processes

The nonlinear autoregressive moving average with exogenous inputs (NARMAX) model structure is an extended version from the NARX structure [38]. Next to taking into account the measured input and output signals for modelling the dynamical function, the structure also incorporates the noise signal ϵ. This result in the following regression vector:

$$\mathbf{x}_{\text{NARMAX}}(k) = \begin{bmatrix} y(k-1) & \cdots & y(k-n) & u(k-1) & \cdots & u(k-m) & \epsilon(k-1) & \cdots & \epsilon(k-l) \end{bmatrix}^T, \tag{2-24}$$

where l indicates the amount of noise signals. The NARMAX structure experience more freedom for modelling dynamic systems in comparison to the other model structures since it also deals with noise [10].

2-2-2 Model order determination

After selecting the model structure it is also required to choose the order of the model which is the selection of regressors n_x. This choice is important since the selected regressors should not overdetermine the unknown function f, but at the same time the regressor should be independent and carry all information necessary to predict the value of the unknown function. This is also known as the Occam's razor principle which tells that the simplest model

representation is preferred [39]. The following example makes this statement clear. If one takes a simple second order system in discrete time like the following:

$$f(\mathbf{x}_k) = y(k) = a_1 y(k-1) + a_2 y(k-2) + b_1 u(k-1) + b_2 u(k-2), \tag{2-25}$$

where a_i and b_i are scaling coefficients that model the dynamics. It is clear from this function that the regression vector $\boldsymbol{x}(k)$ to model the function f is the following:

$$\begin{aligned} f(\mathbf{x}_k) &= \begin{bmatrix} a_1 & a_2 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} y(k-1) & y(k-2) & u(k-1) & u(k-2) \end{bmatrix}^T \\ &= \gamma \mathbf{x}(k). \end{aligned} \tag{2-26}$$

So, in this case the (ARX) regression vector consists of two past inputs and two past outputs. If one chooses another amount of regressors, the model would be under- or overdetermined which possibly results in under- or overfitting of the model respectively. However, it is not always known in advance what the model order should be. Therefore, a couple of methods exist for retrieving the right model order [26].

The easiest method is to brute-force a discrete set of possible vectors of regressors. The advantage of this method is that it requires no prior knowledge of the model, while the biggest disadvantage is a high computational burden. The second method is an embedded method that selects the regressors in the optimisation procedure like the ARD method. The ARD method introduces a set of hyperparameters for each element in the regression vector, and adjusts automatically the contribution of the element in the hyperparameter optimization. The third method is a filter-based method. In this method, the relevant regressors are usually selected based on the statistical properties of the identification data, e.g., correlations in the data, or by using tools from the information theory, e.g, information entropy. The advantage of this method is that it is relatively computational efficient compared to the other methods [26].

2-2-3 Gaussian process modelling with nonlinear predictor models

Now that it is known how the GP identification problem is designed, it is still unknown how the theory of Section 2-1 is used to construct a dynamical model. First, a dataset is constructed from the regression vectors and output data as follows:

$$\mathcal{D} = (\mathbf{x}_i(k), y_i(k))_{i=1}^N, \tag{2-27}$$

where i indicates the sample in the dataset. This dataset is similar to the one used in Section 2-1-2 but now a time dependency is incorporated. So, this dataset also contains N regression vectors and N output measurements, i.e. $\mathcal{D} \in \mathbb{R}^{N \times n_{\mathrm{x}}+1}$. Secondly, a kernel $\boldsymbol{\Sigma}$ is chosen and calculated. The kernel should be chosen to model the system dynamics. For example, in the case of Equation (2-26), a linear kernel can be chosen since the function to be modelled is assumed to be linear. After initializing the GP problem, the true hyperparameters of the GP model are found by maximizing the marginal log likelihood (MLL) function of Equation (2-14)

to find the true hyperparameters of the model. An example of this procedure for an GP-NARX model is summarized in Algorithm 1. The modelling procedure is similar for the other model structures where the regression vector is composed differently.

Algorithm 1 Algorithm for implementing the GP-NARX model [26].

1: **procedure** OPTIMISENARXMODEL(Inputs)
2: Set input data, target data, covariance function, initial hyperparameters
3: **Repeat**
4: calculate $-\ell(\boldsymbol{\theta})$ and its derivative based on input data $\{y(k-1), y(k-2), \ldots, y(k-n), u(k-1), u(k-2), \ldots, u(k-m)\}$
5: Change hyperparameters
6: **Until** $-\ell(\boldsymbol{\theta})$ is minimal

In the literature, the model structure that is most used in GP applications is the NARX method. The authors of [36] use this technique to predict the hysteresis effects of bolted joint structures. Other examples of applications in engineering are the in the automotive sector for learning a model of a gasoline engine [40] and prediction of wave forces on offshore structures [41]. Also the NFIR and NOE methods for GPs are implemented in applications such as making predictions on the total electron content over South-Africa [42] and the modelling of a car-following model for adaptive cruise control [43] respectively. A NARMAX-like structure for GPs is implemented by [4] in an MPC framework to manage the energy consumption of a building. However, no applications or future prospects are found on the nonlinear Box-Jenkins (NBJ).

2-2-4 State Space methods

A special kind of model structure that covers a lot of other model structures are the SS models [44]. A state space representation is a collection of differential/difference equations that represent the dynamical transition of each of the states. An SS model can either be a model for a linear or nonlinear system. However, for this thesis the nonlinear SS model representation is the most interesting. A discrete-time nonlinear SS model is usually written in the following form [26]:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) + \boldsymbol{\epsilon}_k, \;\; \boldsymbol{\epsilon}_k \overset{iid}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}), \\ \boldsymbol{y}_k &= \boldsymbol{g}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) + \boldsymbol{w}_k, \;\; \boldsymbol{w}_k \overset{iid}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}), \end{aligned} \tag{2-28}$$

where $\boldsymbol{f} \in \mathbb{R}^{\boldsymbol{n}}$ are the functions that capture the dynamical equations of the states $\boldsymbol{x}_k \in \mathbb{R}^n$, $\boldsymbol{u}_k \in \mathbb{R}^{n_\mathrm{u}}$ the external (control) input(s), $\boldsymbol{\epsilon}_k$ the process noise that is independent and identically distributed (i.i.d) and Gaussian white, $\boldsymbol{y}_k \in \mathbb{R}^{n_\mathrm{y}}$ the observed measurements, $\boldsymbol{g} \in \mathbb{R}^{n_\mathrm{y}}$ the measurement functions and $\boldsymbol{\epsilon}_k$ the measurement noise that is also i.i.d and Gaussian white. Note that the time step k is now indicated in a subscript. It is however desired in this section to obtain a Gaussian process state space (GPSS) model. To obtain a GPSS model the representation introduced in Equation (2-28) should be slightly changed.

Constructing a GPSS model amounts to learning a GP model of the state transition functions $\boldsymbol{f}$ and the measurement functions $\boldsymbol{g}$ [45]. A single GP is introduced for every state and

measurement mapping since GPs only map their inputs to a one dimensional space [46]. For the state transition function this comes down to the following representation:

$$\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{cases} f_1(\boldsymbol{x}_k, \boldsymbol{u}_k) & \sim \mathcal{GP}\left(\mu_{f_1}, \sigma_{f_1^2}\right) \\ \vdots \quad \vdots & \vdots \\ f_n(\boldsymbol{x}_k, \boldsymbol{u}_k) & \sim \mathcal{GP}\left(\mu_{f_n}, \sigma_{f_n}^2\right) \end{cases}. \tag{2-29}$$

A similar GP representation can be used for the measurement model where the GP mappings represent each measurement function $g_i(\boldsymbol{x}_k, \boldsymbol{u}_k)$. However, it is possible to simplify the GPSS model by assuming that the measurement model is known [47]. This assumption is valid since the measurement model usually correspond to a sensor model which is in most cases (approximately) known. Because of the fact that the data that is used in this thesis is also sensor data, it can be assumed that the measurement model is already known.

The GP state transition functions $f_i(\boldsymbol{x}_k, \boldsymbol{u}_k)$ of Equation (2-29) are obtained by training a GP model. The training procedure is similar to the previously described GP-NARX problem in Section 2-2-3 if one got access to the full state vector $\boldsymbol{x}_k$ and the input vector $\boldsymbol{u}_k$. The only difference is the composition of the dataset which is due to the fact that another model structure is used. The input data for the GP is now a concatenation of the current states $\boldsymbol{x}_k$ and current external input(s) $\boldsymbol{u}_k$ that serve as the regressors, and the target data of the GP is a concatenation of the measured future state x_{k+1}^i that is drawn from the future state vector $\boldsymbol{x}_{k+1}$ [21]. This procedure should be applied to every state of the system to obtain a full GPSS model.

After having obtained the GPSS model, it can be used in a prediction problem to obtain a one step ahead predictor. This one step ahead predictor is obtained by using the theory on GPR of Section 2-1-4 and looks for every state as follows [46]:

$$\begin{aligned} x_{k+1}^i \sim & \mathcal{N}\left(\mu_{f_i}\left(\boldsymbol{x}_{k+1}\right), \sigma_{f_i}^2\left(\boldsymbol{x}_{k+1}\right)\right), \\ \mu_{f_i}\left(\boldsymbol{x}_{k+1} \mid [\boldsymbol{x}_k, \boldsymbol{u}_k]\right) = & \boldsymbol{\Sigma}_{f_i}([\boldsymbol{x}_k, \boldsymbol{u}_k], X)^\top \left(\boldsymbol{\Sigma}_{fi}(X, X) + \sigma_{\epsilon,i}^2\right)^{-1} Y_i, \\ \sigma_{f_i}^2\left(\boldsymbol{x}_{k+1} \mid [\boldsymbol{x}_k, \boldsymbol{u}_k]\right) = & \boldsymbol{\Sigma}_{fi}\left([\boldsymbol{x}_k, \boldsymbol{u}_k], [\boldsymbol{x}_k, \boldsymbol{u}_k]\right) - \boldsymbol{\Sigma}_{fi}\left([\boldsymbol{x}_k, \boldsymbol{u}_k], X\right)^\top \\ & \left(\boldsymbol{\Sigma}_{fi}(X, X) + \sigma_{\epsilon,i}^2\right)^{-1} \boldsymbol{\Sigma}_{fi}\left([\boldsymbol{x}_k, \boldsymbol{u}_k], X\right), \end{aligned} \tag{2-30}$$

where $\mu_{f_i}\left(\boldsymbol{x}_{k+1}\right)$ is the posterior mean of the i-th state, $\sigma_{f_i}^2\left(\boldsymbol{x}_{k+1}\right)$ the posterior variance of the i-th state, $\boldsymbol{\Sigma}_{f_i}$ the kernel matrix of the i-th state, $[\boldsymbol{x}_k, \boldsymbol{u}_k]$ the regression vector, X the input dataset , Y_i the target data of the i-th state, and $\sigma_{\epsilon,i}^2$ the noise variance hyperparameter of the i-th state. Note that in Equation (2-30) the one-step ahead prediction of a single state from the full state vector is applied. To obtain the one-step ahead prediction of all states, the state prediction procedure for all states needs to be applied. This results in the following GPSS:

$$\begin{aligned} \boldsymbol{x}_{k+1} & \sim \mathcal{N}\left(\boldsymbol{\mu}\left(\boldsymbol{x}_{k+1}\right), \boldsymbol{\Sigma}\left(\boldsymbol{x}_{k+1}\right) I\right), \\ \boldsymbol{\mu}\left(\boldsymbol{x}_{k+1} \mid [\boldsymbol{x}_k, \boldsymbol{u}_k]\right) & = \left[\mu_{f_i}\left(\boldsymbol{x}_{k+1}\right), \ldots, \mu_{f_n}\left(\boldsymbol{x}_{k+1}\right)\right]^\top, \\ \boldsymbol{\Sigma}\left(\boldsymbol{x}_{k+1} \mid [\boldsymbol{x}_k, \boldsymbol{u}_k]\right) & = \left[\sigma_{f_i}^2\left(\boldsymbol{x}_{k+1}\right), \ldots, \sigma_{f_n}^2\left(\boldsymbol{x}_{k+1}\right)\right]^\top. \end{aligned} \tag{2-31}$$

The discussed GPSS problem assumes that full state information is available for the GPSS modelling and prediction. However, full-state information is not always available which means that the regression vector misses crucial information for training a GPSS model. To deal with this problem, other modelling approaches should be chosen. A method for tackling this problem in an offline setting is to perform inference based on particle Markov chain Monte Carlo (PMCMC) which is introduced by Frigola et al. [48] The idea behind this method is to first sample state trajectories from a smoothing distribution after which the samples are used to define a predictive density distribution for learning the dynamics by making use of Monte Carlo integration [45]. The hyperparameters of the GP are obtained by sampling the hyperparameters whilst being conditional on the state trajectory. The sampling of the state trajectories and hyperparameters are performed by a PMCMC sampler. A full explanation and derivation is found in [48] and is out of the scope of this thesis.

The work of [48] is improved several times with respect to the computational burden of the method due to the PMCMC sampler and the length of the time-serie when doing predictions. The authors of [49] tackle this problem by introducing sparsity on the posterior of the dynamical system. The sparse posterior approximation used is the variational free energy (VFE) method which is discussed in Section 2-3. Additionally, the authors of [47] altered the method to enable online learning.

Other methods for modelling GPSS in non-fully observable systems are methods that use the expectiation maximization (EM) algorithm. The EM algorithm is a way to iteratively find the maximum likelihood of the model where the latent states (in this case) are unobserved [50]. The method is not fully Bayesian because of the fact that the EM method includes parameter optimization. In [50] a pseudo training set is used to parameterize the state transition function of (2-30). The method is related to sparse GP approximation methods that is introduced later on.

2-2-5 Stochastic inputs for Gaussian Processes

In the previous sections it is assumed that the regressors are pure and not corrupted by noise. The challenge with stochastic input points is that the use of the earlier mentioned GP theory result in an insufficient model or that the prediction is inaccurate. The problem of noise corrupted input points for Gaussian processes is visualized in the following:

$$y(k) = f(\mathbf{x}(k) + \epsilon_{\mathrm{x}}(k)) + \epsilon_{\mathrm{y}}(k), \tag{2-32}$$

where $\epsilon_{\mathrm{x}}(k) \in \mathbb{R}^{n_{\mathrm{x}}}$ represents the input noise for each regressor with the distribution $\epsilon_{\mathrm{x}} \sim \mathcal{N}(0, \Sigma_x)$, and $\epsilon_{\mathrm{y}}(k)$ the output noise for the measurements with the distribution $\epsilon_{\mathrm{y}} \sim \mathcal{N}\left(0, \sigma_{\mathrm{y}}^2\right)$. The noise for each regressor is assumed to be independent of each other resulting in a diagonal covariance matrix $\mathbf{\Sigma}_{\mathrm{x}}$.

Although it remains a hard problem, there are several solutions available in order to deal with stochastic training or prediction input locations. The solutions to the stochastic input problem will be treated separately for the modelling and prediction stage since the problems stand on their own. Therefore, in this section it is first discussed how to deal with stochastic input points in the modelling stage and secondly this is also discussed for the prediction stage.

Modelling with stochastic input points

The first method that deal with noisy inputs is a very subtle method called the noisy input gaussian process (NIGP) [51]. The method approximates the noisy input function by calculating a first-order Taylor approximation about the regression vector $\mathbf{x}(k)$ and assumes that the Taylor approximation about the noisy measurements $\mathbf{x}(k)+\epsilon_{\mathrm{x}}$ is approximately the same. This assumption is the following:

$$f\left(\mathbf{x}+\epsilon_{\mathrm{x}}\right)=f(\mathbf{x})+\epsilon_{\mathrm{x}}^{T}\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}+\mathcal{O}(n^{2})\simeq f(\mathbf{x}+\epsilon_{\mathrm{x}})+\epsilon_{\mathrm{x}}^{T}\frac{\partial f(\mathbf{x}+\epsilon_{\mathrm{x}})}{\partial(\mathbf{x}+\epsilon_{\mathrm{x}})}+\mathcal{O}(n^{2}). \tag{2-33}$$

However, to derive an exact solution of the Taylor expansion it would be required to calculate a distribution over Taylor expansions. This is due to the fact that the function $f(\cdot)$ is modelled by a GP and the partial derivative in Equation (2-33) is therefore also a GP, i.e. derivative of a GP is also a GP. So, instead of deriving an exact solution to the Taylor expansion, the method in [51] considers a partial derivative of the posterior GP mean function with respect to each (noisy) regressor to calculate the Taylor expansion, i.e. $\partial\mu_{f_*}/\partial\mathbf{x}_*$ instead of $\partial f(\mathbf{x}+\epsilon_{\mathrm{x}})/\partial(\mathbf{x}+\epsilon_{\mathrm{x}})$. It is shown that this introduced method contains similar results compared the exact calculation but it is simpler and more efficient to implement. The result of including the derivative of the posterior mean in the Taylor expansion in Equation (2-32) hold the following:

$$y(k)=f(\mathbf{x}(k))+\epsilon_{\mathrm{x}}(k)^{T}\partial_{\bar{\mathrm{f}}}+\epsilon_{\mathrm{y}}(k), \tag{2-34}$$

where $\partial_{\bar{\mathrm{f}}}$ is the partial derivative of the mean with respect to each regressor $\partial\mu_{f_*}/\partial\mathbf{x}_*$, i.e. the gradient of the posterior mean. So, there is an extra element added that represents the uncertainty in the regressor. Now, the expectation and the uncertainty of the output measurements $y(k)$ becomes:

$$\begin{aligned}
\mathbb{E}\left[y(k)\right] &= \mathbb{E}\left[f(\mathbf{x}(k))+\cancelto{0}{\epsilon_{\mathrm{x}}(k)^{T}\partial_{\bar{\mathrm{f}}}}+\cancelto{0}{\epsilon_{\mathrm{y}}(k)}\right]=f(\mathbf{x}(k)),\\
\mathrm{Var}(y(k)) &= \mathbb{E}\left[y(k)^{2}\right]-\mathbb{E}\left[y(k)\right]^{2}\\
&= \mathbb{E}\Big[f(\mathbf{x}(k))^{2}+\cancelto{0}{2f(\mathbf{x}(k))\epsilon_{\mathrm{x}}(k)^{T}\partial_{\bar{\mathrm{f}}}}+\cancelto{0}{2f(\mathbf{x}(k))\epsilon_{\mathrm{y}}(k)}+\cancelto{0}{2\epsilon_{\mathrm{x}}(k)^{T}\partial_{\bar{\mathrm{f}}}\epsilon_{\mathrm{y}}(k)}+\cdots\\
&\qquad \partial_{\bar{\mathrm{f}}}^{T}\epsilon_{\mathrm{x}}(k)\epsilon_{\mathrm{x}}(k)^{T}\partial_{\bar{\mathrm{f}}}+\epsilon_{\mathrm{y}}(k)^{T}\epsilon_{\mathrm{y}}(k)\Big]-f(\mathbf{x}(k))^{2}\\
&= \partial_{\bar{\mathrm{f}}}^{T}\Sigma_{\mathrm{x}}\partial_{\bar{\mathrm{f}}}+\sigma_{\mathrm{y}}^{2}.
\end{aligned} \tag{2-35}$$

If the input uncertainty is incorporated in the Gaussian Process, the prior becomes:

$$\mathbf{y}\sim\mathcal{N}(0,\boldsymbol{\Sigma}+\partial_{\bar{\mathrm{f}}}^{T}\Sigma_{\mathrm{x}}\partial_{\bar{\mathrm{f}}}+\sigma_{\mathrm{y}}^{2}\mathbf{I}). \tag{2-36}$$

Therefore the predictive posterior distribution of a deterministic test point $\mathbf{x}_*$ is calculated as:

$$
\begin{aligned}
E\left(f_*\right) = \mu_{\mathrm{f}_*} &= \boldsymbol{\Sigma}_*^T \left(\boldsymbol{\Sigma} + \sigma_{\mathrm{y}}^2 \mathbf{I} + \operatorname{diag}\left(\Delta_{\bar{\mathrm{f}}} \Sigma_{\mathrm{x}} \Delta_{\bar{\mathrm{f}}}^T\right)\right)^{-1} \mathbf{y}, \\
\operatorname{var}\left(f_*\right) = \sigma_{f_*}^2 &= \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^T \left(\boldsymbol{\Sigma} + \sigma_{\mathrm{y}}^2 \boldsymbol{I} + \operatorname{diag}\left(\Delta_{\bar{\mathrm{f}}} \Sigma_{\mathrm{x}} \Delta_{\bar{\mathrm{f}}}^T\right)\right)^{-1} \boldsymbol{\Sigma}_*,
\end{aligned}
\tag{2-37}
$$

where diag() is the diagonal matrix operator, $\mathbf{I}$ the identity matrix of appropriate size and $\Delta_{\mathrm{f}} \in \mathbb{R}^{N \times n_{\mathrm{x}}}$ a matrix containing all the partial derivatives of the posterior mean $\partial_{\bar{f}}$. However, it is still unclear how to find the partial derivatives $\partial_{\bar{f}}$ when modelling the NIGP. Therefore, in [51] a two-step approach is used. First, the posterior mean is calculated using the normal GP modelling framework of Section 2-1 while ignoring the input noise, and secondly the gradient of the posterior mean $\partial_{\bar{f}}$ is calculated analytically at every training point. This two step approach is used since it is analytically impossible to use Equation (2-37) directly due to the dependency on the same posterior mean, i.e., finding the derivative would require to find the derivative of $\Delta_{\bar{\mathrm{f}}}$ as well.

If one is able to derive the matrix with the gradients of the posterior mean it is used to update the model by introducing the prior of Equation (2-36). So, this involves introducing n_{x} extra hyperparameters that belong to the input noise variances, i.e., the hyperparameters are the diagonal elements in the input noise matrix Σ_{x} that is multiplied by the calculated posterior mean gradient. The extra hyperparameters are trained alongside the other hyperparameters of the model which makes the method easy to implement.

A second method that can deal with noisy training data is to use a Gaussian process latent variable model (GP-LVM). This model treats the inputs $\boldsymbol{x}(k)$ as unobserved variables which are also called the latent variables [52]. The method relies on the problem of variational inference where the latent variables are approximately integrated out. Such methods are known to exist in the approximated Gaussian Processes that are introduced in Section 2-3. The methods that are meant here are the approximated posterior methods such as the VFE method or the power expectation propagation (PEP) method. Since these approximate GP methods are introduced later on, it is not further discussed here.

Prediction with stochastic input points

Sometimes it is the case that the input vector for prediction $\boldsymbol{x}_*$ is uncertain, e.g., $\boldsymbol{x}_* \sim \mathcal{N}\left(\mu_{\mathrm{x}_*}, \boldsymbol{\Sigma}_{\mathrm{x}_*}\right)$ if the uncertain input is Gaussian distributed [53]. The uncertainty is due to the fact that the prediction input vector is corrupted with noise or if the prediction is made based on an uncertainty model such as an GP model. The second reason is usually the case if the uncertainty model is used in a multiple-step ahead prediction of time series. Due to this uncertainty in the prediction input, the predictive distribution is obtained by integrating over the uncertain input $\boldsymbol{x}_*$. In the case of a Gaussian distributed input variable, this is visualized as follows:

$$
p\left(f\left(\mathbf{x}_*\right) \mid \mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right) = \int p\left(f\left(\mathbf{x}_*\right) \mid \mathbf{x}_*, \mathcal{D}\right) p\left(\mathbf{x}_*\right) d\mathbf{x}_*. \tag{2-38}
$$

However, the exact solution to this integral is analytically intractable since $p\left(f\left(\mathbf{x}_*\right) \mid \mathbf{x}_*, \mathcal{D}\right)$ is a complicated function of $\boldsymbol{x}_*$ [53]. Therefore, there are several methods that approximate the predictive posterior distribution. These methods are also referred to as uncertainty propagation.

The first method that is used for the prediction at uncertain inputs is the naive method or equivalently the zero variance method. This method is called naive because it assumes that the inputs in the prediction problem are deterministic [54]. In a multi-step ahead prediction this means that it adds the predictive mean to the observation set at every iterative step while ignoring the uncertainty of the predictive mean. In this way, the prediction is overconfident since it does not account for the uncertainty that is induced in every step of the prediction [53]. The method only incorporates the uncertainty of the GP prediction problem itself, i.e. how similar the prediction input is to the training data [26].

The second method is based on an analytical (Gaussian) approximation of the predictive distribution by only computing the expected mean and variance of the predictive distribution of Equation (2-38), i.e. $p\left(f\left(\mathbf{x}_*\right) \mid \mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right) \approx \mathcal{N}\left(\mu\left(\mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right), \Sigma\left(\mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right)\right)$. This method is also called moment matching. The expected mean and variance are the following [55]:

$$\begin{aligned}
\mu\left(\mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right) =& \mathbb{E}_{\boldsymbol{x}_*}\left[\boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right)\right]\left(\boldsymbol{\Sigma}+\sigma_\epsilon \boldsymbol{I}\right)^{-1} \boldsymbol{y}, \\
\Sigma\left(\mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right) =& \mathbb{E}_{\boldsymbol{x}_*}[\boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{x}_*)] - \left(\left(\boldsymbol{\Sigma}+\sigma_\epsilon \boldsymbol{I}\right)^{-1} - \boldsymbol{y}^T\left(\boldsymbol{\Sigma}+\sigma_\epsilon \boldsymbol{I}\right)^{-1}\left(\boldsymbol{\Sigma}+\sigma_\epsilon \boldsymbol{I}\right)^{-1} \boldsymbol{y}\right) \times \\
& \mathbb{E}_{\boldsymbol{x}_*}\left[\boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right) \boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right)^T\right] - \mu\left(\mu_{\mathrm{x}_*}, \Sigma_{\mathrm{x}_*}\right)^2,
\end{aligned} \tag{2-39}$$

where the expected kernel matrices of the prediction input $\boldsymbol{x}_*$ are defined as follows:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}_*}[\boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{x}_*)] &= \int \boldsymbol{\Sigma}(\boldsymbol{x}_*, \boldsymbol{x}_*) p(\boldsymbol{x}_*) d\boldsymbol{x}_*, \\
\mathbb{E}_{\boldsymbol{x}_*}\left[\boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right)\right] &= \int \boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right) p(\boldsymbol{x}_*) d\boldsymbol{x}_*, \\
\mathbb{E}_{\boldsymbol{x}_*}\left[\boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right) \boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right)^T\right] &= \int \boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right) \boldsymbol{\Sigma}\left(\boldsymbol{x}_*, \boldsymbol{x}\right)^T p(\boldsymbol{x}_*) d\boldsymbol{x}_*.
\end{aligned} \tag{2-40}$$

Now it is dependent on the chosen prior kernel matrix if it is possible to calculate the integrals of Equation (2-40). The method exact moment matching is used if it is possible to calculate the exact integrals. This is for example the case if a linear kernel or Gaussian kernel is used in the GP prediction problem. However, if no analytical solution exists to the integrals of Equation (2-40) one should apply the method approximate moment matching. Approximate moment matching evaluates a second-order Taylor expansion of the kernel function around the mean $\mu_{\boldsymbol{x}_*}$ of the uncertain input $\boldsymbol{x}_*$. A derivation for exact and approximate moment matching is given in [53].

2-3 Approximation Techniques for Gaussian Processes

An issue arises when the GPR problem has to deal with very big datasets that detoriorate the modelling of the (exact) GP due to its computational burden [12]. The computational burden is mostly due to the calculation of the inverse of the model matrix $\boldsymbol{\Sigma}$ as is observed in Equation (2-19). The calculations involved for exact GP regression are in $\mathcal{O}(n^3)$ with regard to the number of observations n. However, excessive calculations are prevented when introducing techniques for making the GPR more efficient. In [16] an overview is provided of

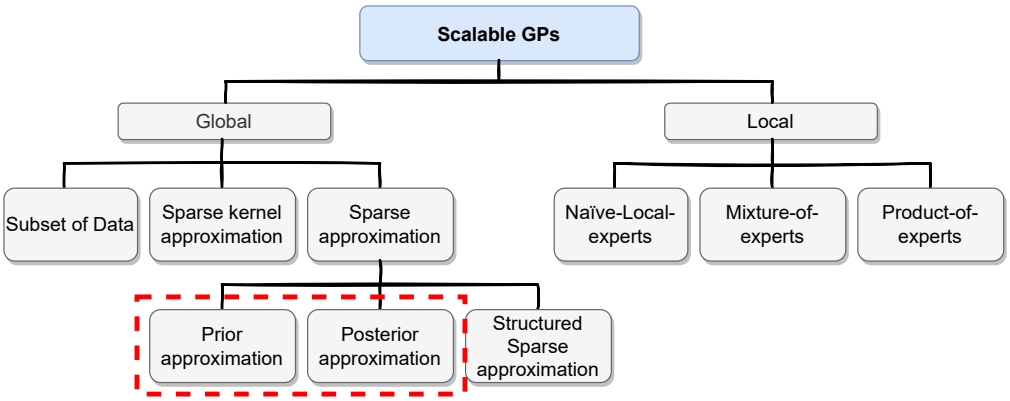


Figure 2-4: Overview of several strategies that improve the efficiency of GPR [16]. The approximation techniques are divided in global and local approximation. In recent years, a lot of contributions are done in this area as is retrieved by looking at the amount of methods. As this is an overview, it is possible that more methods are available. The red box indicates that the methods of interest for this thesis are the prior and posterior approximation methods.

strategies for improving the efficiency of GPs which is presented in Figure 2-4. The theory of approximate GPs is extensive and not all relevant for this thesis. The approximation methods that are interesting for this thesis are indicated by a red box in the figure. However, the other methods are still shortly introduced.

As observed from the figure, a GP can either be approximated at a global or local level in which both a variety of approximation methods exist. The local methods divide the exact GP into 'local experts' which are modelling the function in a specific region. These methods are well explained in [16]. Contrarily, the global methods approximate the entire GP model by introducing a sparse kernel matrix that is easier to invert [16]. The global methods that are not used in this thesis are now shortly introduced.

At the global level, the simplest approximation method is the subset of data (SoD) where a selection of the training data is made for training the approximate GP [56]. Also, a method for approximating the GP at a global level is to sparsify the kernel (sparse kernel approximation) by assuming that training points are uncorrelated if they are some distance apart, $\Sigma(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$ if $|\boldsymbol{x}_i - \boldsymbol{x}_j|$ exceeds a threshold [57]. Another method that is used for sparsification is the sparse spectrum Gaussian process (SSGP) which is only used for stationary kernels. Here, the sparsification is obtained by limiting the size of the Fourier transform of a stationary kernel [58].

The remaining part of this section discusses the sparsification methods that are interesting for this thesis. In Section 2-3-1 the prior approximation methods are discussed and Section 2-3-2 explains the posterior approximation methods.

2-3-1 Prior sparse approximation methods

Many ideas on prior approximation exist which all come down to one central unifying idea. Therefore, this unifying idea is first explained in this section. After having clear how the

GP prior is approximated, the methods for prior approximations and their assumptions are explained.

The central element for the prior approximations is to introduce inducing variables $\mathbf{u} \in \mathcal{R}^m$. The inducing variables $\mathbf{u}$ are found in the function space of the Gaussian process and correspond to an input location called the inducing input $\mathbf{x_u}$. These inputs are either found by applying a certain selection heuristic that chooses the inputs from the training set, or they are optimized over when maximizing the MLL [56], [59]. Other methods for finding the inducing input points are random selection from the training set or clustering techniques [16].

The inducing variables are used for interconnecting the prediction inputs $\mathbf{f}_*$ and training inputs $\mathbf{f}$. The training and prediction inputs are just observations in the latent function space, i.e. $\mathbf{f} = [f_1, f_2, \ldots, f_n]^\top$, where $f_i = f(\mathbf{x}_i)$ (observe that this is without noise). Now that some notations are introduced, it is also required for later usages to relate the inducing inputs to the training and test inputs. This is done by making the training and test inputs conditionally dependent on the inducing variable. Therefore, the following conditionals are obtained [56]:

$$\begin{aligned} &\text{training conditional:} \quad p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}\left(K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\mathbf{ff}} - K_{\mathbf{fu}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}}\right), \\ &\text{test conditional:} \quad p(\mathbf{f}_* \mid \mathbf{u}) = \mathcal{N}\left(K_{\mathbf{f}_*\mathbf{u}}K_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\mathbf{f}_*\mathbf{f}_*} - K_{\mathbf{f}_*\mathbf{u}}K_{\mathbf{uu}}^{-1}K_{\mathbf{uf}_*}\right), \end{aligned} \tag{2-41}$$

where the term $\boldsymbol{K}_{..}$ refers to the kernel matrix $\boldsymbol{\Sigma}(\cdot,\cdot)$ where the subscript $\boldsymbol{f}$ is used for the training input $\boldsymbol{x}$, $\boldsymbol{f}_*$ for the prediction input $\boldsymbol{x}_*$, and $\boldsymbol{u}$ for the inducing input $\boldsymbol{x}_u$. Note that the conditionals in Equation (2-41) are exact and therefore no sparsification has yet been performed. The sparsification is obtained by approximating the exact joint prior probability distribution $p(\mathbf{f}_*, \mathbf{f})$ of the GP. This is also the reason why this method is named prior approximation. For this purpose, it is required to first define the exact prior probability distribution. This distribution is obtained by marginalizing out the inducing variables as follows [56]:

$$p(\mathbf{f}_*, \mathbf{f}) = \int p(\mathbf{f}_*, \mathbf{f}, \mathbf{u})\, \mathrm{d}\mathbf{u} = \int p(\mathbf{f}_*, \mathbf{f} \mid \mathbf{u})\, p(\mathbf{u})\mathrm{du}, \quad \text{with } p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{uu}}). \tag{2-42}$$

This joint prior probability distribution is the short hand notation for the term in the numerator of Equation (2-16). The sparsification is then obtained by introducing an approximation of the exact prior probability distribution of Equation (2-42). This approximation is obtained by assuming that both the test inputs $\mathbf{f}_*$ and training inputs $\mathbf{f}$ are conditionally independent given $\mathbf{u}$ [56]. Therefore, the following approximate relation is observed:

$$p(\mathbf{f}_*, \mathbf{f}) \simeq q(\mathbf{f}_*, \mathbf{f}) = \int q(\mathbf{f}_* \mid \mathbf{u})\, q(\mathbf{f} \mid \mathbf{u}) p(\mathbf{u}) \mathrm{d}\mathbf{u}. \tag{2-43}$$

Note that an approximation of the training and test conditional is present in the integral of Equation (2-43)

	$q(\mathbf{f} \mid \mathbf{u})$[1]	$q(\mathbf{f}_* \mid \mathbf{u})$[2]	Joint Prior Distr.
SoR/DIC	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{fu}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{0}\right)$	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{f}_*\mathbf{u}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{0}\right)$	$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{Q}_{\mathbf{ff}} & \boldsymbol{Q}_{\mathbf{ff}_*} \\ \boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}} & \boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}_*} \end{bmatrix}\right)$
DTC/PLV/PPA	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{fu}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{0}\right)$	$p(\mathbf{f}_* \mid \mathbf{u})$	$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{Q}_{\mathbf{ff}} & \boldsymbol{Q}_{\mathbf{ff}_*} \\ \boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}} & \boldsymbol{K}_{\mathbf{f}_*\mathbf{f}_*} \end{bmatrix}\right)$
FITC/SGPP	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{fu}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\text{FITC}}\right)$[3]	$p(\mathbf{f}_* \mid \mathbf{u})$	$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{Q}_{\mathbf{ff}} - K_{\text{FITC}} & \boldsymbol{Q}_{\mathbf{ff}_*} \\ \boldsymbol{Q}_{*\mathbf{f}} & \boldsymbol{K}_{\mathbf{f}_*\mathbf{f}_*} \end{bmatrix}\right)$[3]
FIC	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{fu}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\text{FITC}}\right)$[3]	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{f}_*\mathbf{u}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\text{FIC}}\right)$[3]	$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{Q}_{\text{ff}} - K_{\text{FITC}} & \boldsymbol{Q}_{\mathbf{ff}_*} \\ \boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}} & \boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}_*} - K_{\text{FIC}} \end{bmatrix}\right)$[3]
PITC	$\mathcal{N}\left(\boldsymbol{K}_{\mathbf{fu}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\mathbf{u}, K_{\text{PITC}}\right)$[3]	$p(\mathbf{f}_* \mid \mathbf{u})$	$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{Q}_{\text{ff}} - K_{\text{PITC}} & \boldsymbol{Q}_{\mathbf{ff}_*} \\ \boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}} & \boldsymbol{K}_{\mathbf{f}_*\mathbf{f}_*} \end{bmatrix}\right)$[3]

Table 2-2: Several methods for sparse prior approximation [56].

$$
\begin{aligned}
K_{\text{FITC}} &= \text{diag}\left[\boldsymbol{K}_{\mathbf{ff}} - \boldsymbol{Q}_{\mathbf{ff}}\right] \\
K_{\text{FIC}} &= \text{diag}\left[\boldsymbol{Q}_{\mathbf{f}_*\mathbf{f}_*} - \boldsymbol{K}_{\mathbf{f}_*\mathbf{f}_*}\right] \\
K_{\text{PITC}} &= \text{blockdiag}\left[\boldsymbol{K}_{\mathbf{ff}} - \boldsymbol{Q}_{\mathbf{ff}}\right]
\end{aligned}
\tag{2-44}
$$

which is indicated by $q(\cdot)$. The integral in Equation (2-43) is designed such that the test input $\mathbf{f}_*$ and training inputs $\mathbf{f}$ in the approximation can only communicate through the inducing variable $\mathbf{u}$. They are therefore independent which is the foundation of this theory [56].

As mentioned before, the (unifying) approximation that is introduced in Equation (2-43) is a generalization for a variety of prior sparsification methods. The methods differ in the way how the approximate training and test conditional of Equation 2-43 are defined. An overview of these methods is listed in Table 2-2. Due to lack of space, the prior joint covariance entries are simplified using $Q_{\mathbf{a},\mathbf{b}} \triangleq K_{\mathbf{a},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}K_{\mathbf{u},\mathbf{b}}$. The prior joint distributions can be used to calculate the posterior distribution by using the Bayesian inference of Equation (2-19).

As retrieved from Table 2-2, the several approaches differ with respect to the variance of the approximate training conditional and the (approximate) test conditional. The explanation of these sparsification methods is extensively explained by the authors of [56] and elaborated on in the following. All stressed sparse likelihood approximations have a computational complexity of $\mathcal{O}(nm^2)$. For the coming explanations there is referred to Table 2-2.

The first method is the subset of regressors (SoR) [60], [61] (adapted by [62]) or equally the deterministic inducing conditional (DIC) [56]. This method approximates the exact GPR by using a deterministic relation between $\mathbf{f}$ and $\mathbf{u}$, and $\mathbf{f}_*$ and $\mathbf{u}$ by assuming zero variance. The downside of this method is that the prior distribution is degenerate because of the fact that this sparse model limits the number of independent functions that can be drawn from this prior. This fact leads to an overconfidence in the predictive covariance of the drawn realizations, even with enough training points.

Another prior approximation method that can actually deal with the predictive uncertainties whilst also having the same predictive mean as the SoR method, is the deterministic training

[1] Approximate training conditional
[2] Approximate test conditional
[3] See Equation (2-44)

conditional (DTC) [63] (also called projected latent variables (PLV) [64] or projected process approximation (PPA) [30]). The method only approximates the (deterministic) training conditional and, differently from SoR, takes the exact test conditional. This results in a more conservative predictive covariance. However, there should be noted that DTC is very prone to overfitting if the induced variables are not learned properly [65].

The last three methods are also quite similar. The fully independent training conditional (FITC) [56] (or sparse Gaussian processes using pseudo-inputs (SGPP) [66]), the fully independent conditional (FIC) [56] and the partially independent training conditional (PITC) [56] approximations are non-deterministic. The methods differ in the way how the variance of the approximate training conditional is defined and/or how the (approximate) test conditional is defined. As pointed out in [65], these methods are able to reassure exact GP if the pseudoinputs are chosen optimally. However, this is quite challenging since the model and inference are no longer seperated which introduces confusion in interpretation and modification. An explanation for this observation is provided in [65].

2-3-2 Posterior sparse approximation methods

The last sparse approximation method that is discussed are the posterior approximation methods [16]. These methods approximate the posterior distribution by performing approximate inference. The exact GP prior is retained by using this method. This Section introduces two sparse posterior approximations, beginning with the VFE method and ending with the PEP method. The introduced methods have a computational complexity of $\mathcal{O}(nm^2)$.

The most well known method in the posterior approximations is the VFE method. The VFE method relies on a bound F_q which is also called the evidence lower bound (ELBO) or VFE. The ELBO is written as the difference between the log-marginal likelihood of the exact posterior $p(\boldsymbol{y})$ distribution and the Kullback–Leibler (KL) divergence of a variational distribution $q(\cdot)$ and the exact posterior $p(\cdot)$. This is visualized as follows [67]:

$$F_q = \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\left[\log \frac{p\left(\boldsymbol{y},\boldsymbol{f},\boldsymbol{u}\right)}{q\left(\boldsymbol{f},\boldsymbol{u} \mid \boldsymbol{y}\right)}\right] = \log p(\boldsymbol{y}) - \mathrm{KL}\left(q\left(\boldsymbol{f},\boldsymbol{u} \mid \boldsymbol{y}\right) \| p\left(\boldsymbol{f},\boldsymbol{u} \mid \boldsymbol{y}\right)\right). \tag{2-45}$$

The KL divergence is a measure for similarity between probability distributions [68]. In this case the similarity of the variational distribution $q(\mathbf{f},\mathbf{u} \mid \mathbf{y})$ and the true posterior $p(\mathbf{f},\mathbf{u} \mid \mathbf{y})$ is measured.

The KL divergence in Equation (2-45) is a positive-semidefinite expression. This is due to the fact that the divergence saturates if the variational distribution equals the exact posterior, i.e. $q(\mathbf{f},\mathbf{u} \mid \mathbf{y}) = p(\mathbf{f},\mathbf{u} \mid \mathbf{y})$. By observing this fact and by observing that the exact posterior is constant for the variational distribition it can be concluded that minimizing the KL divergence is equivalent to maximizing the VFE. The benefit of maximizing the VFE bound is that this allows us to jointly find the inducing inputs and hyperparameters while also being able to approximate the evidence and the posterior. This maximized VFE bound is also calculated analytically in [67] by finding the optimal choice of the variational distribution. The details of the calculation are out of the scope of this thesis. However, the result of this calculation is interesting. The maximized VFE bound is an objective function that can be used in an optimization problem to find the hyperparameters of the sparse GP and optionally

the inducing inputs. The maximized VFE bound, which is in fact a marginal log likelihood function, is the following [27]:

$$\mathcal{L}_{\text{VFE}} = \frac{N}{2}\log(2\pi) + \underbrace{\frac{1}{2}\log\left|\boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}\right|}_{\text{complexity penalty}} + \underbrace{\frac{1}{2}\mathbf{y}^\top\left(\boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}\right)^{-1}\mathbf{y}}_{\text{data fit}} + \underbrace{\frac{1}{2\sigma_\epsilon^2}\operatorname{tr}(\boldsymbol{K}_{\mathbf{ff}} - \boldsymbol{Q}_{\mathbf{ff}})}_{\text{trace term}}, \quad (2\text{-}46)$$

Notice that this marginal likelihood is very similar to the one that is obtained for the DTC method [65]. The only difference is the added trace term. This term acts as a regularizer that prevents overfitting. As explained in the previous section, overfitting is the main issue for obtaining a sparse model with the DTC method. Therefore, it is obtained that the VFE method has an increased performance compared to the DTC method.

The MLL of Equation (2-46) can be used to obtain a sparse GP model of the VFE method. However, it is still unknown how this model is used for predictions at unseen input locations. So, an expression for the sparse posterior should be introduced. This sparse posterior is also a general result of the derivation in [67]. From this derivation it is obtained that the general form of the approximate GP posterior of the VFE method is the following:

$$\begin{aligned}
\mu_{\mathrm{f}_*}^{\text{VFE}} &= \frac{1}{\sigma_\epsilon^2}\boldsymbol{K}_{\mathrm{f}_*\mathbf{u}}\left(\boldsymbol{K}_{\mathbf{uu}} + \frac{1}{\sigma_\epsilon^2}\boldsymbol{K}_{\mathbf{uf}}\boldsymbol{K}_{\mathbf{fu}}\right)^{-1}\boldsymbol{K}_{\mathbf{uf}}\boldsymbol{y}, \\
\sigma_{\mathrm{f}_*}^{\text{VFE}} &= \boldsymbol{K}_{\mathrm{f}_*\mathrm{f}_*} - \boldsymbol{K}_{\mathrm{f}_*\mathbf{u}}\boldsymbol{K}_{\mathbf{uu}}^{-1}\boldsymbol{K}_{\mathbf{u}\mathrm{f}_*} + \boldsymbol{K}_{\mathrm{f}_*\mathbf{u}}\left(\boldsymbol{K}_{\mathbf{uu}} + \frac{1}{\sigma_\epsilon^2}\boldsymbol{K}_{\mathbf{uf}}\boldsymbol{K}_{\mathbf{fu}}\right)^{-1}\boldsymbol{K}_{\mathbf{u}\mathrm{f}_*}.
\end{aligned} \quad (2\text{-}47)$$

The second posterior approximation to be discussed in this thesis is the PEP method. This method is introduced by [65] and is based on the expectation propagation (EP) algorithm of [69]. The PEP is a generalization of the EP algorithm that unifies the FITC and DTC prior approximations with the VFE method. It creates therefore a hybrid framework.

The algorithm of the PEP method is thoroughly described in [65]. The details of this algorithm is out of the scope of this thesis. However, the found results in [65] are more interesting. They obtain a general closed loop result at convergence of the algorithm for Gaussian regression cases. This result is an analytically tractable approximate log marginal likelihood function. It is obtained that this MLL function is a hybrid connection of the FITC and VFE method. The (hybrid) connection is visualized as follows:

$$\begin{aligned}
\mathcal{L}_{\text{PEP}} = &-\frac{n}{2}\log 2\pi - \frac{1}{2}\log\mid \alpha K_{\text{FITC}} + \boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}\mid \\
&-\frac{1}{2}\boldsymbol{y}^{\mathbf{T}}\left(\alpha K_{\text{FITC}} + \boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}\right)^{-1}\boldsymbol{y} - \frac{1-\alpha}{2\alpha}\operatorname{tr}\left[\log\left(\boldsymbol{I} + \frac{\alpha}{\sigma_\epsilon^2}\left(\boldsymbol{K}_{\mathbf{ff}} - \boldsymbol{Q}_{\mathbf{ff}}\right)\right)\right],
\end{aligned} \quad (2\text{-}48)$$

where $\alpha \in (0, 1]$ is a scaling constant. It is recognized that the objective function in Equation (2-48) is the same as the MLL function of the VFE method if α approaches zero. However, if α approaches 1 it is observed that the FITC method is recovered. All α values in between are hybrid connections of the two methods. It is claimed that the PEP method outperforms other sparsification methods due to the ability to combine sparse methods [65].

The PEP method does also have a general result for the approximate posterior. This approximate posterior is very similar to the approximate posterior of the VFE method of Equation (2-47). The only difference is that there are extra variables introduced in order to obtain the hybrid connection. Therefore, the following approximate posterior for the PEP method is obtained [65]:

$$\begin{aligned}\mu_{\mathrm{f}_*}^{\mathrm{PEP}} &= \frac{1}{\alpha K_{\mathrm{FITC}} + \sigma_\epsilon^2 \boldsymbol{I}} \boldsymbol{K}_{\mathrm{f}_*\mathbf{u}} \left(\boldsymbol{K}_{\mathbf{uu}} + \frac{1}{\alpha K_{\mathrm{FITC}} + \sigma_\epsilon^2 \boldsymbol{I}} \boldsymbol{K}_{\mathbf{uf}} \boldsymbol{K}_{\mathbf{fu}} \right)^{-1} \boldsymbol{K}_{\mathbf{uf}} \boldsymbol{y}, \\ \sigma_{\mathrm{f}_*}^{\mathrm{PEP}} &= \boldsymbol{K}_{\mathrm{f}_*\mathrm{f}_*} - \boldsymbol{K}_{\mathrm{f}_*\mathbf{u}} \boldsymbol{K}_{\mathbf{uu}}^{-1} \boldsymbol{K}_{\mathbf{u}\mathrm{f}_*} + \boldsymbol{K}_{\mathrm{f}_*\mathbf{u}} \left(\boldsymbol{K}_{\mathbf{uu}} + \frac{1}{\alpha K_{\mathrm{FITC}} + \sigma_\epsilon^2 \boldsymbol{I}} \boldsymbol{K}_{\mathbf{uf}} \boldsymbol{K}_{\mathbf{fu}} \right)^{-1} \boldsymbol{K}_{\mathbf{u}\mathrm{f}_*}.\end{aligned} \tag{2-49}$$

2-4 Model predictive control with Gaussian process dynamical models

The dynamical behaviour that is captured by GP learning is very useful in control applications. In most control techniques an accurate model of the system is needed to assure stability, safety and sufficient performance of the controller. The benefit of using GPs in combination with control is the ability of GPs to model (nonlinear) dynamical systems with little prior knowledge while also providing a measure of uncertainty [14]. This section aims to focus on how the Model Predictive Control (MPC) framework is used in combination with GPs.

MPC is a finite horizon optimal control method that calculates the optimal control input based on prediction [70]. The prediction with horizon N_{p} is based on the model of the (discrete-time) system. At every iteration of the control system, the first control input is chosen from a sequence of optimized control inputs. The method is increasingly popular in control applications because of the fact that the optimization problem allows constraints. However, modelling dynamical systems with GPs does not always result in deterministic state trajectories and therefore requires a different approach than traditional MPC problems.

The model for calculating the prediction of the state trajectories in a stochastic fashion is the identified GP model which consist of a mean and a variance [71]. The mean and variance are used to design the cost function and constraints of the MPC problem. However, due to the predictive nature of MPC, the GP is simulated N_p steps ahead which results in a propagation of the uncertainty of the GP. Several methods for uncertainty propagation exist as described in Section 2-2-5.

When combining MPC with GP, the problem still consist of an objective function and constraints. There are almost no restrictions on the design of the objective function in the GP-MPC problem [26]. The only restriction is that positive definite weighting matrices should be used in order to preserve the convexity in the objective function. A common used objective function for tracking a reference with stochastic MPC is to take the expectation of a common quadratic MPC objective function as follows [14]:

$$\begin{aligned} l\left(y_i - y_{i,\mathrm{ref}}, u_i - u_{i,\mathrm{ref}}\right) &= \|y_i - y_{i,\mathrm{ref}}\|_Q^2 + \|u_i - u_{i,\mathrm{ref}}\|_R^2, \\ \mathbb{E}\left(l\left(y_i - y_{i,\mathrm{ref}}, u_i - u_{i,\mathrm{ref}}\right)\right) &= \|\mu_i^y - y_{i,\mathrm{ref}}\|_Q^2 + \mathrm{tr}\left(Q\Sigma_i^y\right) + \|\mu_i^u - u_{i,\mathrm{ref}}\|_R^2 + \mathrm{tr}\left(R\Sigma_i^u\right), \end{aligned} \tag{2-50}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are positive definite weighting matrices, i the step indication, $y_{i,\text{ref}}$ the reference at step time i, u the control input, y the output of the system, $\boldsymbol{\Sigma}^*$ the covariance matrices of the specified parameter *, and $\text{tr}(\cdot)$ the trace operator. The control input u_i is often assumed deterministic, i.e. $\mathbb{E}[u_i^2] = u_i^2$ [24]. This simplifies the expectation of the cost function by dropping the trace term of the control input.

The constraints of a stochastic MPC problem allow chance constraints. The chance constraints are used for improving the robustness of the MPC controller by incorporating the uncertainty of the model. This comes from the fact that the controller is depending on the predictions of the GP model which are uncertain. The uncertainty possibly leads to an undesirable violation of the constraints by the actual system [72]. The chance constraints are the following [14]:

$$\begin{aligned} \Pr(y(k) \in \mathcal{Y}) &\geq p_\text{y}, \\ \Pr(u(k) \in \mathcal{U}) &\geq p_\text{u}, \end{aligned} \tag{2-51}$$

where p_* indicates the probability that the control input and model output remain in a set $\mathcal{U}$ and $\mathcal{Y}$ respectively. The constraints are calculated by approximating the chance constraints. One way for implementing this is to introduce a two-sided linear constraint [73]. An example of this constraint for the output trajectory is:

$$\Pr\left(y_\text{min} - \mu_i^y \leq (\Sigma_i^y)^{\frac{1}{2}}\, \xi \leq y_\text{max} - \mu_i^y\right) \geq p_\text{y} \;\forall\; i \in [k, k+N_p-1]\,, \tag{2-52}$$

where y_min and y_max are the minimum and maximum values of the output trajectory, and ξ a normally distributed random variable with zero mean and unit variance. This two-sided chance constraint is approximated by the following constraints [73]:

$$\begin{aligned} y_\text{min} - \mu_i^y &\leq \Phi^{-1}(\epsilon^\star)\,(\Sigma_i^y)^{\frac{1}{2}} \\ \mu_i^y - y_\text{max} &\leq \Phi^{-1}(\epsilon^\star)\,(\Sigma_i^y)^{\frac{1}{2}} \\ y_\text{min} - y_\text{max} &\leq 2\Phi^{-1}(\epsilon^\star/2)\,(\Sigma_i^y)^{\frac{1}{2}}\,, \end{aligned} \tag{2-53}$$

where $\epsilon^\star = (1-p_\text{y})/1.25$ and $\Phi^{-1}(\cdot)$ the inverse of the cumulative Gaussian distribution function. Other methods for approximating the chance constraints are discussed in [74] and [14].

The GP-MPC problem is best understood by providing an example. In [4], an output-feedback MPC is implemented of which the multiple-step ahead prediction of the GP model is based on a zero-variance principle. The zero-variance principle does not account for the uncertainty propagated in earlier predictions of the multiple-step ahead prediction. In the example, the uncertainty of the prediction is incorporated in the cost function for punishing the uncertain model trajectories of the GP. Therefore, the following cost function is used:

$$J(\mathbf{x}, y_\text{ref}, \mathbf{u}, \sigma_\text{y}) = \sum_{\tau=0}^{N_\text{p}-1} \left(\bar{y}_{t+\tau} - y_\text{ref}\right)^2 + u_{t+\tau}^T R u_{t+\tau} + \lambda \sigma_{t+\tau}^2, \tag{2-54}$$

where $\mathbf{x}$ is the regression vector, y_ref the reference trajectory, $\bar{y}$ the mean trajectory of the GP for predicting multiple timesteps ahead in time, u the control inputs, σ_y^2 the variance at a

specific input, λ a positive weighting scalar that punishes uncertainty and R a positive-definite weighting-term.

The constraints of the minimization problem that define the model behaviour, actuator constraints and chance constraints are as follows:

$$\begin{aligned} \text{minimize } & J(\mathbf{x}, \mathbf{y}_{ref}, \mathbf{u}, \sigma_{\mathbf{y}}) \\ \text{subject to } & \bar{y}_{t+\tau} = \mu\left(x_{t+\tau}\right) + \boldsymbol{\Sigma}_{\mathbf{y}\mathrm{f}_*}^T \boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}(Y - \mu(X)) \\ & \sigma_{t+\tau}^2 = \boldsymbol{\Sigma}_{\mathrm{f}_*\mathrm{f}_*} - \boldsymbol{\Sigma}_{\mathbf{y}\mathrm{f}_*}^T \boldsymbol{\Sigma}_{\mathbf{yy}}^{-1} \boldsymbol{\Sigma}_{\mathbf{y}\mathrm{f}_*} \\ & u_{t+\tau} \in \mathcal{U} \\ & \Pr\left(y_{t+\tau} \in \mathcal{Y}\right) \geq 1 - \epsilon. \end{aligned} \tag{2-55}$$

The constraints define the prediction of the GP model for N_p steps ahead. Note that the mean function $\bar{y}_t$ is calculated differently than is introduced earlier in this thesis. This is due to a non-zero prior on the mean [26]. The controller is also subject to actuator constraints $\mathcal{U}$ and GP output constraint $\Pr\left(y_{t+\tau} \in \mathcal{Y}\right)$ that keeps the output of the system within some set with probability $1 - \epsilon$.

In the literature, there are several examples found of stochastic MPC problems that use black-box GP models. In [4] and [73] the authors apply the stochastic MPC framework with a NARMAX-like structure for managing the energy consumption of a building. The GP-NARX structure is implemented by [22] where the GP model is used in combination with an output feedback MPC to control the concentration of a certain reactant in a continuous stirred-tank reactor. Another approach is used in [75] and [21], where a GPSS system is implemented that is used for an NMPC controller to model the concentration of reactants. However, the GPSS model assumes full state information which simplifies the problem significantly.

Chapter 3

Methodology

This chapter outlines the methodology for the performed research. The study is conducted to a double pendulum system in both simulation and a physical setup. The simulation environment is meant to build a deeper understanding of the considered system and to allow for the rapid development of a subsequent Gaussian process (GP) identification and control algorithm. When the algorithm yields promising results in simulation, it will be evaluated on a real-world double pendulum setup. The actual setup has been made available by the research laboratory of the Delft Center for Systems and Control (DCSC) at the Delft University of Technology (TU Delft).

To obtain insights of the implementation choices being made, this chapter is split into three sections. Section 3-1 describes the first-principles modelling of the double pendulum system that is used for simulation, and, the specifications of the system for both the simulation and physical setup. This section is meant to define the systems that are used for data acquisition and control. As the goal of this research is to develop a fully data-driven GP-MPC algorithm, the systems are treated as black-boxes.

Subsequently, Section 3-2 defines the research that is performed with the specified algorithm. This involves among others a specification of the kernels functions, the data acquisition, the model structures and the validation metrics.

Lastly, Section 3-3 explains what algorithm is implemented for identifying and controlling the black box systems. This section incorporates the theory on Gaussian processes (GPs) and MPC that is acquired previously. It also tackles the practical and numerical challenges of implementing the algorithm.

3-1 Double Pendulum System

This section discusses first-principles model of the double pendulum system that is derived for creating a simulation environment for developing the GP-MPC algorithm. Also, the physical system that is available in laboratory is further specified for clarification. Therefore,

Section 3-1-1 explains the derivation and implementation of the first-principles model, and Section 3-1-2 specifies the employed physical setup. For the simulation it is needed to obtain a dynamic model of the system that is suitable for numerical evaluation.

3-1-1 First-principles model derivation

This section derives a nonlinear first-principles model of a double pendulum system. The derivation is supplemented with model parameters, positive force directions and assumptions.

Figure 3-1 presents a visualization of the considered double pendulum system. The model in the figure consists of a motor, two beams with length l_1 and l_2, and an end effector point mass m_3. The motor exerts a torque T on the system, which results in a rotation of the beams with angles θ_1 and θ_2 respectively. The positive rotational direction is anti-clockwise. In the model, it is assumed that the links have a mass m_1 and m_2 respectively. The mass also induces an inertia of the beams that is indicated with I_1 for beam 1 and I_2 for beam 2. The inertia caused by the point mass is assumed to be negligible. Also, the links experience linear viscous friction at their pivot points. This is indicated by the friction coefficients b_1 and b_2 respectively. The symbol g is used to indicate the gravitational constant.

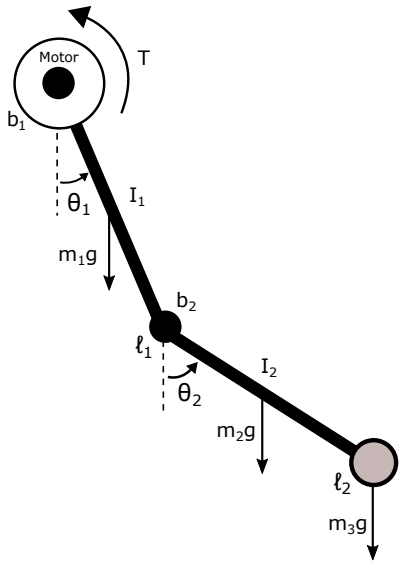


Figure 3-1: Visualization of the double pendulum system that is used for simulation. The model consists of a motor that exerts a torque T on the system, two beams with length l_1 and l_2 and an end effector point mass m_3. If the motor exerts a positive (anti-clockwise) torque on the system, the beams rotate with angles θ_1 and θ_2 respectively. It is also assumed that the links have masses m_1 and m_2, and that the pivot points of the beams experience linear viscous friction with friction coefficients b_1 and b_2. The inertia of the links is specified by I_1 and I_2 respectively.

A dynamical mathematical model of the system of Figure 3-1 is constructed by calculating the equations of motion (EoMs). To this end, Lagrangian mechanics are used. The resulting equations of this calculation are the following:

$$\underbrace{\begin{bmatrix} J_1 & J_3\cos(\theta_1-\theta_2) \\ J_3\cos(\theta_1-\theta_2) & J_2 \end{bmatrix}}_{M(\theta_1,\theta_2)} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = -\underbrace{\begin{bmatrix} J_3\dot{\theta}_2^2\sin(\theta_1-\theta_2) \\ -J_3\dot{\theta}_1^2\sin(\theta_1-\theta_2) \end{bmatrix}}_{V(\theta_1,\theta_2,\dot{\theta}_1,\dot{\theta}_2)} - \underbrace{\begin{bmatrix} \mu_1\sin(\theta_1) \\ \mu_2\sin(\theta_2) \end{bmatrix}}_{G(\theta_1,\theta_2)} - \underbrace{\begin{bmatrix} (b_1+b_2)\dot{\theta}_1 \\ -b_2\dot{\theta}_2 \end{bmatrix}}_{D(\dot{\theta}_1,\dot{\theta}_2)} + \underbrace{\begin{bmatrix} T \\ 0 \end{bmatrix}}_{F}, \quad (3\text{-}1)$$

with:

$$\begin{aligned} J_1 &= \frac{1}{3}m_1l_1^2 + m_2l_1^2 + m_3l_1^2, \\ J_2 &= \frac{1}{3}m_2l_2^2 + m_3l_2^2, \\ J_3 &= \frac{1}{2}m_2l_1l_2 + m_3l_1l_2, \\ \mu_1 &= gl_1(\frac{1}{2}m_1 + m_2 + m_3), \\ \mu_2 &= gl_2(\frac{1}{2}m_2 + m_3). \end{aligned} \quad (3\text{-}2)$$

The full derivation of the EoMs for this double pendulum model is provided in Appendix B-1. The states for the nonlinear state space (NSS) representation of Equation (3-1) are $\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2$ which are the angular displacement and velocity for both beams respectively. Also, the vectors and matrices that are involved in the equation represent the inertia matrix $M(\theta_1, \theta_2)$, the Coriolis vector $V(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$, the gravity vector $G(\theta_1, \theta_2)$, the dissipative energy vector $D(\dot{\theta}_1, \dot{\theta}_2)$ and the external force vector F. Note that the $(\dot{\cdot})$-notation indicates the derivative with respect to time, i.e., $\mathrm{d}/\mathrm{d}t$.

It is however that the controller should be implemented digitally for the physical setup which means that the continuous-time state space representation of Equation (3-1) is not realistic for simulation. Equation (3-1) is therefore converted to discrete-time by using the fourth-order method of Runge-Kutta (RK-4) [76].

3-1-2 Hardware setup

The second part of this thesis consists of testing a GP-MPC algorithm on a real-world double pendulum setup. As mentioned previously, the double pendulum setup has been made accessible by the DCSC in their research laboratory. A picture of the setup that is available in the laboratory is visualized in Figure 3-2.

Figure 3-2 visualizes the real setup and a schematic representation of the real setup. The setup consists of two links that are free to move in the vertical plane. The upper link in the figure is driven by a DC motor which is the only actuator of the system. So, the second link is not motorized. Both of the arms can rotate 360 degrees.

Sensors in the setup enable measurements of angle θ_1 of the first link and the angular rotation of the second link θ_2. Also, the input signal to the DC motor of the system is provided in Volts. So, the setup is a single input multiple output (SIMO) system due to its single input and two outputs structure. To send the input signals and to read off the output signals, the realtime

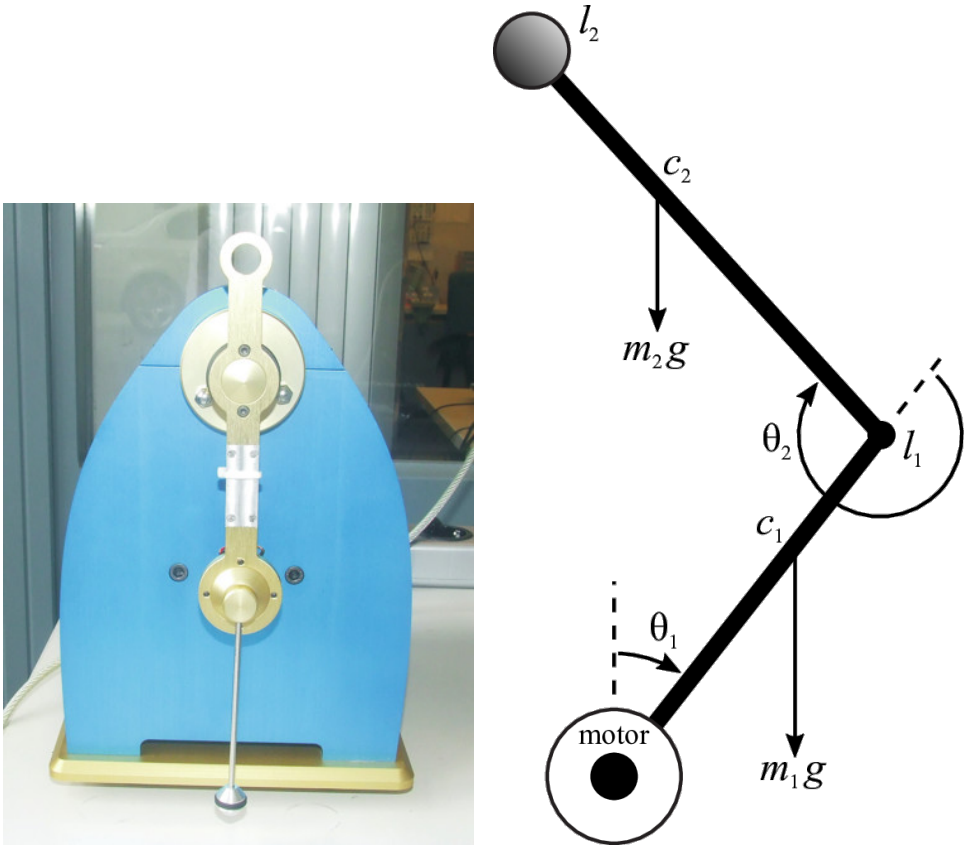


Figure 3-2: Visualization of the double pendulum system that is available in the DCSC laboratory [77]. The figure on the left is a photograph of the real system. The figure on the right is a schematic representation of the physical system. The schematic representation consists of a motor and two links. The links have an angular displacement of θ_1 and θ_2 respectively. Also the length, masses and friction coefficient of the two links are indicated by l_1, l_2, m_1, m_2, c_1 and c_2 respectively.

system is connected to a computer. The software that is used for this purpose is Matlab Simulink. Note that the measured angular rotation of the sensor of θ_2 is directly connected to the the angular rotation of θ_1. In other words, the sensor of θ_2 measures negative values of θ_1 if θ_1 rotates positively. To overcome this dependency, the measurements of angle θ_2 are augmented in Simulink by adding both angles together, i.e., $\theta_2^{\text{aug}} = \theta_1 + \theta_2$. The augmentation of angle θ_2 simplifies the system for control purposes, e.g., θ_2^{aug} handles constant references.

3-2 Research setup

This section clarifies which setup is used for the performed research. Section 3-2-1 discusses the acquisition of identification data for both the simulation and physical setup. Furthermore, Section 3-2-2 specifies the kernel functions and Section 3-2-2 explains the model structures. Lastly, Section 3-2-4 discusses the validation metrics for model verification.

3-2-1 Data acquisition

The data is acquired by providing the dynamical systems of Section 3-1 an input signal. After providing the input signal, both the input and output signals are collected in a dataset. In obtaining a sufficient dataset, the input signal should excite as many frequencies as possible which is a common practice in system identification.

In the linear system identification theory, the pseudo random binary sequence (PRBS) input signal is most often used to excite the system. This signal fluctuates between two values while exciting a lot of frequencies of the system. However, as stressed in [78], the output data of the system misses crucial information if the PRBS signal is used for the identification of nonlinear systems. This is due to the fact that the PRBS signal only provides information at two input values. Therefore, in this thesis it is chosen to work with another input signal.

The signal that is used in this thesis is the amplitude modulated pseudo random binary sequence (APRBS) [78]. This signal is modified from the PRBS signal to be suitable for nonlinear system identification. The modification is obtained by introducing different amplitudes for the pseudo random signal. This ensures that the data consists of more information by covering a great part of the input space. However, it should be noted that the length of the signal should be chosen sufficient for ensuring a proper coverage of the input space. Also, the hold time should be chosen properly. The hold time is the minimum length of time at which the signal remains constant. This is necessary for the system to settle and ensures the system of not getting stuck in a specific operating region.

3-2-2 Kernels

A great variety of kernels of kernel functions are existing in the literature. Therefore, a selection of kernel functions is made to investigate in this thesis. The selection is based on the characteristics of kernel functions that might suit the behaviour of a double pendulum system. The kernels that are explored in this thesis are the squared exponential (SE) kernel, the periodic kernel and a multiplication of these two kernels. In the remaining part of this section, the implementation and the reason of choice of the kernels are discussed.

Squared Exponential kernel

The Squared Exponential kernel (SE) is a kernel that is the most used kernel in GP applications [26]. It is known to have attractive modelling properties and it is able to model complex nonlinear functions. The SE kernel is implemented by also making use of the automatic relevance determination (ARD) property for embedding regressor importance. The following representation of the SE kernel is used for a computationally fast implementation:

$$\Sigma_{\text{SE,ARD}}\left(\boldsymbol{X}_1, \boldsymbol{X}_2\right) = \sigma_{\text{f}}^2 e^{(\circ \boldsymbol{D}_{\text{ARD}})}, \tag{3-3}$$

where $\circ$ indicate the Hadamard exponential with the following matrix D_{ARD}:

$$\boldsymbol{D}_{\text{ARD}} = -\frac{1}{2}\left(\left[\boldsymbol{X}_{1,\Lambda} \overset{\times N_{\boldsymbol{X}_2}}{\cdots\cdots}\right] - 2\boldsymbol{X}_1\Lambda^{-1}\boldsymbol{X}_2^T + \begin{bmatrix}\boldsymbol{X}_{2,\Lambda}^T \\ \vdots \\ \times N_{\boldsymbol{X}_1}\end{bmatrix}\right), \tag{3-4}$$

$$\begin{aligned} \boldsymbol{X}_{1,\Lambda} &= (\boldsymbol{X}_1 \odot \boldsymbol{X}_1) \begin{bmatrix} l_1^{-2} & l_2^{-2} & \cdots & l_{n_x}^{-2} \end{bmatrix}^T \\ \boldsymbol{X}_{2,\Lambda} &= (\boldsymbol{X}_2 \odot \boldsymbol{X}_2) \begin{bmatrix} l_1^{-2} & l_2^{-2} & \cdots & l_{n_x}^{-2} \end{bmatrix}^T, \end{aligned} \tag{3-5}$$

where $\odot$ indicate an element-wise multiplication, $\left[\boldsymbol{X}_{\cdot,\Lambda} \overset{\times N_{\boldsymbol{X}_\cdot}}{\cdots\cdots}\right]$ an N-time vector repetition and $\boldsymbol{X}_1 \in \mathbb{R}^{N_{\boldsymbol{X}_1} \times n_x}$, and $\boldsymbol{X}_2 \in \mathbb{R}^{N_{\boldsymbol{X}_2} \times n_x}$ indicate the matrices that contain the data, i.e. the stacked regression vectors. The reason of this kernel implementation is the computational efficiency due to the fact that only matrix operations are used.

Periodic kernel

The periodic kernel is a kernel that is known to be able to capture periodic behaviour of systems [26]. This kernel is interesting for this thesis since a double pendulum also exhibit some periodic behaviour. Therefore, this kernel is also implemented in this thesis for research. The periodic kernel function is the following [79]:

$$\Sigma_{\text{periodic}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\text{f}}^2 \exp\left(-\frac{1}{2}\sum_{z=1}^{n_{\text{x}}}\left(\frac{1}{l_z^2}\sin^2\left(\frac{\pi}{T_{\text{p},z}}|\mathbf{x}_{i,z} - \mathbf{x}_{j,z}|\right)\right)\right), \tag{3-6}$$

where the z in $\mathbf{x}_{\cdot,z}$ indicates the z'th element of the regressor. The representation of the kernel (3-6) is also rewritten in a form where it only consists of matrix operations. This is achieved by introducing element wise operations which is also obtained for the SE kernel.

Composite kernel

The last kernel that is considered in this thesis is a composite kernel of the SE kernel and periodic kernel. The kernel operation performed to obtain this kernel is multiplication. The resulting kernel is locally periodic. This composite kernel might result in an improved model for the double pendulum compared to the previous introduced kernels since it is able to model complex periodic behaviour. The kernel representation is the following:

$$\Sigma_{\text{C}}(\boldsymbol{X}_1, \boldsymbol{X}_2) = \Sigma_{\text{SE,ARD}}(\boldsymbol{X}_1, \boldsymbol{X}_2)\,\Sigma_{\text{Periodic}}(\boldsymbol{X}_1, \boldsymbol{X}_2) \tag{3-7}$$

3-2-3 Model structure

The model structures that are considered in this thesis are the nonlinear autoregressive model with exogenous input (NARX), nonlinear output error (NOE), and the augmented nonlinear state space (NSS) methods which are selected on the base of their attractive implementation properties and their promising modelling power for nonlinear system identification. Unlike the other methods discussed in the theory, the considered methods are not dependent on the error dynamics or unobserved states which makes them easier to implement with GPs. The

remainder of this section explains what regression vectors are used for GP modelling and prediction.

The GP-NARX regression vector contains the past inputs and outputs of the system that are structured in an auto-regressive manner. For doing accurate predictions with the model, it is expected that the regression vector requires the past outputs of both angles θ_1 and θ_2 for accurately modelling the dynamics. This expectation comes from the fact that the beams are interconnected which means that their dynamics are entangled. The NARX regression vector is assumed to contain the same structure for modelling both θ_1 and θ_2 and is the following:

$$\begin{aligned}\boldsymbol{x}_{\text{NARX}}(k) = \big[&\theta_1(k) \quad \theta_1(k-1) \quad \cdots \quad \theta_1(k-n+1) \quad \theta_2(k) \quad \theta_2(k-1) \quad \cdots \\ &\cdots \quad \theta_2(k-n+1) \quad u(k) \quad u(k-1) \quad \cdots \quad u(k-m+1)\big]^T,\end{aligned} \tag{3-8}$$

where the amount of regressors n and m are found by analyzing the linearized first principles model since it is assumed that the resulting NARX model is a concatenation of local linear auto-regressive models.

The GP-NOE regresion vector is defined similar to the GP-NARX method, only the measured outputs are changed to prediction outputs of the model. Therefore, the following regression vector is used:

$$\begin{aligned}\boldsymbol{x}_{\text{NOE}}(k) = \big[&\hat{\theta}_1(k) \quad \hat{\theta}_1(k-1) \quad \cdots \quad \hat{\theta}_1(k-n+1) \quad \hat{\theta}_2(k) \quad \hat{\theta}_2(k-1) \quad \cdots \\ &\cdots \quad \hat{\theta}_2(k-n+1) \quad u(k) \quad u(k-1) \quad \cdots \quad u(k-m+1)\big]^T,\end{aligned} \tag{3-9}$$

where the amount of regressors n and m are found similarly to the GP-NARX method. Note that training the GP-NOE models requires an iterative scheme of training and prediction due to the availability of the predicted outputs in the regression vector.

Lastly, the augmented GP-NSS model structure is a general representation for the other model structures. The states are found in the double pendulum system of Equation (3-1) which include the angular displacement for both beams θ_1 and θ_2, and the corresponding angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$. However, these states are not fully observable due to missing sensors in the physical setup (and for that reason it is assumed that it is hidden in the simulation as well). Therefore, the angular velocity values are approximated by introducing a numerical derivative which result in an augmented NSS. The used numerical derivative is the backward difference formula. By using this approximation, the resulting autoregressive regression vector for the augmented GP-NSS is as follows:

$$\boldsymbol{x}_{\text{GPSS}}(k) = \Big[\theta_1(k) \quad \theta_1(k) - \theta_1(k-1) \quad \theta_2(k) \quad \theta_2(k) - \theta_2(k-1) \quad u(k)\Big]^T. \tag{3-10}$$

3-2-4 Validation

The model that is found by optimizing the marginal log likelihood (MLL) should be tested on validity. For this purpose, a validation dataset is constructed that is used for analysis.

Note that the data in the validation set have not been observed when training the model. The validation dataset is structured as introduced in Section 3-2-3 and used in a prediction problem. After predicting the values with the model, it is compared to the real data by using a performance measure. The first performance measure used in this thesis is the variance accounted for (VAF) which is defined as follows [80]:

$$\text{VAF} = \max\left(0, \left(1 - \frac{\frac{1}{N}\sum_{k=1}^{N} \|y_{\text{real}}(k) - \mathbb{E}\left(\hat{y}(k)\right)\|_2^2}{\frac{1}{N}\sum_{k=1}^{N} \|y_{\text{real}}(k)\|_2^2}\right) \cdot 100\%\right), \tag{3-11}$$

where $\|\cdot\|_2$ is the 2-norm, y_i^{real} is the real observed value, $\mathbb{E}\left(\hat{y}_i\right)$ the predicted value and N the number of samples. This performance measure is used for calculating the similarity percentage of the model compared to the real observed value. The second used performance measure is the mean standardised log loss (MSLL) which is defined as follows [33]:

$$\text{MSLL} = \frac{1}{2N}\sum_{i=1}^{N}\left[\ln\left(\sigma_i^2\right) + \frac{\left(y_i^{\text{real}} - \mathbb{E}(\hat{y}_i)\right)^2}{\sigma_i^2}\right] - \frac{1}{2N}\sum_{i=1}^{N}\left[\ln\left(\sigma_y^2\right) + \frac{\left(y_i^{\text{real}} - \mathbb{E}(\mathbf{y})\right)^2}{\sigma_y^2}\right], \tag{3-12}$$

where $\sigma_i^2 = \sigma_{\text{f},i}^2 + \sigma_\epsilon^2$ is the sum of the variance of the predicted output $\sigma_{\text{f},i}^2$ with the noise variance σ_ϵ^2 at the i-th prediction sample, σ_y^2 the variance of the measured output data and $\mathbb{E}(\mathbf{y})$ the mean of the measured data. The MSLL performance measure differs from the VAF performance measure in the sense that it also considers the uncertainty of the prediction. The model is good if the MSLL outputs a value of zero, but the model is better if the MSLL outputs a negative value [26]. The formula retrieves that erroneous but uncertain predictions are diminished in value, while certain and erronous (overconfident) predictions increase the MSLL value. So, the performance measure punishes overconfident models which adds an extra dimension to the performance measure for probabilistic models compared to the VAF.

3-3 The algorithm

The goal in this thesis is to make a dynamical model of double pendulum system using GPs in both simulation and a real-world setup which are used for controlling the simulation/real-world system. For this purpose, a GP-model predictive control (MPC) algorithm is proposed that is outlined in this section. Figure 3-3 visualizes the algorithm and shows that the algorithm is carried out in an offline and online phase.

The offline phase starts by initializing an exact GP training that optimizes the hyperparameters over the (exact) negative marginal log likelihood (NMLL) function. However, the exact GP is expected not to be feasible for incorporation in the realtime GP-MPC algorithm due to time consuming calculations. Therefore, a sparse GP model training is initialized that requires a set of inducing points. These inducing points are found by expoiting the exact GP model and are retrieved by selecting the regression vectors that contain the most information for the exact GP. The exact mechanism for this heuristic is explained later on. After obtaining the most important regression vectors, the same is cycle is repeated for training a sparse GP model which is eventually incorporated in the online phase. So, the offline phase contains an iterative process of data acquisition, GP identification and validation. Note that

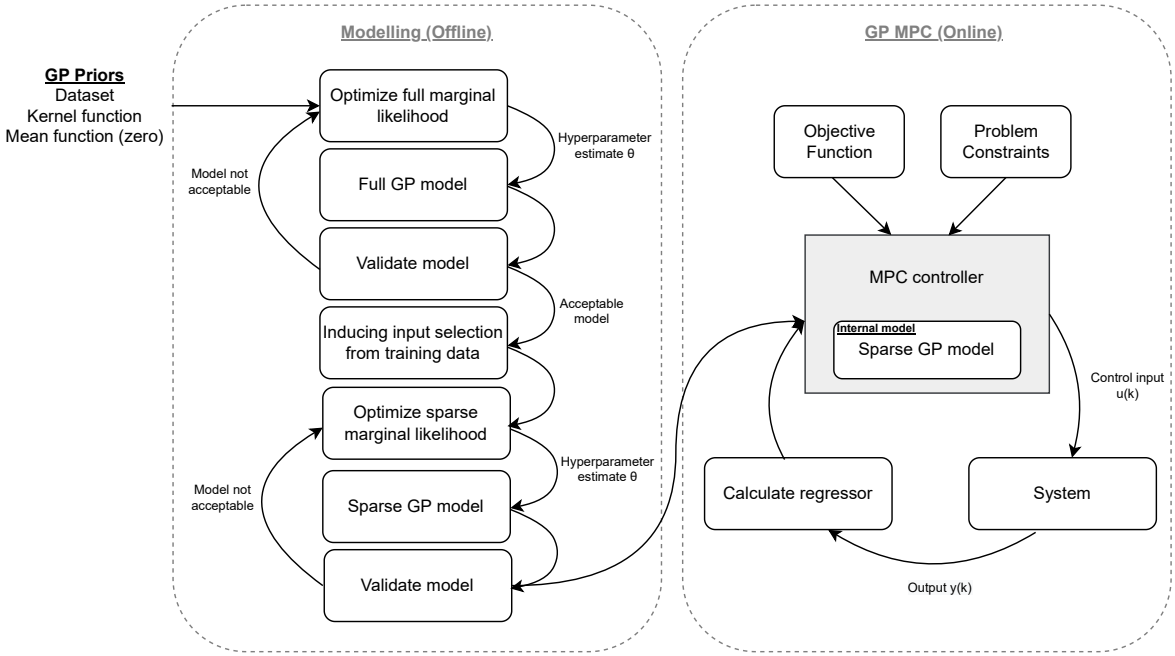


Figure 3-3: Visualization of the algorithm that is used in this thesis. The algorithm consists of an offline and online phase. In the offline phase, a sparse GP model is constructed by using inducing inputs that are found after training a exact GP model. The inducing inputs are the training inputs that contain the most information for the full GP. After validating the sparse model the model is ready for incorporation in the online phase. In the online phase an MPC controller is employed that uses the sparse GP model as internal model. The controller calculates the control inputs for controlling the system.

the offline phase is carried out two times since the online phase performs multivariable control that requires a model for both angles θ_1 and θ_2.

The online phase consists of an (online) MPC control framework that exploits the sparse GP models that are obtained in the offline phase. The sparse GP models are unchanged in the online phase and used as (offline) internal models for predicting the system over the prediction horizon of the MPC. The MPC controller is initialized with a multivariable objective function that is constrained. Also, the MPC controller directly communicates with the double pendulum system by refreshing the regression vector with the measured angles θ_1 and θ_2 at each iteration of the controller. The controller is therefore single-shooting.

The remainder of this section discusses the implementation of the algorithm and its design choices. The algorithm is fully implemented in Matlab R2022a [81]. Section 3-3-1 discusses a numerical stable implementation for learning the hyperparameters of a full GP model, and the use of the exact GP model in a prediction problem. Section 3-3-2 obtains the same is for the sparse GP model. Lastly, Section 3-3-3 explains the control algorithm of the online phase.

3-3-1 Implementation of the full Gaussian Process

The full GP is implemented following the theory on GPs of Chapter 2-1. The optimal hyperparameters of the GP model are found by employing a NMLL optimization of which the resulting GP model is then used in a prediction problem. The NMLL function and GP prediction solutions are restated:

$$-\ell(\boldsymbol{\theta}) = -\ln p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = \overbrace{\frac{1}{2}\ln\left(|\boldsymbol{\Sigma}|\right)}^{\text{complexity term}} + \overbrace{\frac{1}{2}\mathbf{y}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{y}}^{\text{data-fit term}} + \overbrace{\frac{N}{2}\ln(2\pi)}^{\text{normalisation const.}}, \tag{3-13}$$

$$\begin{aligned} E\left(\mathrm{f}_*\right) &= \mu_{\mathrm{f}_*} = \boldsymbol{\Sigma}_*^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{y}, \\ \mathrm{var}\left(\mathrm{f}_*\right) &= \sigma^2_{\mathrm{f}_*} = \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_*, \end{aligned} \tag{3-14}$$

where f_* indicate the predicted function value, μ_{f_*} the mean of the predicted function value, $\sigma^2_{\mathrm{f}_*}$ the variance of the predicted output value $\boldsymbol{\Sigma}$ the kernel matrix and $\mathbf{y}$ the output values observed in training. However, as seen from Equation (3-13) the NMLL function contains the inverse of the kernel matrix $\boldsymbol{\Sigma}$ which is known to be less efficient and numerically less stable than a Cholesky decomposition [33]. The Cholesky decomposition exploits the positive-semi definite property of the kernel matrix and can be interpreted as a square root operation of a matrix [82]. The Cholesky decomposition of the kernel matrix Σ is the following:

$$\mathrm{chol}(\boldsymbol{\Sigma}) = \boldsymbol{L} \rightarrow \boldsymbol{\Sigma} = \boldsymbol{L}^T\boldsymbol{L}, \tag{3-15}$$

with $\boldsymbol{L}$ an uppertriangular matrix. By using the Cholesky decomposition, the following terms for the matrix inverse and log determinant are obtained:

$$\boldsymbol{\Sigma}\alpha = \mathbf{y} \rightarrow \alpha = (\boldsymbol{L})^{-1}\left(\boldsymbol{L}^T\right)^{-1}\mathbf{y}, \;\; \log\left(\det(\boldsymbol{\Sigma})\right) = 2\sum_{i=1}^{n}\log\left(L_{ii}\right). \tag{3-16}$$

By using the rewritten determinant and inverse of Equation (3-16), the following NMLL is obtained:

$$-\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n}\ln\left(L_{ii}\right) + \frac{1}{2}\mathbf{y}^{\mathrm{T}}\alpha + \frac{N}{2}\ln(2\pi), \tag{3-17}$$

which is ready to be implemented as an objective function in an optimization problem. The full derivation of the terms in Equation (3-16) and the NMLL of Equation (3-17) can be found in the Appendix A-2. For the implementation, the NMLL is optimized over the kernel hyperparameters by the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm since this algorithm is efficient for unconstrained non-convex and nonlinear optimization problems.

The implementation of the prediction solution of Equation (3-14) is also rewritten in terms of Cholesky factorizations. The resulting solution is the following:

$$\begin{aligned}\mu_{\mathrm{f}_*} &= \boldsymbol{\Sigma}_*^{\mathrm{T}}\alpha,\\ \sigma^2_{\mathrm{f}_*} &= \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^{\mathrm{T}}\left(\boldsymbol{L}\right)^{-1}\left(\boldsymbol{L}^T\right)^{-1}\boldsymbol{\Sigma}_*,\end{aligned} \tag{3-18}$$

where α and $\boldsymbol{L}$ are precalculated.

3-3-2 Implementation of the sparse Gaussian process

The approximated GP that is used in this thesis is the power expectation propagation (PEP) method of the posterior sparse approximations. Unlike the prior approximation methods, the posterior approximation methods never overfit the data because of the fact that the Kullback–Leibler (KL) divergence is minimized in the optimization problem. Minimizing the KL divergergence ensures that the approximate probability distribution matches the exact GP probability distribution. This is a major advantage for the posterior approximation methods. Next to the benefits fo the PEP method over prior approximation methods, the PEP method is also advantageous compared to the other posterior approximation methods. This comes from the fact that the method offers a hybrid connection which enables to switch between a prior and posterior approximation. The method is therefore very flexible and able to produce accurate model descriptions for a large variety of problems. Accordingly, it is argued that the method outperforms other state-of-the-art sparsification methods in terms of prediction power [65].

The authors in [65] suggest to take the switching constant α to be between 0.5 and 0.8 to obtain the best all round performance. Therefore, in this thesis a switching constant of 0.5 is used. However, the PEP MLL of Equation (2-48) also suffers from numerical instabilities. Therefore, to enhance implementation, Equation (2-48) should also be rewritten in terms of the Cholesky factorization. The coming calculation for numerical stability is inspired by [65]. First, the MLL of Equation (2-48) is restated:

$$\begin{aligned}\mathcal{L}_{\mathrm{PEP}} = &-\frac{n}{2}\log 2\pi - \frac{1}{2}\log\mid \alpha K_{\mathrm{FITC}} + \boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}\mid\\ &-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\left(\alpha K_{\mathrm{FITC}} + \boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}_n\right)^{-1}\boldsymbol{y} - \frac{1-\alpha}{2\alpha}\operatorname{tr}\left[\log\left(I + \frac{\alpha}{\sigma_\epsilon^2}\left(\boldsymbol{K}_{\mathbf{ff}} - \boldsymbol{Q}_{\mathbf{ff}}\right)\right)\right],\end{aligned} \tag{3-19}$$

with the following matrices:

$$\begin{aligned}K_{\mathrm{FITC}} &= \operatorname{diag}\left[K_{\mathbf{ff}} - Q_{\mathbf{ff}}\right],\\ Q_{\mathbf{ff}} &= K_{\mathbf{fu}}K_{\mathrm{uu}}^{-1}K_{\mathbf{uf}}.\end{aligned} \tag{3-20}$$

It is observed from Equations (3-19) and (3-20) that the MLL of the sparse PEP method also containse an inverse term $\left(\alpha K_{\mathrm{FITC}} + \boldsymbol{Q}_{\mathbf{ff}} + \sigma_\epsilon^2\boldsymbol{I}\right)^{-1}$. However, this inverse term contains another inverse which originates from the inducing input kernel K_{uu}^{-1}. Therefore, the inverse term is first be rewritten in a Cholesky decomposition for efficiency and numerical stability purposes. For this purpose, the Woodbury identity matrix [83] is used which is calculated as follows:

$$\left(K_{\mathrm{d}} + K_{\mathbf{fu}} K_{\mathrm{uu}}^{-1} K_{\mathrm{u}\mathbf{f}}\right)^{-1} = K_{\mathrm{d}}^{-1} - K_{\mathrm{d}}^{-1} K_{\mathbf{fu}} \left(K_{\mathrm{uu}} + K_{\mathrm{u}\mathbf{f}} K_{\mathrm{d}}^{-1} K_{\mathbf{fu}}\right)^{-1} K_{\mathrm{u}\mathbf{f}} K_{\mathrm{d}}^{-1}, \tag{3-21}$$

with the following diagonal matrix:

$$K_{\mathrm{d}} = \alpha \operatorname{diag}\left[K_{\mathbf{ff}} - Q_{\mathbf{ff}}\right] + \sigma_{\epsilon}^2 \boldsymbol{I}. \tag{3-22}$$

The Woodbury identity matrix representation of Equation (3-21) is suitable for the rewrite to Cholesky decomposition terms. This is possible since it is easy and numerical stable to invert the diagonal matrix K_{d}. The derivation of the numerical stable MLL of the PEP method is presented in Appendix A-3. The result of this calculation is the following:

$$\begin{aligned} \mathcal{L}_{\mathrm{PEP}} = &- \frac{n}{2} \log 2\pi - \sum_{i=1}^{m} \log\left(L_{\mathrm{B}ii}\right) - \frac{1}{2} \operatorname{tr}\left[\log\left(K_{\mathrm{d}}\right)\right] - \frac{1}{2} \boldsymbol{y}^{\mathrm{T}} \left(K_{\mathrm{d}}^{-1}\right) \boldsymbol{y} + \boldsymbol{y}^{\mathrm{T}} \left(\boldsymbol{C}\boldsymbol{C}^{T}\right) \boldsymbol{y} \\ &- \frac{1-\alpha}{2\alpha} \operatorname{tr}\left[\log\left(I + \frac{\alpha}{\sigma_{\epsilon}^2}\left(\boldsymbol{K}_{\mathbf{ff}} - K_{\mathrm{d}} \boldsymbol{A}^{T} \boldsymbol{A}\right)\right)\right], \end{aligned} \tag{3-23}$$

with the following matrices:

$$\begin{aligned} \operatorname{chol}(\boldsymbol{K}_{\mathrm{uu}}) &= \boldsymbol{L}_{\mathrm{uu}}, \\ \boldsymbol{A} &= \left(\boldsymbol{L}_{\mathrm{uu}}^{T}\right)^{-1} K_{\mathrm{u}\mathbf{f}} K_{\mathrm{d}}^{-0.5}, \\ \boldsymbol{B} &= \boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^{T}, \\ \operatorname{chol}(\boldsymbol{B}) &= \boldsymbol{L}_{\mathrm{B}}, \\ \boldsymbol{C}^{T} &= \left(\boldsymbol{L}_{\mathrm{B}}^{T}\right)^{-1} \boldsymbol{A} K_{\mathrm{d}}^{-0.5}, \end{aligned} \tag{3-24}$$

where u indicate the inducing input points, and $\mathbf{f}$ the observed output values. The MLL of Equation (3-23) can be used as an objective function in an optimization problem for finding the sparse hyperparameters. In this thesis, the sparse hyperparameters are also found by using the BFGS algorithm for optimization.

Also, the approximate posterior of the PEP method for doing predictions is rewritten in terms of the matrices of Equation (3-24). For this purpose, the following representation is obtained:

$$\begin{aligned} \mu_{\mathrm{f}_*}^{\mathrm{PEP}} &= \boldsymbol{K}_{\mathrm{f}_*\mathrm{u}} \left(\boldsymbol{L}_{\mathrm{uu}}\right)^{-1} \left(\boldsymbol{L}_{B}\right)^{-1} \boldsymbol{C}^{T} \boldsymbol{y} = \boldsymbol{K}_{\mathrm{f}_*\mathrm{u}} \alpha_{PEP} \\ \sigma_{\mathrm{f}_*}^{\mathrm{PEP}} &= \boldsymbol{K}_{\mathrm{f}_*\mathrm{f}_*} - \boldsymbol{K}_{\mathrm{f}_*\mathrm{u}} \left(\left(\boldsymbol{L}_{\mathrm{uu}}\right)^{-1} \left(\boldsymbol{L}_{\mathrm{uu}}^{T}\right)^{-1} - \left(\boldsymbol{L}_{\mathrm{uu}}\right)^{-1} \left(\boldsymbol{L}_{B}\right)^{-1} \left(\boldsymbol{L}_{B}^{T}\right)^{-1} \left(\boldsymbol{L}_{\mathrm{uu}}^{T}\right)^{-1}\right) \boldsymbol{K}_{\mathrm{u}\mathrm{f}_*}, \end{aligned} \tag{3-25}$$

where α_{PEP}, $\boldsymbol{L}_{\mathrm{uu}}$ and $\boldsymbol{L}_{\mathrm{B}}$ are precalculated.

Now that it is known how the sparse algorithm is implemented, it is still unknown how the inducing variables $\boldsymbol{u}$ are chosen. The inducing input points $\boldsymbol{x}_{\mathrm{u}}$ are chosen to be a subset of the

training inputs. The method for selecting these inputs is based on a theory introduced in [84]. Here, the inducing inputs are chosen based on a selection criterion that is computationally attractive to evaluate. The criterion selects the inducing points as a subset from the exact training dataset. The subset of inducing points is constructed by selecting the training points that contain the most information for a trained exact GP model. This selection is based on an acceptance threshold. Note that the inducing points contain the most information if they have a low value when the kernel covariance function is evaluated with respect to a training input. The heuristic behind this concept is the fact that a low covariance of an evaluated training point implies that the point does not have a lot in common with the other included datapoints in the dataset.

3-3-3 Implementation of the Gaussian process model predictive control framework

After finishing the modelling phase of the algorithm, the models of θ_1 and θ_2 are used in a model predictive control framework for controlling the double pendulum system. The MPC controller is implemented in an online fashion of which the control inputs are calculated based on two internal sparse GP models. This calculation is performed at each time instance which makes the controller single-shooting. The internal sparse models are used for making multi-step-ahead predictions over the prediction horizon of the MPC of angles θ_1 and θ_2. Therefore, the objective function that is used in the MPC controller is the following:

$$J(\mu_\theta, \theta_{\text{ref}}, \mathbf{u}) = \sum_{\tau=0}^{N_\text{p}-1} Q_1 \left(\mu_{\theta_{1,t+\tau}} - \theta^{\text{ref}}_{1,t+\tau}\right)^2 + Q_2 \left(\mu_{\theta_{2,t+\tau}} - \theta^{\text{ref}}_{2,t+\tau}\right)^2 + \Delta u_{t+\tau}^T S \Delta u_{t+\tau}, \quad (3\text{-}26)$$

where the scalars Q_1, Q_2 and S are weights, $\mu_{\theta_\cdot}$ the model trajectory for the specified angle $\theta_\cdot$, $\theta^{\text{ref}}_\cdot$ the reference trajectory for the specified angle $\theta_\cdot$ and Δu_i the difference between two consecutive control inputs $\Delta u_i - \Delta u_{i-1}$ that prevents a fast fluctuation of the control inputs.

The objective function of Equation (3-26) is also subjected to constraints. The constraints are designed in the way that the sparse GP model is used to calculate $N_\text{p} - 1$ steps ahead and to keep the control input within actuator bounds. Therefore, the GP-MPC problem is defined as follows:

$$\begin{aligned} \text{minimize } & J(\mu_\theta, \theta_{ref}, \mathbf{u}) \\ \text{subject to } & \mu_{\theta_{1,t+\tau}} = \boldsymbol{K}_{\theta_1}(\boldsymbol{x}_{*,t+\tau-1}, \boldsymbol{x}_\text{u})\, \alpha_{\theta_{1,\text{PEP}}} \\ & \mu_{\theta_{2,t+\tau}} = \boldsymbol{K}_{\theta_2}(\boldsymbol{x}_{*,t+\tau-1}, \boldsymbol{x}_\text{u})\, \alpha_{\theta_{2,\text{PEP}}} \\ & u_{t+\tau} \in \mathcal{U}, \end{aligned} \quad (3\text{-}27)$$

where $\boldsymbol{K}_{\theta_i}$ indicates the kernel matrix of the specified angle displacement, $\boldsymbol{x}_{*,t+\tau-1}$ the regressor at time $t+\tau-1$ that is used for prediction, $\boldsymbol{x}_\text{u}$ the inducing points and $\alpha_{\theta i,\text{PEP}}$ the PEP model matrix for the specified angle displacement. Note that a naive approach is used for the uncertainty propagation since the uncertainty of the predictions is not propagated over the prediction horizon. In this implementation, uncertainty propagation is not attractive since

the controller only uses the mean values of the sparse GP models. This design choice is made since the controller requires to solve a nonlinear non-convex optimization problem online at every iteration of the controller. It is known that this kind of problems are computationally challenging to solve, especially in an online fashion. Therefore, the objective function and the constraints of the MPC problem are kept to a minimum to reassure a faster performance.

The software that is used in this thesis for solving the nonlinear non-convex problem is Casadi [85]. Casadi is an open source tool for nonlinear optimization that is also suitable for using in a nonlinear model predictive controller (NMPC). The software works both in Matlab and Simulink. It can also be used in real life control applications.

Chapter 4

Results

This chapter presents the results of the proposed GP-MPC algorithm obtained from the simulation environment and the physical setup. First, the results from the simulation are discussed which provide initial insights in the problem and the first answers to the research questions of this thesis. The simulation environment is used to explore what kernel function and model structure is most suitable to describe the dynamics of the considered system and for use in the foreseen control structure. Also, the most appropriate settings for implementing the GP-MPC framework in realtime are outlined in the simulation phase. After finding the first results, the GP-MPC algorithm is tested on the real-world setup.

This chapter is structured as follows. Section 4-1 compares the several model structures and kernel functions that have been specified previously. For this purpose, an exact GP training is employed of which the performance of the resulting models is compared. The outperforming combination of model structure and kernel function is used for implementation in the eventual control algorithm.

Section 4-2 describes the power expectation propagation (PEP) approximation method for sparsifying the previously obtained exact GP model. The resulting approximate GP is evaluated regarding the performance with respect to the number of inducing points.

Section 4-3 implements the approximate GP models in the proposed GP-MPC framework and are tested. The GP-MPC algorithm is tested on performance and calculation time with respect to the prediction horizon and the number of inducing points.

Finally, the algorithm is tested on the physical setup in Section 4-4. This involves implementing and testing the proposed GP-MPC algorithm and evaluating the controller performance, in terms of reference tracking and disturbance rejection.

4-1 Comparison model structures and kernel funtions

This section compares the kernel functions and model structures that are specified previously. The comparison involves two GP models, one for each angle of the considered system. The

reason for considering two models is the fact that GPs are restricted to model one dynamical function at a time while it is foreseen that the proposed GP-MPC controller performs multivariable control on the single input multiple output (SIMO) double pendulum system. Multivariable MPC control requires to predict multiple system values over the prediction horizon.

For initial validation purposes, both the dynamical models are trained using exact GPs with an identification dataset $\mathcal{D}_{\text{id}}$ consisting of 700 regression vectors $\boldsymbol{x}_i(k)$ and augmented with output measurements $y_i(k)$, i.e., $\mathcal{D}_{\text{id}} = (\boldsymbol{x}_i(k), y_i(k))_{i=1}^{700}$. Observe that output values $y_i(k)$ in the dataset are unique regarding the models for θ_1 and θ_2. The dataset is obtained by providing an amplitude modulated pseudo random binary sequence (APRBS) signal with step size $h = 0.05$ s to the simulated double pendulum system. The outputs θ_1 and θ_2 of the double pendulum model are corrupted with Gaussian white noise with variance $\text{var}(\sigma_\epsilon^2) = 0.005$ which is comparable to the sensor noise of the physical system. After training, the results are validated by using the variance accounted for (VAF) and mean standardised log loss (MSLL) performance measures. The validation dataset $\mathcal{D}_{\text{val}}$ consists of 300 (new) regression vectors and target values of which the input/output signals are visualized in Figure 4-1. The outperforming combination of model structure and kernel function eventually forms the proposed GP-MPC algorithm for the data-driven control of a double pendulum system.

The remainder of this section discusses the model structures that are tested on the specified kernel functions. The model structures are selected based on their attractive implementation properties and promising modelling power for nonlinear systems. In Section 4-1-1, the nonlinear autoregressive model with exogenous input (NARX) structure is tested in both a one-step-ahead and multiple-step-ahead prediction. The difference between one-step-ahead and multiple-step-ahead prediction is how the data is used for validation. Also, the nonlinear output error (NOE) and augmented nonlinear state space (NSS) model structures are evaluated in a same manner in Section 4-1-2 and Section 4-1-3 respectively.

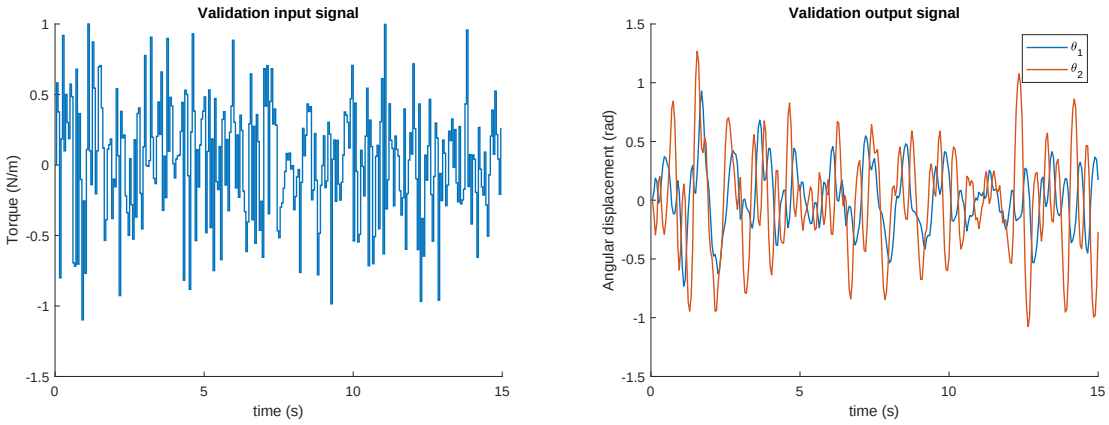


Figure 4-1: Validation signals used for constructing the validation dataset. The APRBS input signal (left) to the system excites the system persistently and is suitable for nonlinear system identification. The output response (right) of the double pendulum system consists of the angles θ_1 and θ_2. The input/output validation signals indicate that the system is SIMO.

4-1-1 Comparison GP-NARX method

This section discusses the validation results on the GP-NARX method. The NARX model structure consists of a regression vector containing true measured input/output values which are structured in an auto-regressive fashion. For validating the results, the GP-NARX is evaluated in both one- and multiple-step-ahead prediction. The one-step-ahead prediction is considered for testing if the GP-model retrieves a correct relation from the true data, and the multiple-step-ahead is used for validating if the model is suitable in a model predictive control framework. That is because the MPC controller calculates the control input based on multi-step-ahead predictions of the model over the prediction horizon N_p. The amount of regressors used in the NARX regression vector is $n = 4$ past outputs of both θ_1 and θ_2, and $m = 4$ past inputs. This model order is retrieved after linear analysis of the first-principles model in which it is assumed that the resulting NARX model is similar to a concatenation of local linear ARX models. Eventually, the NARX model structure is tested on the three kernel functions which are previously introduced.

The performance of the GP-NARX models for each of the mentioned kernel functions are validated using the VAF and MSLL performance measure on both the models θ_1 and θ_2. The VAF performance measure calculates the percentile of similarity between the real and predicted output values of which mediocre models achieve a VAF of $70\% < \mathrm{VAF} \leq 85\%$ and high performing models have VAF values higher than 85%. Next to this, the MSLL performance measure also considers the uncertainty of the prediction with respect to the real value by penalizing certain and incorrect prediction values while 'rewarding' (un)certain and (in)correct predictions. The models containing lower MSLL values are referred to as better models.

The values of the performance measures for the one-step-ahead prediction are presented in Table 4-1. For obtaining the values in the table, the GP model output is compared to the real output of the signal. Figure 4-2 visualizes an example of the one-step-ahead prediction of a GP-NARX model with the squared exponential (SE) kernel which is trained, evaluated and compared against the validation dataset. After repeating this process for all model structures and kernel functions in Table 4-1, it is concluded that each combination of kernel functions with the GP-NARX method result in a proper one-step-ahead predictor for both models θ_1

	Sq. Exp. (SE)		Periodic (P)		Composite (SE × P)	
	VAF	MSLL	VAF	MSLL	VAF	MSLL
Model θ_1	98.9	-2.78	99.7	-2.86	99.6	-2.94
Model θ_2	99.7	-3.01	99.6	-2.87	99.8	-3.22

Table 4-1: Performance comparison of the NARX model structure with different kernel functions using a one-step-ahead prediction. Both the models of θ_1 and θ_2 are validated. The used performance measures are the VAF and MSLL. The VAF value indicates the percentile of similarity between the prediction and the measured output signal. The MSLL value also incorporates a measure of uncertainty of which a lower value indicates a better model. Observe that the one-step-ahead prediction with the NARX method result in high VAF values ($> 85\%$) and negative MSLL values. This points out the high performance of the models in a one-step-ahead prediction.

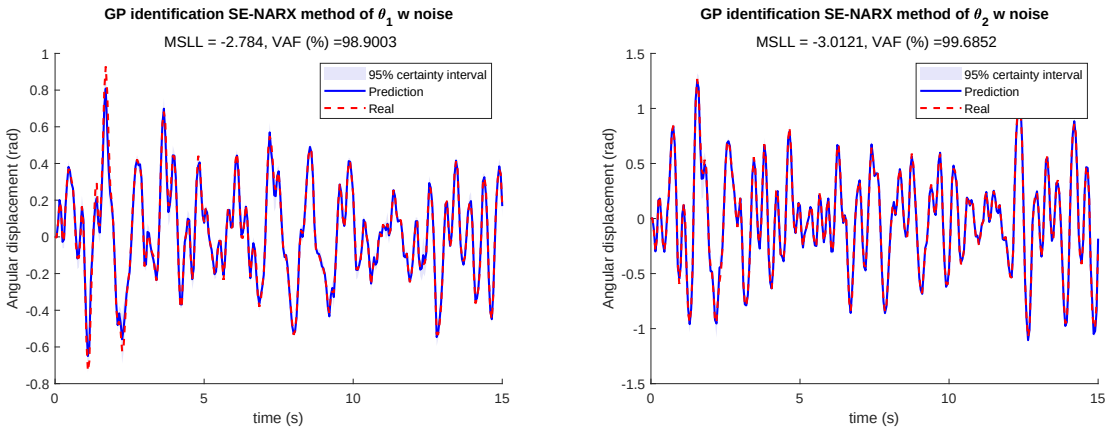


Figure 4-2: Example of a validation of the one-step-ahead prediction using the GP-NARX method with the SE kernel function. The models are trained with 700 regression vectors and validated with the validation dataset of Figure 4-1. The figure validates the GP-NARX model for θ_1 (left) and θ_2 (right). The subtitle of the figure indicates a negative MSLL and a high VAF for both models which show that the model prediction is similar to the validation signal (VAF $> 85\%$) and the uncertainty in the model is well distributed (MSLL < 0).

and θ_2. The high VAF values indicate that the mean trajectory of the models is around 99% similar to the real outputs, while the negative MSLL value also indicates that both the predictions and the uncertainty of the models are well fitted. This is observed for all the investigated kernel functions which are therefore of similar performance.

The one-step-ahead prediction is however not fully representative for the control application that is used in this thesis, as the GP model is used in an MPC controller. In an MPC framework, the one-step-ahead predictions are fed back to the regression vector for predicting the model N_p timesteps ahead. So, the data of the true measurements is in particular used in the regression vector for predicting the first timestep in the prediction horizon N_p, and the consecutive timesteps in the prediction horizon also use the data of previous predictions of the GP model. Therefore, the NARX prediction vector for predicting the model over the horizon N_p is as follows:

$$
\begin{aligned}
\boldsymbol{x}_*(k) =& \big[\theta_1(k) \quad \theta_1(k-1) \quad \cdots \quad \theta_1(k-n+1) \quad \theta_2(k) \quad \theta_2(k-1) \quad \cdots \\
& \qquad \cdots \quad \theta_2(k-n+1) \quad u(k) \quad u(k-1) \quad \cdots \quad u(k-m+1)\big]^T \\
\boldsymbol{x}_*(k+1) =& \big[\hat{\theta}_1(k+1) \quad \theta_1(k) \quad \cdots \quad \theta_1(k-n+2) \quad \hat{\theta}_2(k+1) \quad \theta_2(k) \quad \cdots \\
& \qquad \cdots \quad \theta_2(k-n+2) \quad u(k+1) \quad u(k) \quad \cdots \quad u(k-m+2)\big]^T \\
& \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\boldsymbol{x}_*(k+N_\mathrm{p}-1) =& \big[\hat{\theta}_1(k+N_\mathrm{p}-1) \quad \hat{\theta}_1(k+N_\mathrm{p}-2) \quad \cdots \quad \hat{\theta}_1(k-n+N_\mathrm{p}) \quad \hat{\theta}_2(k+N_\mathrm{p}-1) \quad \cdots \\
& \quad \cdots \quad \hat{\theta}_2(k+N_\mathrm{p}-2) \quad \hat{\theta}_2(k-n+N_\mathrm{p}) \quad u(k+N_\mathrm{p}-1) \quad u(k+N_\mathrm{p}-2) \quad \cdots \quad u(k-m+N_\mathrm{p})\big]^T,
\end{aligned}
\tag{4-1}
$$

where $\hat{\theta}_1(\cdot)$ and $\hat{\theta}_2(\cdot)$ are the predictions of the corresponding GP model, and $\boldsymbol{x}_*(\cdot)$ is the regression vector for predicting the value multiple timesteps ahead. So, the prediction vector of the NARX method for predicting multiple timesteps ahead is dependent on predictions of

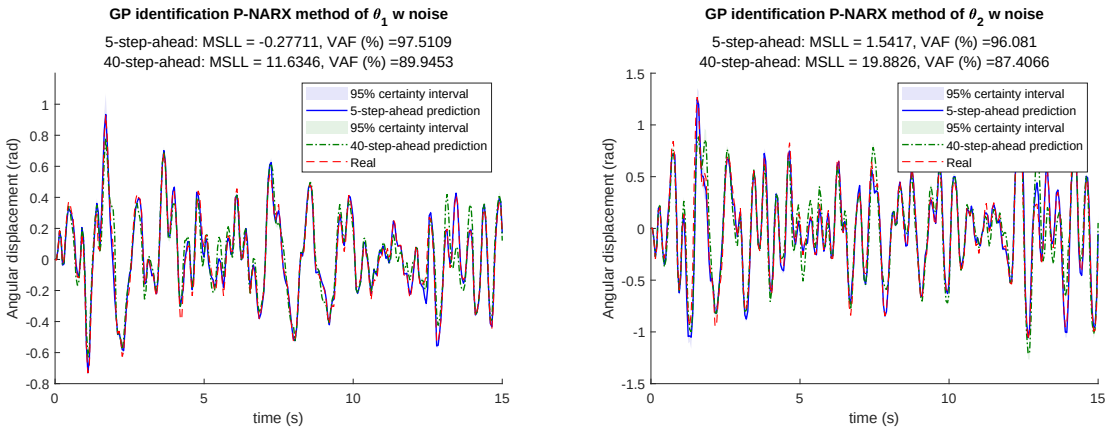


Figure 4-3: Example of a validation comparison of the 5- and 40-step-ahead prediction using the GP-NARX method with the periodic kernel function. The models θ_1 (left) and θ_2 (right) are trained with 700 regression vectors and validated with the validation dataset of Figure 4-1. By increasing the N_p-step-ahead prediction horizon, the VAF value is slightly decreases and the MSLL value is increases. So, the model deteriorates if used to predict further ahead in time which is expected since the error of the prediction accumulates. Both the models still obtain a high performance (VAF $> 85\%$) and the 5-step-ahead prediction also shows proper uncertainty bounds (low MSLL). However, the uncertainty of the 40-step-ahead prediction model is less representative (far positive MSLL). This is probably due to the used zero-variance method which does not account for the uncertainty propagation of previous predictions.

both the models θ_1 and θ_2.

When using the NARX model in a controller, the first regressor of the multiple-step-ahead prediction is updated in each iteration of the controller with the true measurements. However, visualizing the trajectories for N_p-step-ahead prediction at each time instance leads to confusing figures in the validation stage. Therefore, another approach is required for validating the model in a multiple-step-ahead prediction. For validation purposes, the GP models of θ_1 and θ_2 are predicted N_p timesteps ahead before a new true (simulation) measurement regression vector becomes available. This means that each N_p-step-ahead prediction interval of Equation (4-1) is concatenated through the validation interval. In other words, if the prediction horizon N_p equals 20, the signal is validated by interconnecting $300/20 = 15$ prediction intervals.

An example for visualizing the multiple-step-ahead validation is presented in Figure 4-3 where the GP-NARX model structure is trained with an SE kernel and validated in a 5 and 40-timestep-ahead prediction. This procedure is repeated for every combination of model structure and kernel function of which the results are presented in Table 4-2. Table 4-2 observes that the performance of the model decreases if the model is predicted further ahead in time due to the decreasing VAF and increasing MSLL value. However, the VAF values indicate that the GP-NARX models remain high performing ($> 85\%$) for each combination of kernel function. Also, the MSLL values increase significantly if predicted further in time which is probably due to certain but less accurate predictions. This is also referred to as an overconfident model. The overconfident models are probably a result of the zero-variance method used for uncertainty propagation in a multiple-step-ahead prediction. The zero-variance method

		Sq. Exp. (SE)		Periodic (P)		Composite (SE × P)	
		VAF	MSLL	VAF	MSLL	VAF	MSLL
5-step	Model θ_1	96.5	-1.48	97.5	-0.277	98.7	-1.76
	Model θ_2	96.8	2.29	96.1	1.54	98.4	0.53
10-step	Model θ_1	94.7	0.606	96.5	1.17	97.2	0.0173
	Model θ_2	95.2	6.32	92.4	4.91	96.8	5.00
20-step	Model θ_1	91.9	3.57	92.3	6.26	95.2	1.25
	Model θ_2	91.7	13.1	88.0	11.4	91.1	12.6
40-step	Model θ_1	87.6	8.46	89.9	11.6	85.1	16.0
	Model θ_2	85.9	23.1	87.4	19.9	76.4	57.5

Table 4-2: Performance comparison of the NARX model structure with different kernel functions using a five-, ten-, twenty- and forty-step-ahead prediction for models θ_1 and θ_2. The VAF performance measure indicates the similarity of the model prediction with the validation signal which deteriorates if the model is predicted further ahead in time. This is expected since the model predictions rely on its previous predictions which accumulate errors over time. However, the VAF performance indicates a high model performance (VAF >85%) except for the model θ_2 with a composite kernel predicting 40 timesteps ahead. This model has a mediocre model performance ($70\% <$ VAF $\leq 85\%$) that might still be suitable for control. Furthermore, the increasing MSLL value indicates that the multiple-step-ahead predictions become overconfident (certain but less accurate) if predicted further in time. This is probably due to the zero-variance method used for predicting multiple-steps-ahead that does not account for propagated uncertainty. It is therefore that the MSLL values show worse model performance if the model is predicted further in time.

ignores the uncertainty that propagates in each consecutive prediction. It is further observed that all of the combinations of kernel functions are again of similar performance.

4-1-2 Comparison GP-NOE method

This section discusses the results for the GP-NOE model structure. The NOE model structure uses a regression vector containing the past prediction values of the model which are structured in an auto-regressive manner. This model structure reduces the simulation error by fitting the predicted values $\hat{y}_i(k)$ to the corresponding measured output $y_i(k)$. The GP-NOE model is also evaluated on the three indicated kernel functions. The considered model order of the NOE regression vector is equal to the regression vector used for the NARX structure, i.e., the considered regression vector contains $n = 4$ past prediction outputs and $m = 4$ past inputs. After training, the GP-NOE models are validated by using the same validation strategy as in Section 4-1-1 where the model is tested in one-step-ahead and multiple-step-ahead prediction

	Sq. Exp. (SE)		Periodic (P)		Composite (SE × P)	
	VAF	MSLL	VAF	MSLL	VAF	MSLL
Model θ_1	17.5	-0.108	0.383	-0.00252	0.00	-0.00258
Model θ_2	1.81	0.00793	0.00553	0.0192	0.00100	0.0197

Table 4-3: Performance comparison of the NOE model structure with different kernel functions using a one-step-ahead prediction for both models θ_1 and θ_2. The VAF performance measures show an inferior performance (VAF $\leq 50\%$) of the GP-NOE models in a one-step-ahead prediction which is also obtained after visual expectation. As indicated in the text, this bad model performance is possibly caused in the training phase. However, the low MSLL values indicate a proper one-step-ahead predictor which possibly originated from high uncertainties in the GP-NOE model. The MSLL formula diminishes its value if high error predictions are uncertain.

scenarios.

The results for the one-step-ahead prediction are presented in Table 4-3. The low VAF values indicate that the model is not sufficiently describing the double pendulum system in a one-step-ahead prediction which is also observed after visual inspection. The bad performance of the model is possibly caused in the training phase due to increasingly deteriorating predictions of the GP-NOE model. The proposed GP-NOE model structure predicates its training dataset upon predictions of both the models θ_1 and θ_2 which are obtained by retraining and predicting

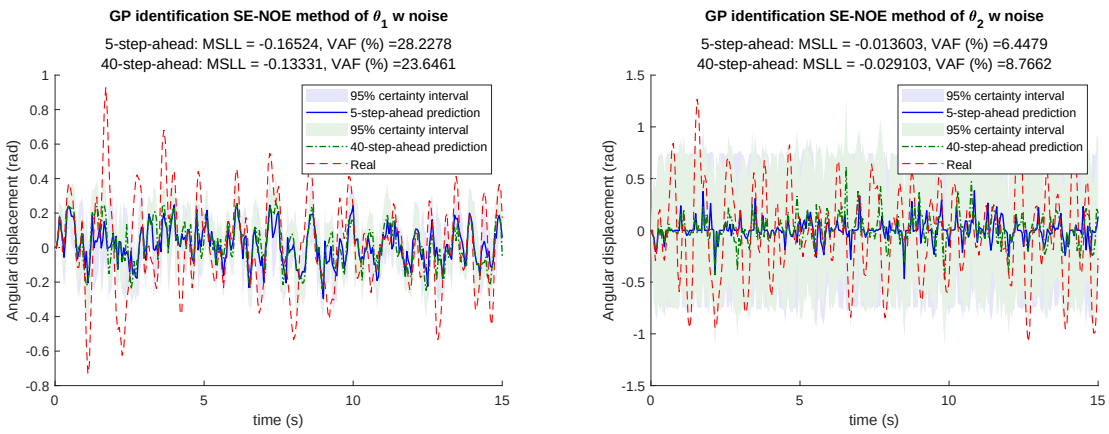


Figure 4-4: Example of a validation comparison of the 5- and 40-step-ahead prediction using the GP-NOE method with the SE kernel function. The models θ_1 (left) and θ_2 (right) are trained with 700 regression vectors and validated with the validation dataset of Figure 4-1. Observe that the model performs better if predicted further ahead in time in comparison to the one-step-ahead prediction, but the GP-NOE method still show an inferior performance (VAF $\leq 50\%$) for all kernel functions. As explained in the text, this might be due to the training phase where the dataset with model predictions deteriorated over time because of suboptimal hyperparameter solutions of the negative marginal log likelihood (NMLL) optimization. However, the low MSLL values show that the model has sufficient performance which is possibly a result of the definition of the MSLL function, i.e., the MSLL formula diminishes prediction errors that contain high uncertainty.

		Sq. Exp. (SE)		Periodic (P)		Composite (SE × P)	
		VAF	MSLL	VAF	MSLL	VAF	MSLL
5-step	Model θ_1	28.2	-0.165	0.00	0.00601	0.00	0.00226
	Model θ_2	6.45	-0.0136	0.00	0.0269	0.0387	0.0196
10-step	Model θ_1	22.3	-0.129	0.00	0.00412	0.235	0.000501
	Model θ_2	8.92	-0.0254	0.00	0.0239	0.0413	0.0195
20-step	Model θ_1	25.0	-0.142	0.00	0.00605	0.200	0.000424
	Model θ_2	10.9	-0.0369	0.00	0.0323	0.0647	0.0194
40-step	Model θ_1	23.6	-0.133	0.00	-0.000192	0.00	0.00172
	Model θ_2	8.77	-0.0291	0.00	0.0294	0.0333	0.0195

Table 4-4: Performance comparison of the NOE model structure with different kernel functions using a five-, ten-, twenty- and forty-step-ahead prediction for both models θ_1 and θ_2. The VAF value for the SE kernel indicates a higher performance compared to the one-step-ahead predictions which should be the case since a feature of the NOE structure is a minimal simulation error. However, the model is still inferior and not usable in the control framework VAF $< 50\%$. This is also observed for the models trained with the other kernel functions. As indicated in the text, this possibly comes from the training phase where the model deteriorated due to suboptimal hyperparameters from the NMLL optimization. Furthermore, the low MSLL values indicate a proper model which is incorrect after visual inspection. This is possibly due to the nature of the MSLL formula that diminishes high errors of uncertain predictions.

the models iteratively. However, these predictions deteriorate if one of both models becomes inaccurate because of suboptimal hyperparameter solutions from the NMLL optimization. This effect could not be prevented in this thesis and is observed as the main drawback of this modelling method for all kernel functions. Next to this observation, the GP-NOE method is advantageous in predicting the system multiple timesteps ahead since a feature of this method is a minimal simulation/full prediction error. It is therefore that a one-step-ahead validation is not the best measure for validating the GP-NOE method. Nonetheless, the negative MSLL value indicates a proper model, but after visual inspection, it is seen that this is probably caused by high uncertainties that are present in the model. The reason for the negative MSLL values is retrieved by analyzing the formula for the MSLL. The MSLL formula indicates that bad predictions are diminished in value if they occur with high uncertainty. Also, the formula squares the prediction error leading to low values for prediction errors below one. Referring to the validation output signal, the prediction errors are always below one for model θ_1 and most of the time for θ_2. Combining the high uncertainties in the predictions and the relatively low prediction error, the MSLL performance metric calculates low values.

The GP-NOE method is also tested in a multiple-step-ahead prediction where it is expected

that the performance increases in comparison to the one-step-ahead prediction because of the property of the NOE method to contain minimal simulation error. For this purpose, the prediction vector for predicting multiple timesteps ahead of Equation (4-1) is fed to the GP-NOE model to evaluate its appropriateness for MPC.

Figure 4-4 visualizes an example that compares the validation of a NOE model with the SE kernel for a 5- and 40-step-ahead prediction. The validation shows that the VAF performance of the models slightly increases compared to the one-step-ahead prediction. However, the models are still poorly performing and not suitable for application in the control framework because of a very low VAF value (VAF $< 50\%$). This is possibly due to the deteriorating performance in the training phase that is previously explained as the main drawback of the GP-NOE method in this thesis.

Table 4-4 provides the remaining results on the multiple-step-ahead prediction of the remaining kernel functions. It is concluded from the low VAF values that the GP-NOE method for all kernel combinations result in undescriptive models for all the multiple-step ahead predictions due to previously mentioned reasons. Therefore, the GP-NOE models are not feasible for implementation in the GP-MPC framework.

4-1-3 Comparison augmented GP-NSS method

The last model structure that is considered in this thesis is the augmented NSS model structure. The regression vector used for this method contains (all) the discrete-time states of the system that are fed to the GP in an auto-regressive manner. This regression vector differs from the NARX structure in the sense that the vector contains an approximation of the velocity of the system, and, the model is of first order. So, the augmented GP-NSS regression vector contains the angles, the augmented rotational velocities and the control input of the system.

Once more, the augmented NSS model structure is validated in a one-step-ahead and multiple-step-ahead prediction. The results for the one-step-ahead predictions are visualized in Table 4-5. From the VAF and MSLL values it is concluded that the augmented GP-NSS method perform well as a one-step-ahead predictor due to a VAF higher than 85% and a low MSLL value for all kernel function combinations. Observe that the one-step-ahead prediction of the

	Sq. Exp. (SE)		Periodic (P)		Composite (SE $\times$ P)	
	VAF	MSLL	VAF	MSLL	VAF	MSLL
Model θ_1	96.1	-1.63	92.4	-1.29	97.2	-1.85
Model θ_2	98.0	-1.89	97.7	-1.84	97.7	-1.85

Table 4-5: Performance comparison of the augmented NSS model structure with different kernel functions using a one-step-ahead prediction for both models θ_1 and θ_2. It is concluded that the augmented GP-NSS method achieves a high one-step-ahead prediction performance for all kernel functions due to a high VAF (VAF $> 85\%$) and negative MSLL. The one-step-ahead performance of the augmented GP-NSS method is comparable to the GP-NARX method.

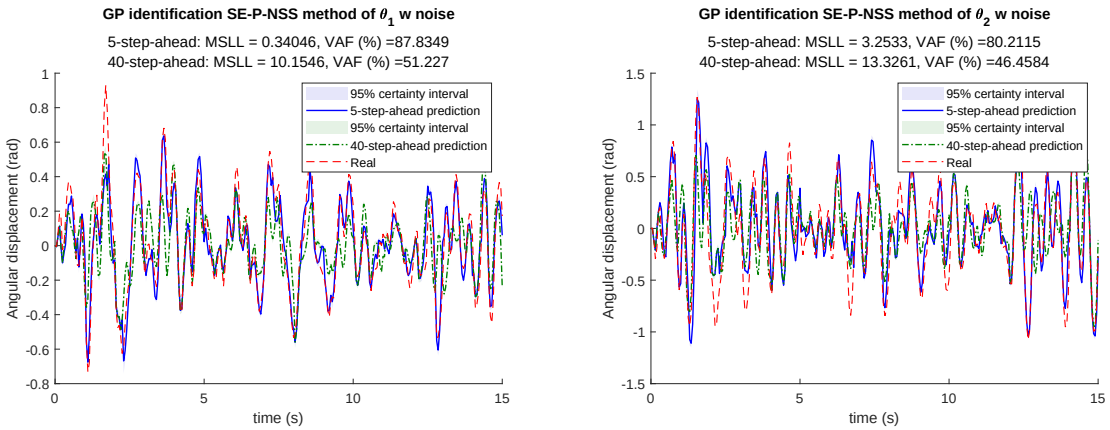


Figure 4-5: Example of a validation comparison of the 5- and 40-step-ahead prediction using the augmented GP-NSS method with the composite kernel function. The models θ_1 (left) and θ_2 (right) are trained with 700 regression vectors and validated with the validation dataset of Figure 4-1. Both the models show a mediocre ($70\% <$ VAF $< 85\%$ and relatively low MSLL) to high (VAF $> 85\%$ and low MSLL) performance when predicted a couple steps ahead in time. However, the models deteriorate fast in performance because error accumulation if predicted further in time which is seen from the decreasing VAF and increasing MSLL value. Compared to the GP-NARX method, the augmented GP-NSS method deteriorates faster in performance if predicted further in time and is therefore less suitable for incorporation in the GP-MPC framework.

augmented GP-NSS method performs similar to GP-NARX method resulting from the VAF and MSLL values which means that considering fewer regressors does not deteriorate the one-step-ahead performance. Furthermore, the SE and composite kernel achieve a slightly better VAF value for the model of θ_1 which make these models better performing on the validation set.

The augmented GP-NSS method is also validated in a N_p-step-ahead prediction. For this purpose, the following multiple-step-ahead NSS prediction vector is used:

$$\begin{aligned}
\boldsymbol{x}_*(k) =& \begin{bmatrix}\theta_1(k) & \theta_1(k)-\theta_1(k-1) & \theta_2(k) & \theta_2(k)-\theta_2(k-1) & u(k)\end{bmatrix}^T \\
\boldsymbol{x}_*(k+1) =& \begin{bmatrix}\hat{\theta}_1(k+1) & \hat{\theta}_1(k+1)-\theta_1(k) & \hat{\theta}_2(k+1) & \hat{\theta}_2(k+1)-\theta_2(k) & u(k+1)\end{bmatrix}^T \\
& \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\boldsymbol{x}_*(k+N_\mathrm{p}-1) =& \begin{bmatrix}\hat{\theta}_1(k+N_\mathrm{p}-1) & \hat{\theta}_1(k+N_\mathrm{p}-1)-\hat{\theta}_1(k+N_\mathrm{p}-2) & \hat{\theta}_2(k+N_\mathrm{p}-1) & \cdots \\
& \cdots \quad \hat{\theta}_2(k+N_\mathrm{p}-1)-\hat{\theta}_2(k+N_\mathrm{p}-2) \quad \cdots \quad u(k+N_\mathrm{p}-1)\end{bmatrix}^T
\end{aligned} \tag{4-2}$$

Figure 4-5 shows an example of the augmented GP-NSS models trained with the composite kernel where the prediction vector of Equation (4-2) is used for a 5- and 40-step-ahead prediction. The decreasing VAF and increasing MSLL values indicate that the models of θ_1 and θ_2 deteriorate significantly if predicted further ahead in time. The VAF value for the 40-step-ahead predictions even drops to a value around 50% which makes the models too inaccurate for the MPC implementation. The reason for the deteriorating models is possi-

		Sq. Exp. (SE)		Periodic (P)		Composite (SE × P)	
		VAF	MSLL	VAF	MSLL	VAF	MSLL
5-step	Model θ_1	82.6	0.410	42.5	2.59	87.8	0.341
	Model θ_2	81.7	4.15	62.8	9.60	80.2	3.25
10-step	Model θ_1	80.5	0.760	29.2	3.64	77.8	2.94
	Model θ_2	79.3	5.37	9.38	24.2	72.9	5.42
20-step	Model θ_1	69.6	2.58	0.00	6.64	70.2	5.23
	Model θ_2	66.8	10.8	0.00	66.8	60.5	9.09
40-step	Model θ_1	58.3	4.26	0.00	6.39	51.2	10.2
	Model θ_2	58.9	13.9	0.00	118	46.5	13.3

Table 4-6: Performance comparison of the augmented NSS model structure with different kernel functions using a five-, ten-, twenty- and forty-step-ahead prediction of both models θ_1 and θ_2. Observing the decreasing VAF and increasing MSLL values if predicted further in time, it is concluded that the augmented GP-NSS model structure deteriorates significantly. This is probably due to a fast error accumulation in the model. The model becomes insufficient for the GP-MPC framework if predicted further than 20 steps ahead in time because of a VAF lower than 70% with a far positive MSLL. This is seen for all kernel functions, especially the periodic kernel function. The periodic kernel function deteriorates very fast compared to the one-step-ahead prediction and is therefore not suitable to predict multiple timesteps ahead.

bly error propagation which is a feature of multiple-ahead-predictions. Error propagation is an accumulation of prediction errors at each consecutive prediction. In this case, the errors accumulate fast which is probably a downside of the augmented GP-NSS regression vector.

The remaining results of the multiple-step-ahead prediction are presented in Table 4-6. The table confirms the fast deteriorating performance of the augmented GP-NSS for multiple-step-ahead predictions. The combination that deteriorates the fastest is the periodic kernel function which obtains an inferior performance (VAF $< 50\%$ and high MSLL) for all considered prediction cases. This concludes that the model with a periodic kernel is infeasible to be used in a multiple-step-ahead prediction. However, the other kernel functions achieve better performance. The SE and composite kernel achieve a mediocre performance (70%VAF $< 85\%$ and relatively low MSLL) if predicted 10 timesteps ahead in time or less. Nonetheless, the performance of the models with a prediction horizon larger than $N_\mathrm{p} = 20$ is still insufficient (%VAF$<70\%$ and relatively high MSLL). Comparing the augmented GP-NSS method to the GP-NARX method, it is concluded from the performance measures that the GP-NARX method always achieves a higher performance.

4-1-4 Choice of model structure and kernel function

After obtaining the validation results on all combinations of kernel functions and model structures, a single combination is chosen for the remainder of this research. From the VAF and MSLL performance measures, it is concluded that the GP-NARX method outperforms the other model structures with respect to all examined kernel functions. Therefore, it is evident that this research is continued by selecting the GP-NARX model structure. However, the results of the GP-NARX method with respect to the kernel functions are quite similar as all combinations are performing sufficiently well on this method. For that reason, the kernel function that requires the least amount of hyperparameters is selected for continuing this thesis. So, the GP-NARX model structure with the SE kernel function is used in the remainder of this thesis.

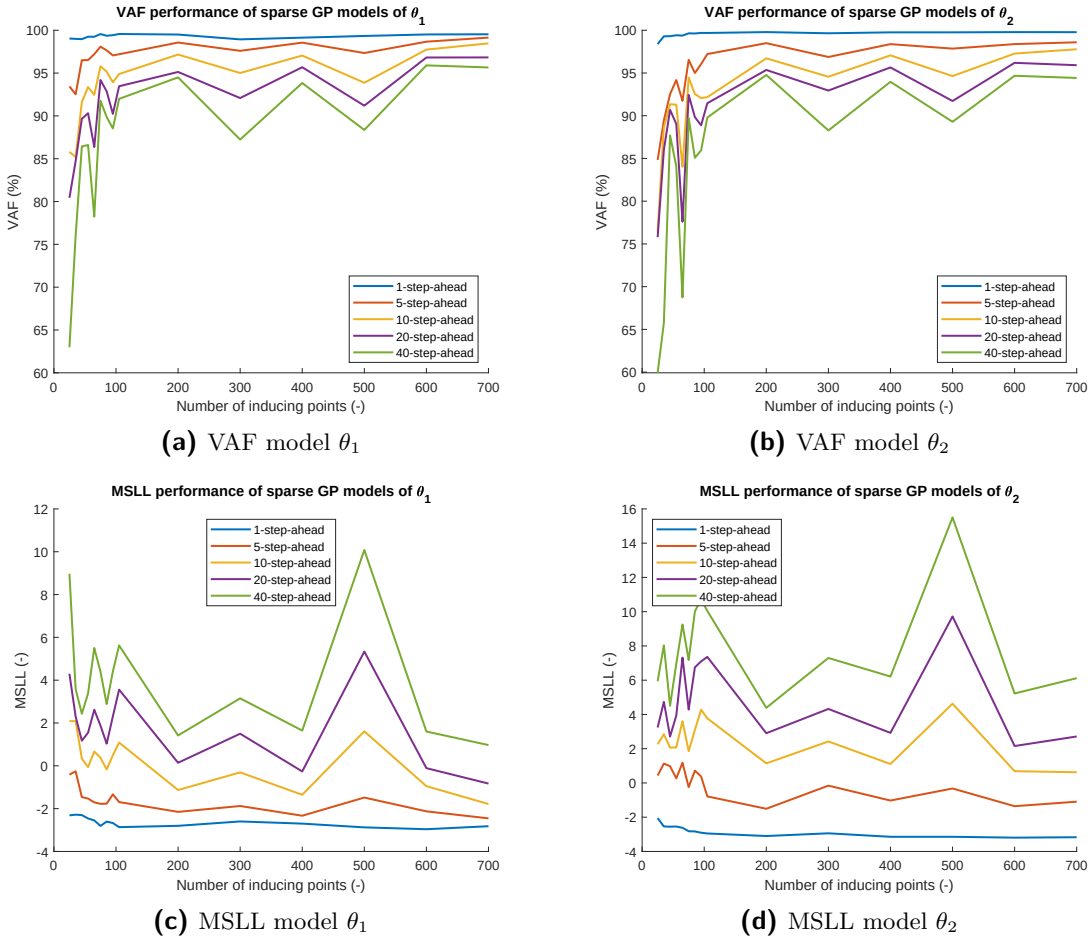
4-2 Learning approximate dynamic Gaussian process models

This section investigates the performance of the approximate GPs using the PEP algorithm with the previously selected model structure and kernel function. For examining the possibilities of dynamic modelling with approximate GPs, several approximate models are trained with different amounts of inducing points. The inducing points are obtained by selecting the datapoints from the exact GP training dataset that hold the most information for the exact GP-NARX model of the previous section. After training the approximate GP models, the models are verified by using the validation signals of Figure 4-1, and the VAF and MSLL performance measures.

Figure 4-6 visualizes both the VAF and MSLL performance of models θ_1 and θ_2 for different amounts of inducing points and different prediction horizons. The one- and multiple-step-ahead predictions are obtained by using the same validation strategy as in Section 4-1. All the figures observe that both models θ_1 and θ_2 deteriorate in performance if the prediction horizon enlarges. This is as expected since the models accumulate the error if predicted further ahead in time which was also seen in the previous sections. Also, the VAF and MSLL values indicate that the number of inducing points positively influences the performance of the approximate models due to an increasing trend in the VAF values and decreasing MSLL trend. However, some values differ from this trend which are outliers. For example, the approximate model with 500 inducing points experiences a significant increase in MSLL value when increasing the prediction horizon. After visual inspection and observing the VAF value (around 90%) of this model, it is concluded that this is possibly due to certain and erroneous predictions that indicate an overconfident model.

Analyzing the overall performance of the models, the conclusion is drawn that almost all models might be feasible to be used in the control framework since the models achieve mediocre ($70\% < \text{VAF} \leq 85\%$ and relatively low MSLL) to a high (VAF $> 85\%$ and low MSLL) performance. Nonetheless, some models that are 40 predicted timesteps ahead in time with low numbers of inducing points are performing insufficient (VAF $< 70\%$ and high MSLL) for control purposes.

In conclusion, it is possible to approximate the exact GP models for a significant decrease of considered datapoints while still obtaining mediocre ($70\% < \text{VAF} \leq 85\%$ and relatively low



(a) VAF model θ_1 **(b)** VAF model θ_2

(c) MSLL model θ_1 **(d)** MSLL model θ_2

Figure 4-6: VAF (**a** & **b**) and MSLL (**c** & **d**) performance of the approximated GP models θ_1 (**a** & **c**) and $\theta(2)$ (**b** & **d**) with respect to the number of inducing points and the length of the prediction horizon. Both the performance measures indicate that the models deteriorate if predicted further ahead in time which is observed for all numbers of inducing points. This is expected due to the fact that errors in the model accumulate if predicted further in time which was also observed previously. Furthermore, the increasing trend in the VAF values shows that the model performance increases if the model contains more inducing points. Also, the MSLL values show this trend except for the approximate model with 500 inducing points. This MSLL outlier is probably caused by an overconfident prediction model due to confident erroneous predictions as the VAF value of 90% indicates a good performing model. However, the conclusion is drawn that both models obtain a mediocre ($70\% <$ VAF $\leq 85\%$ and relatively low MSLL) to high performance (VAF $> 85\%$ and low MSLL) for all numbers of inducing inputs and prediction horizons except for some models that are predicted 40 timesteps ahead in time. High performance models are suitable to be incorporated for control and mediocre models might be suitable.

MSLL) to high (VAF $> 85\%$ and low MSLL) performing models. The mediocre performing models might be suitable in the control framework while the high performing models are certainly feasible for incorporation in the MPC framework.

4-3 Model predictive control using approximate Gaussian process models

This section incorporates the approximate GP models in the GP-MPC framework for testing the reference tracking performance. Furthermore, the calculation time of the controller is evaluated with respect to the number of inducing points of the approximate GP and the length of the prediction horizon. Both the performance and calculation time are necessary to investigate since the controller should perform well and the computational time of the control inputs should be calculated faster than the sampling time of the controller if the GP-MPC framework is incorporated in the physical setup. The GP-MPC is designed to calculate the control input for each iteration of the controller, i.e., the controller is single-shooting. Note that the resulting feasible set of sampling time and prediction horizon is used as a lower bound for implementing the GP-MPC algorithm in the real-time environment due to hardware restrictions.

The remainder of this section is consist of two parts. Section 4-3-1 evaluates the performance of the GP-MPC algorithm for reference tracking, and Section 4-3-2 tests the computational time of the controller.

4-3-1 Performance of reference tracking controller

This section evaluates the tracking performance of the controller by using the approximate GP models of the previous section. For testing the tracking performance of the controller the mean tracking error is considered and is calculated by:

$$\mu_{\text{ref}} = \frac{1}{N} \sum_{k=1}^{N} \mid y(k) - y_{\text{ref}}(k) \mid, \tag{4-3}$$

where N is the amount of simulated points, k the step time, y the observed output of the system and y_{ref} the output reference value.

The mean tracking error of Equation (4-3) is evaluated for analyzing the performance of the reference tracking controller with respect to the prediction horizon and number of inducing points. An example of the performance of the reference tracking controller is visualized in Figure 4-7 where the controller of Section 3-3-3 is implemented with an approximate GP model containing 400 inducing points and a prediction horizon of $N_{\text{p}} = 20$. It is observed that the reference is tracked with some overshoot before settling, and some fluctuations around the reference at 5 s $\leq t \leq$ 10 s. The overshoot might be due to the tight tuning of the controller by punishing the tracking errors while the fluctuations might be caused by bad performances of the model in this operating region.

The evaluation strategy for obtaining Figure 4-7 is repeated several times for different numbers of inducing points and prediction horizons. The results of the performance for both angle θ_1 and θ_2 are visualized in Figure 4-8 of which the standard deviations of the plot are provided in Appendix C-1–C-4. From the figure it is observed that both θ_1 and θ_2 obtain a lower tracking error if the system is controlled with a prediction horizon higher than $N_{\text{p}} = 10$. However, there is no significant difference obtained if the prediction horizon is increased from $N_{\text{p}} = 20$

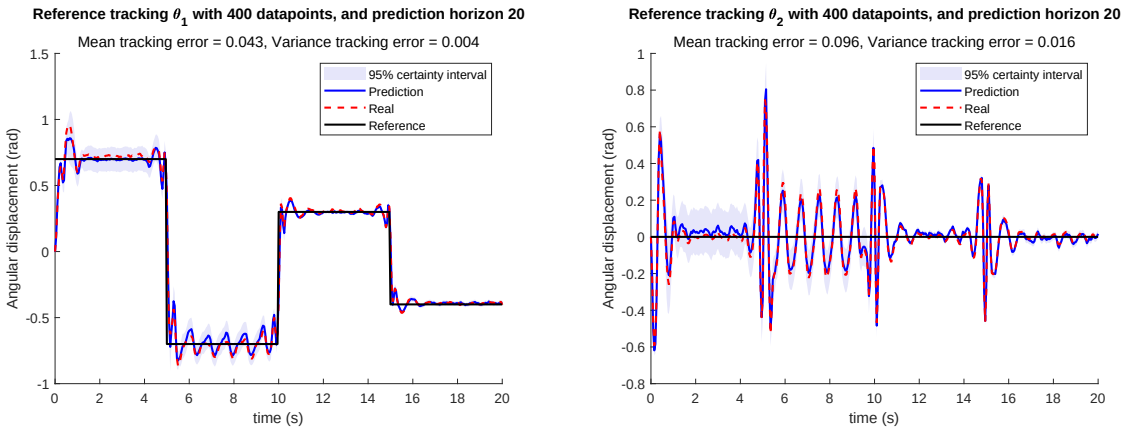


Figure 4-7: Example of the reference tracking controller using MPC for the double pendulum system. The prediction horizon in this example is $N_\mathrm{p} = 20$ with a sparse GP as internal model with 400 inducing points. The control objective of the first beam (left) is to track the reference path of the angle θ_1 in the lower half circle of the operating space. The second beam (right) is subjected to remain in the stable down-down position. The tracking performance is analyzed by the mean tracking error which is depicted in the subtitles. It is concluded that the controller tracks the reference path of θ_1 and dampens the vibrations of θ_2 except for some fluctuations around the reference at $5 \text{ s} \leq t \leq 10$ s. A possible reason for this observation is a bad performing model at this specific operating range.

to $N_\mathrm{p} = 40$. Also, from the figures it is unclear if the number of inducing points influences the control performance significantly since it seems that the lowest tracking errors are obtained for both a low and high number of inducing points. In conclusion, the optimal prediction horizon is $N_\mathrm{p} = 20$ because of the relatively low tracking error.

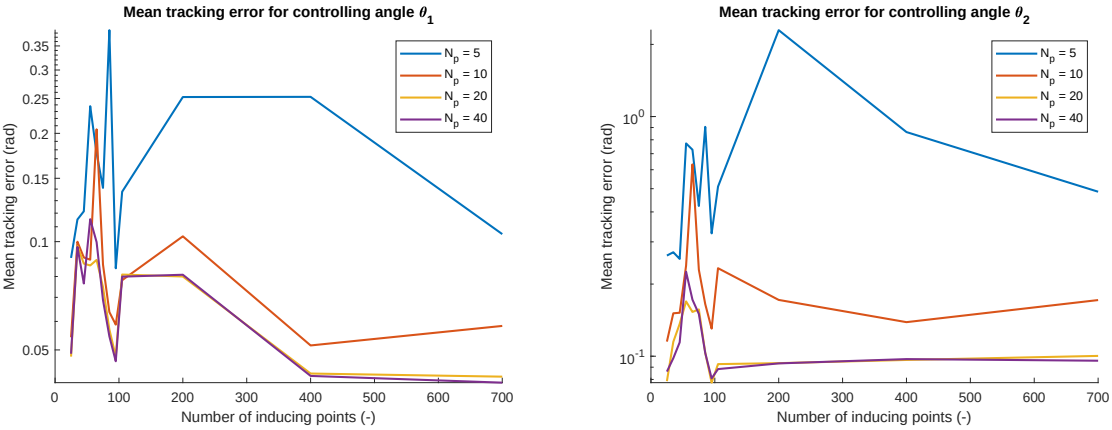


Figure 4-8: Mean tracking error of the model predictive controller using sparse GP models with different amounts of inducing points and different prediction horizons for both angles θ_1 (left) and θ_2 (right). The tracking error indicates that the optimal prediction horizon is $N_\mathrm{p} = 20$ as a lower prediction horizon worsens the tracking performance while a higher prediction horizon does not increase the performance significantly. However, it cannot be concluded if the number of inducing points in the model influences the control performance as the lowest tracking errors are obtained for both low and high numbers of inducing points.

4-3-2 Computation time

This section finishes the simulation part of the research by calculating and analyzing the average calculation time of one iteration of the GP-MPC with respect to the number of inducing points and the prediction horizon. To incorporate the GP-MPC algorithm in a realtime environment it is important to make sure that the controller calculates the control input faster than the sampling time $h = 0.05$ s of the control system. The average calculation time of the controller is constrained by the hardware when writing this thesis, i.e., hardware restrictions.

The results of the average calculation time of the controller with their standard deviation bounds are visualized in Figure 4-9. Referring to the standard deviation bounds, the conclusion is drawn that a prediction horizon of $N_{\mathrm{p}} = 40$ is not suitable for incorporation in the realtime implementation since this requires a calculation time higher than the sampling time. This is also observed for low numbers of inducing points where the standard deviation exceeds the sampling time limit. Next to this, controlling the system with a prediction horizon of $N_{\mathrm{p}} = 20$ shows a necessity to keep the number of inducing points as low as possible due to a fast crossing of the $h = 0.05$ s bound. Nonetheless, the figure shows that the controller obtains a sufficient calculation time for a broad number of inducing points if the controller consists of a prediction horizon of $N_{\mathrm{p}} = 5$ or $N_{\mathrm{p}} = 10$. It is however that controlling the system with a prediction horizon of $N_{\mathrm{p}} = 5$ or $N_{\mathrm{p}} = 10$ deteriorates the performance of the controller as was observed in the previous section. Therefore, it is concluded that the prediction horizon of $N_{\mathrm{p}} = 20$ with a low number of inducing points (as low as possible) contains the best trade-off between calculation time and model performance.

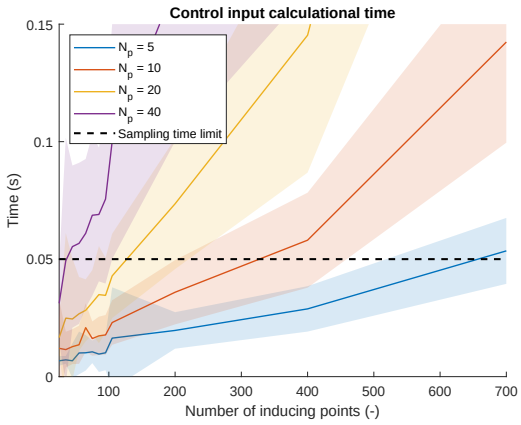


Figure 4-9: Average time with standard deviation bounds for calculating one iteration of the MPC controller using a different prediction horizon and approximate GP models with different amount of inducing points. For sufficient control in realtime, the controller is constrained to calculate the control inputs faster than $0.05s$ which is the sampling time of the control system. The conclusion is drawn that the prediction horizons of $N_p = 5$ and $N_p = 10$ achieve sufficient calculational times ($t_{\mathrm{calc}} < 0.05$) for large set of numbers of inducing points, while the prediction horizon $N_p = 40$ cannot be used for realtime control if taking into account the standard deviation. Furthermore, the prediction horizon $N_p = 20$ is feasible to be used for realtime control for low numbers of inducing points.

4-4 Realtime control using approximate Gaussian process models

The last phase in this research is to implement the GP-MPC algorithm in the physical double pendulum system. This system has been made available by the research laboratory of the Delft Center for Systems and Control (DCSC) at the TU Delft. For testing and validating the GP-MPC algorithm in the realtime environment, a similar approach is followed as in the simulation phase. This approach involves the training and validation of an approximate GP model and testing the performance of the controller in a reference tracking and disturbance rejection case. The findings obtained in the previous sections are incorporated as prior knowledge for implementing the GP-MPC algorithm in the physical setup.

The remaining part of this section discusses the results of the GP-MPC algorithm on the physical double pendulum setup. Section 4-4-1 elaborates on the validation signals used for analyzing the approximate GP model performance. Section 4-4-2 elaborates on the validation of the approximate GPs models. At last, Section 4-4-3 ends the chapter by discussing two control cases which are reference tracking and disturbance rejection.

4-4-1 Validation signals

This section explains the validation signals used for verifying the GP models. The approximate GP models are trained and validated by providing an APRBS signal to the physical setup after which both the outputs θ_1 and θ_2 are collected for constructing the regression vectors. In contrast to other well-known validation signals, the APRBS signal ensures a persistent excitation of the system that is also suitable for nonlinear system identification.

Figure 4-10 visualizes the validation signals used for testing the performance of the approximate GP model. There is observed that the APRBS input signal is bounded between the values -1 and 1 since these are the actuator bounds of the physical system. The output

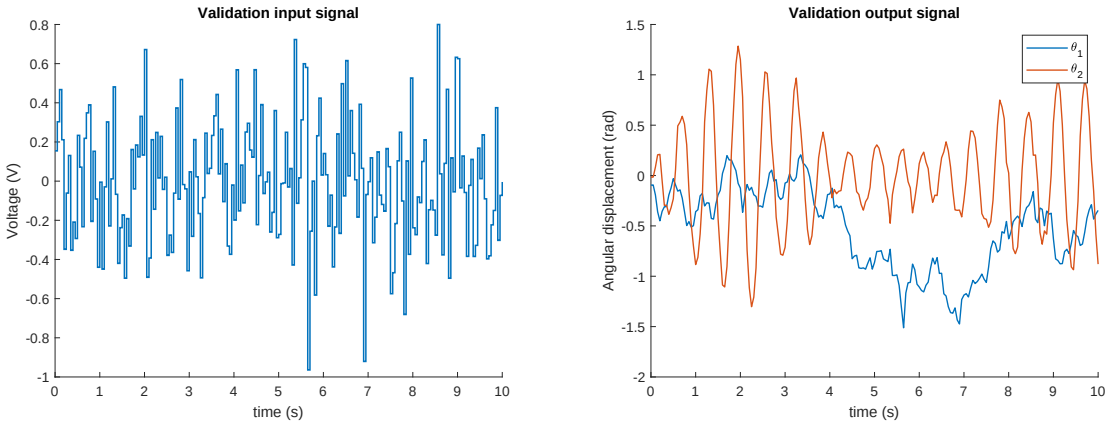


Figure 4-10: Validation signals used for constructing the validation dataset of the physical double pendulum system. The APRBS input signal (left) to the system excites the system persistently for nonlinear system identification. The validation output signal (right) consists of the angles θ_1 and θ_2 which are retrieved from sensor measurements.

signals show a slow rotation pace for the first beam with angle θ_1 while the second beam is rotating at a higher frequency. The reason for this is the fact that the first beam is connected to the DC motor and therefore experiences motor friction while the second beam is free to rotate.

4-4-2 Learning approximate Gaussian process models for realtime control

This section explains the training of the approximate GP models and tests their feasibility for incorporating the models in the GP-MPC framework. The inducing points for training the approximate GP are obtained by making a selection of the regression vectors from the dataset that is used for training an exact GP model. This selection is based on the previously used selection heuristic which selects the regression vectors that provide the most information for the exact GP model.

The number of inducing points is chosen to be as low as possible since it is obtained in Section 4-3 that issues arise regarding the computational time of the GP-MPC controller if too many inducing points are incorporated. Therefore, the approximate GP is trained by using 25 inducing points from an exact dataset of 1000 regression vectors. Moreover, it is previously obtained that a prediction horizon of $N_p = 20$ provides the best trade-off regarding the tracking performance and calculational time of the controller. For that reason, the approximate GP is only validated for the 20-step-ahead prediction case.

Figure 4-11 depicts a 20-step-ahead prediction of the approximate GP models of θ_1 and θ_2. In the figure, both angles are observed to accurately predict the approximate GP model for 20 timesteps ahead in time as a VAF higher than 95% is achieved and MSLL values lower than -0.6. Therefore, it is suitable to be incorporated in the GP-MPC algorithm for controlling the double pendulum system.

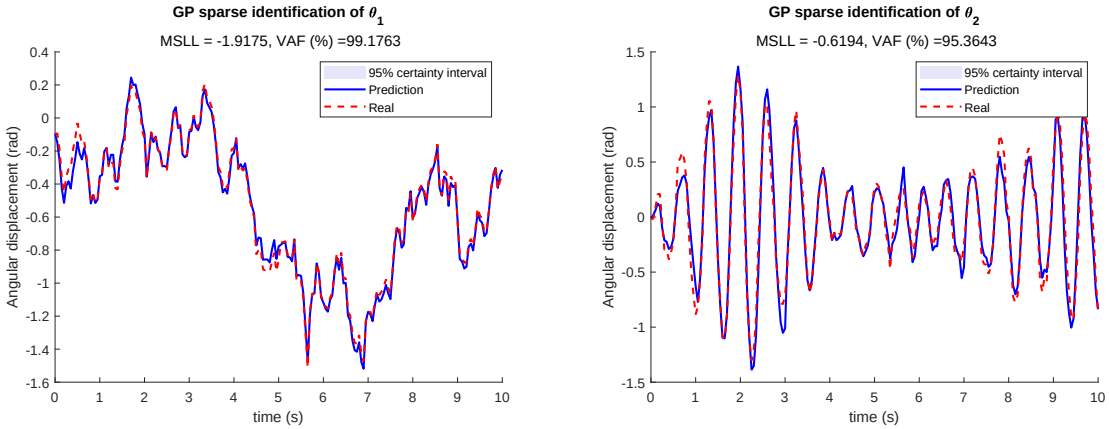


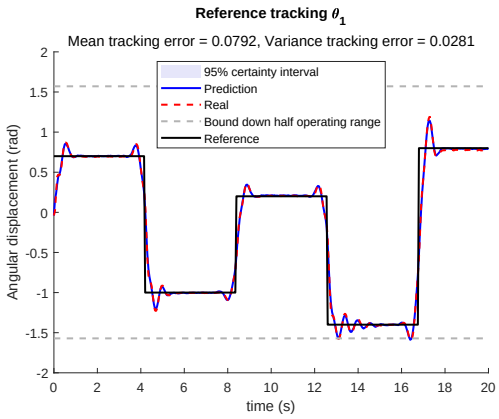
Figure 4-11: Validation of the 20-step-ahead prediction of the physical double pendulum model using the GP-NARX method with the SE kernel function for models θ_1 (left) and θ_2 (right). The model is trained using the PEP sparse GP algorithm with 25 inducing points and validated with VAF and MSLL performance measures. The models contain a high performance due to a VAF value of around 95% and negative MSLL values. So, the approximate GP models are feasible for incorporation in the GP-MPC algorithm.

4-4-3 Realtime control of the double pendulum system

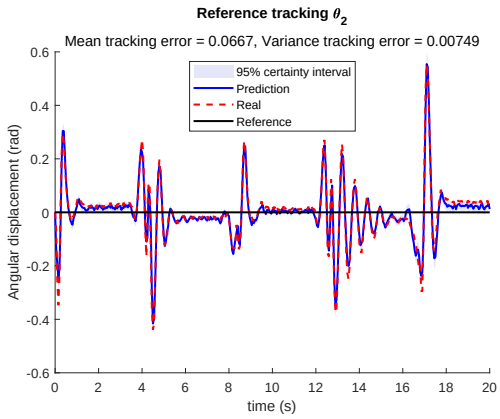
After obtaining an accurate approximate GP model it is tested in the GP-MPC algorithm on the physical double pendulum system. In this section, two control cases are considered which are reference tracking and disturbance rejection. For both control cases, a prediction horizon of $N_p = 20$ is used for running the MPC controller. In the remainder of this section, both control cases are discussed.

Reference tracking

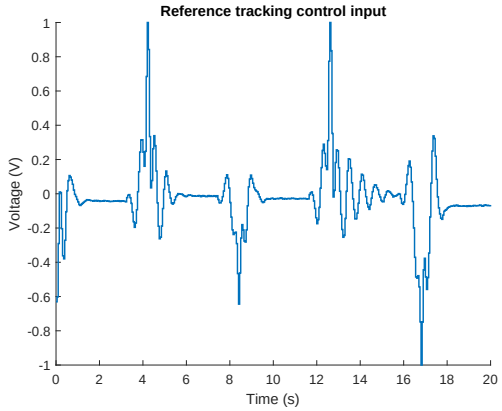
The first control case that is tested on the physical pendulum system is reference tracking control. For this purpose, a reference trajectory is applied to the controller which is a step signal with varying amplitude for θ_1 and the stable down position for θ_2 ($\theta_2 = 0$). The results of the reference tracking controller with the corresponding control input are visualized in Figure 4-12. As observed in Figure 4-12, the controller is able to track the reference trajectory with a mean tracking error of $\mu_{\text{ref}} \approx 0.08$. By controlling the system, angle θ_1 experiences overshoot when settling to the reference trajectory which is probably caused by the controller trying to damp angle θ_2. This effect might be reduced by tuning the MPC weighting terms of the controller. Also, the double pendulum system experiences some vibrations when settling to the reference around $12.5 \leq T \leq 16.5$. This behaviour could be explained by the fact that it might be harder for the control system to damp vibrations when θ_1 reaching its horizontal position, i.e., $\theta_1 = -0.5\pi$ or $\theta_1 = 0.5\pi$.



(a) Reference tracking θ_1



(b) Reference tracking θ_2

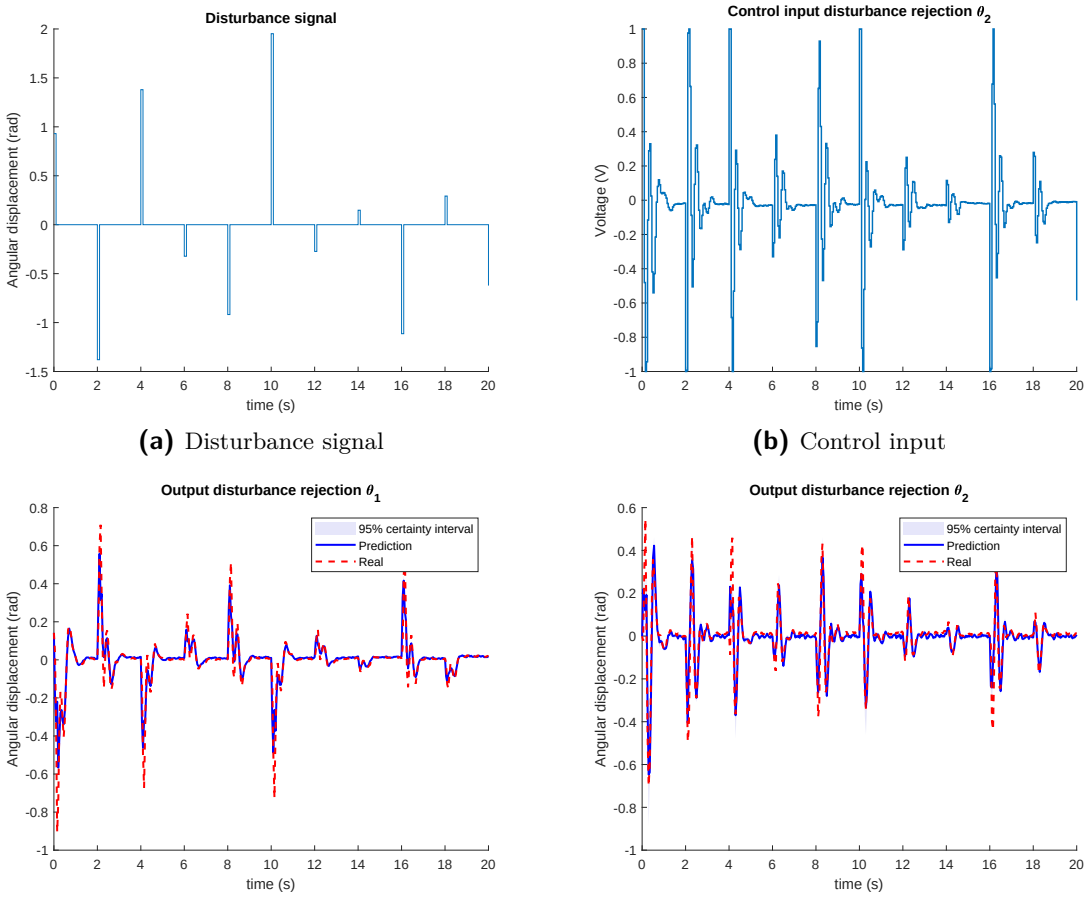


(c) Control input

Figure 4-12: Reference tracking of angles θ_1 (**a**) and θ_2 (**b**), and the control input (**c**) of the physical double pendulum system . Angle θ_1 tracks a reference trajectory (black) while θ_2 is kept in its equilibrium (down) position. It is concluded that the GP-MPC controller tracks the references properly with a mean tracking error under 0.08. However, angle θ_1 experiences some overshoot when settling, and both angles show vibrations when approaching the horizontal position of θ_1 ($\theta_1 \approx -\pi$). The overshoot might be reduced by retuning the controller. The vibrations are possible due to the fact that it is difficult for the control system to damp vibrations in the horizontal beam position.

Disturbance rejection

The last control case in this thesis is disturbance rejection. The disturbance is generated by adding a pulse signal with varying amplitude to the output θ_2 of the double pendulum system. This pulse signal simulates a tap disturbance that forces the system to leave its equilibrium position. The disturbing pulse signal, the control input and the disturbance rejection for models θ_1 and θ_2 are visualized in Figure 4-13. It is observed that both beams are stabilized within a second in their down position after being disturbed by the pulse signal. This means that the controller is robust to potential disturbances applied to the system.



(a) Disturbance signal

(b) Control input

(c) Disturbance rejection θ_1

(d) Disturbance rejection θ_2

Figure 4-13: The disturbance signal (**a**), control input (**b**), and outputs θ_1 (**c**) and θ_2 (**d**) of the double pendulum system in a disturbance rejection control case. The disturbance is added to the angle θ_2 of the systems which represents a disturbing tap on the second beam that forces the system out of the equilibrium position. From the control input and system output signals it is seen that the disturbance is directly counteracted on by controlling the system to its stable down position. So, the GP-MPC controller is robust to disturbances.

Chapter 5

Conclusions and Recommendations

5-1 Conclusion

This thesis investigates Gaussian processes (GPs) with the aim of learning dynamic systems in a purely data-driven fashion which are also suitable for the application in a model predictive control (MPC) framework for controlling a real-world setup. In the current literature, GP-MPC is studied more extensively as a supporting element to conventional modelling approaches. In cases where fully data-driven approaches are used for GP-MPC, the research mainly considers implementations in simulation environments. However, for incorporating GP-MPC in physical applications it is necessary to investigate fully data-driven GP-MPC approaches in a realtime environment, which is a largely unexplored field of research in the current literature. Using GPs for learning dynamic and realtime systems is promising due to their potential data efficiency and ability to model nonlinear systems. Therefore, the goal of this thesis is formulated as follows:

Research goal: Develop an integrated realtime control application and fully data-driven solution for the (nonlinear) dynamic modelling and predictive control by employing GPs for a double pendulum system.

The contributions of this thesis aim to show the modelling power of GPs in a realtime environment and to encourage the development of fully data-driven control methods which might extend or improve future technologies. To fulfill the thesis goal, Gaussian process regression theory is used for the fully data-driven modelling of nonlinear dynamical systems to be controlled in realtime using MPC. For this purpose, a kernel function and model structure is identified using exact GPs for capturing the dynamics of the double pendulum system in the open-loop stable operating area. However, exact GPs for realtime MPC requires a computational time higher than the sampling time of the realtime system. Therefore, approximate GPs for offline system identification are employed that use the power expectation propagation (PEP) sparsification method for reducing the calculation time of predictions, while

also preserving a high performance of the GP models. Lastly, the offline approximate GP models were used in a realtime GP-MPC algorithm to control a double pendulum system. As a result, the GP-MPC algorithm was able to make fast and accurate predictions of the dynamical behaviour of the system which ensured the performance of the realtime controller. This GP-MPC algorithm is developed on the basis of multiple subquestions that were stated in the introduction and concluded in the remaining part of this section.

For defining the dynamical GP models it is necessary to specify a kernel function and a dynamical model structure. The kernel function predefines the behaviour of the GP model while the model structure determines the physical interpretation of the system. Defining the kernel function and model structure enables the GP dynamic model to describe the system dynamics properly. Therefore, it is investigated what combination from a given set of model structures and kernel functions is most suitable for describing the dynamics of the double pendulum in an MPC framework. Therefore, the first subquestion was formulated as follows:

Subquestion I: Which combination of model structure and kernel function is suitable for capturing the dynamics of a double pendulum? And what are the limitations and possibilities of the GP dynamical models to be incorporated in a model predictive control framework?

For the purpose of exploring kernel performance, three kernel functions were investigated including the squared exponential (SE) kernel, periodic (P) kernel, and composite (P times SE) kernel. All kernel functions are subject to three model structures, namely, the nonlinear autoregressive model with exogenous input (NARX), the nonlinear output error (NOE) model, and the augmented nonlinear state space (NSS) model. The combinations of model structures and kernel functions are validated by calculating the performance measures variance accounted for (VAF) and the mean standardised log loss (MSLL). For answering subquestion I, the VAF and MSLL of the one- and multiple-step-ahead prediction are taken into account since the one-step-ahead prediction validates the overall model performance and the multiple-step-ahead validates the performance of the model for the MPC application.

After an extensive analysis of all combinations of kernels and model structures, it is concluded that the GP-NARX and augmented GP-NSS model structures both are able to provide accurate one-step-ahead prediction performance for all investigated kernel functions. However, the performance of the augmented GP-NSS model structure deteriorates significantly in performance for multi-step-ahead prediction scenarios. A possible reason for this observation is that the GP-NSS model suffers from a fast error accumulation in the predictive regression vector which undermines its modelling accuracy. Contrarily, for every combination of kernel function, it is observed that the GP-NARX method achieves a sufficient performance when predicting multiple timesteps ahead in time. It is shown that the performance of the GP-NARX method for each investigated kernel function is similar in a multiple-step-ahead prediction. At last, the GP-NOE model structure seems not to be able to make accurate model predictions for both one- and multiple-step-ahead predictions regarding all combinations of kernel functions. This might be a result of this method being prone to predicate upon the predicted model values coming from models that are non- or suboptimal solutions of the marginal log likelihood (MLL) optimization. Retraining GP-NOE models containing a dataset with non- or suboptimal solutions increase the risk of errors entering the model.

By considering all the results, the GP-NARX method seems to be most suitable to describe the dynamics of the double pendulum system with equal performance for all investigated kernel functions. It is therefore that the SE kernel is selected to be the most appropriate kernel function for modelling the dynamics of the double pendulum system since this kernel function is the simplest model representation among the investigated functions.

Until now, this thesis has only considered exact GPs for learning the dynamics of the double pendulum. However, datasets for exact GPs might rapidly increase in size because of the availability of big data. Exact GPs are computationally inefficient for large-scale datasets due to the computation of the matrix inverse of vast kernel matrices. Therefore, a second subquestion was formulated for approximating the exact GP:

Subquestion II: How is an approximate GP used for accelerating the involved calculations for constructing dynamical models? And is the performance of the model affected by the approximation?

For answering this question, the PEP sparsification method was implemented which was obtained as the most promising in the literature. The PEP method hybridly connects the prior and posterior GP approximation methods by using a scaling constant. This constant was kept at a constant value of $\alpha = 0.5$ in this research. Also, the inducing points for training the approximate GPs were found to be a subset of the training dataset of the exact GP. The subset included the training points containing the most information for the exact GP as inducing points.

The research on approximate GPs has been performed on the chosen GP-NARX model structure with SE kernel function. The resulting approximate GP models were validated on performance using the performance measures with respect to the number of inducing points and both one- and multiple-step-ahead predictions.

From the results, it is found that the approximate GP models achieve higher performance when increasing the number of inducing points. Next to this, extending the prediction capabilities to multiple-step-ahead predictions deteriorates the performance of the approximate GP models. However, it is observed that almost all the approximate GP models show high performance (VAF $>$ 85% and low MSLL) with respect to all numbers of inducing points, and all investigated multi-step-ahead predictions. Only some models that were trained with a low number of inducing input points had mediocre performance (70% $<$ VAF $<$ 85% and relatively low MSLL) if the models were used in a 40-step-ahead prediction.

The main goal of this thesis is the implementation of the GP-MPC algorithm for realtime control. Implementing the GP-MPC in a realtime environment requires a faster-than-realtime calculation of the control input. Therefore, the next subquestion considers the investigation of the performance and computational time of the MPC controller in relation to the controller prediction horizon and the number of inducing points of the approximate GP model:

Subquestion III: How is an approximate GP employed in an MPC framework, such that it is capable of realtime control of a double pendulum system taking into account the computational time and effectivity of the GP-MPC algorithm?

For answering this question, a multivariable online MPC controller is designed that calculates the control input based on offline identified approximate GP models. The controller is designed for multivariable reference tracking, where the first beam should follow a varying reference signal, and the second beam should remain vertical, in its equilibrium down orientation. Also, actuator constraints are imposed as constraints to the predictive controller to keep the control input within actuator bounds. After designing the controller, the number of inducing inputs of the approximate GP model and the length of the controller prediction horizon valid are investigated to effectuate realtime control.

By analyzing the mean tracking error of the controller, the conclusion is drawn that the optimal prediction horizon for the MPC (among the investigated) is $N_\mathrm{p} = 20$. This observation results from the fact that a lower prediction horizon decreases the control performance (higher tracking error) while a higher prediction horizon does not increase the control performance significantly. However, from the tracking error, it cannot be concluded if a higher number of inducing points result in better control performance since the lowest mean tracking errors are obtained for both low and high numbers of inducing points.

Next to considering the tracking error of the GP-MPC controller, the controller should also be tested on computation time for realtime performance. The realtime controller is single-shooting and therefore constrained to calculate the control inputs faster than the sampling time of $h = 0.05$. The results show that the controller cannot satisfy this constraint if the system is controlled with a prediction horizon of $N_\mathrm{p} = 40$. This large prediction horizon always violates the sampling time constraint when taking into account the variance of the mean calculational time. Contrarily, the prediction horizons of $N_\mathrm{p} = 5$ and $N_\mathrm{p} = 10$ are able to produce a faster-than-realtime computation time for a broad range of number of inducing points. Also, the prediction horizon of $N_\mathrm{p} = 20$ is able to control the system with sufficient sampling times for low numbers of inducing points.

Combining both the results of the tracking error and the computation time, it is shown that a prediction horizon of $N_\mathrm{p} = 20$ with a low number of inducing points results in the optimal setting for implementing the controller in realtime. This setting ensures an acceptable control performance while also being able to calculate the control inputs at a sufficient rate.

Until now, the GP-MPC algorithm has only been tested in a simulation environment. For completing this research, the last subquestion involves the algorithm being implemented in the physical double pendulum setup which is as follows:

Subquestion IV: How is the resulting GP-MPC algorithm implemented in the physical double pendulum system? And what performance can be observed?

For answering this question, a physical laboratory-scale double pendulum system has been employed. For controlling the double pendulum system, the effectuation of the GP-MPC algorithm consists of two phases, i.e., an offline identification and an online control phase. The offline phase covers the identification of a nonlinear dynamical model using approximate GP-NARX models after which the models are used in the model predictive controller in the online phase.

For the offline phase of the algorithm, a dataset is collected by applying a sufficiently exciting nonlinear identification signal to the system and measuring the angular displacement. Eventually, two approximate GP models have been trained with both 25 inducing points as it was found in Subquestion III to keep the inducing points to a minimum. It is shown that constructing the model with 25 inducing points leads to accurate predictions of the physical double pendulum systems, even for predicting 20 timesteps ahead in time which was received to be the optimal prediction horizon in Subquestion III. Therefore, the approximate GP models are sufficient to be used in the realtime model predictive controller.

The online phase of the algorithm uses the controller of Subquestion 5-1 for controlling both beams. It is seen that the controller is able to sufficiently track the reference trajectory. However, the controller seems to experience more difficulties when reaching its horizontal position which can be explained by the fact that it is harder to control the double pendulum when the beam is horizontally oriented. Also, the controller is tested for its disturbance rejection. For this purpose, a pulse signal is applied to the angular displacement of the second beam θ_2. The controller shows robust performance to the disturbance since it is directly counteracted by the controller.

5-2 Recommendations for future research

This thesis presented a GP-MPC algorithm for the realtime control of a double pendulum system which is capable to stabilize and control the system in the lower half circle of the operating range. However, the proposed algorithm is still a prove of concept, that can be improved in future research. Therefore, this section discusses the identified recommendations for the current study and future research.

Recommendations on the current study

1. Incorporate uncertainty propagation for multiple-step-ahead predictions. The multiple-step-ahead predictions of the GP model are now conservative due to the use of the zero-variance method for the uncertainty propagation. By using the zero-variance method, the uncertainty in a multiple-step-ahead prediction is the result of unobserved data points in training which makes it independent of the uncertainty of previously obtained predictions. However, this previously induced uncertainty should be propagated over the prediction horizon for obtaining realistic uncertainty measures. Including uncertainty propagation in the algorithm would increase the accuracy of the uncertainty in a multi-step-ahead prediction of the GP.

2. Incorporate the uncertainty measure of GPs in the MPC. Yet, the uncertainty is not incorporated in the constraints and objective function of the MPC framework. The implementation ignores therefore an advantage of GPs which is their measure of uncertainty of the prediction. Incorporating the uncertainty in the MPC framework improves the robustness of the controller. The robustness is either obtained by penalizing uncertain predictions in the objective function and/or by incorporating chance constraints that reassure the system to remain within operating bounds at a certain probability.

3. Incorporate realistic noise assumptions in the GP model. Now, the proposed model structure (NARX) suffers from unrealistic noise assumptions. To incorporate the noise in the model, several tools are available, such as the noisy input gaussian process (NIGP). The accuracy of the GP models is likely to increase when realistic noise assumptions are made.

4. Test the algorithm on more sophisticated control objectives. The control objective in this thesis is to stabilize and control the double pendulum system in the lower half circle of the operating range. Also, the second beam of the double pendulum system is controlled to remain in its stable equilibrium position. However, it would be more challenging for the algorithm to stabilize the system in a non-stable equilibrium position which would involve closed-loop identification of the system.

Recommendations for future research

1. Research the possibilities for realtime GP-MPC to be used in an online environment. Now, the proposed GP-MPC algorithm uses an offline identified GP model for the prediction of future states without updating the model online. However, the algorithm could be more adaptive if the model receives online updates using recently acquired data.

2. Research the possibilities to do online negative marginal log likelihood (NMLL) optimization for identifying the GP model which is now performed offline. However, for direct implementation of the GP-MPC algorithm without actively monitoring the system identification process, research can be employed for performing online hyperparameter optimization. This involves solving a nonlinear and non-convex optimization problem time-to-time and taking safety measures if the resulting GP model fails to identify the system properly.

3. Research the possibilities of acquiring data using active learning (also called optimal experiment design) for identifying dynamical models with GPs. This might improve the quality of the dataset since active learning can be designed to maximize the learning rate of the GP. Active learning might also be useful if the GP is used for online identification.

4. Increase the computational speed of the GP-MPC algorithm. Currently, the MPC cannot deal with big datasets for realtime control since it is computationally demanding to solve the optimization problem at each iteration of the control system. However, the proposed GP-MPC algorithm might need larger datasets for controlling the double pendulum system in its entire operating range. Also, the speed of the controller deteriorates if the uncertainty of the GP model is incorporated which might result in an infeasible calculational speed of the controller. A possible solution for this is proposed in [86], where a fast GP-based model predictive control framework is proposed for decreasing the computational time of the GP-MPC while also incorporating uncertainty propagation. Research should be employed if this method provides advantages in a realtime environment.

5. Obtain a controller that also takes into account environmental effects, control efficiency or other control objectives by the use of economic MPC for GPs. Economic MPC has

gain more interest in recent years due to its ability to punish for specific economic objective criteria while also imposing constraints [87].

6. Change the MPC controller to a robust MPC controller for increased robust performance. Robust MPC deals with disturbances or uncertainties in the system which might influence the performance of the control system [88]. This controller can possibly be beneficial for several control objectives.

Appendix A

Gaussian Process derivations

A-1 Calculation of the posterior distribution of a prediction

The posterior that is used in the prediction problem is visualized as follows [26]:

$$p\left(f_* \mid \mathcal{D}, \boldsymbol{\Sigma}, \mathbf{x}_*\right) = \frac{p\left(\left[\mathbf{y}^T, f_*\right]^T \mid \boldsymbol{\Sigma}, \mathbf{x}, \mathbf{x}_*\right)}{p\left(\mathbf{y} \mid \boldsymbol{\Sigma}, \mathbf{x}\right)}. \tag{A-1}$$

The math involved to find the posterior starts by observing that the unknown function variables follow a Gaussian distribution. The definition of a Gaussian probability density function $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a random vector $\mathbf{x}$ is as follows:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\boldsymbol{K}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})}, \tag{A-2}$$

where $\boldsymbol{\mu}$ is the expectation/mean $E(\mathbf{x})$ of the random variable, and $\boldsymbol{K}$ the variance matrix $var(\mathbf{x})$.

The numerator and denominator of Equation (A-1) are rewritten by making use of Equation (A-2) to obtain the posterior. This results in the following equations:

$$p\left(\left[\mathbf{y}^T, y_*\right]^T \mid \mathcal{D}, \boldsymbol{\Sigma}, \mathbf{x}_*\right) = \frac{1}{(2\pi)^{\frac{N+1}{2}} |\boldsymbol{\Sigma}_{N+1}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left([\mathbf{y}^T, y_*]\boldsymbol{\Sigma}_{N+1}^{-1}[\mathbf{y}^T, y_*]^T\right)}, \tag{A-3}$$

$$p\left(\mathbf{y} \mid \boldsymbol{\Sigma}, \mathbf{x}\right) = \frac{1}{(2\pi)^{\frac{N}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\mathbf{y}\boldsymbol{\Sigma}^{-1}\mathbf{y}^T\right)}, \tag{A-4}$$

where $\boldsymbol{\Sigma}_{N+1} \in \mathbb{R}^{N+1 \times N+1}$ is defined as the kernel function that incorporates the covariance of the prediction $\boldsymbol{\Sigma}_{**}$ together with the cross-covariance between the prediction and the model $\boldsymbol{\Sigma}_*$. The matrix is visualized as follows:

$$\mathbf{\Sigma}_{N+1} = \begin{bmatrix} \mathbf{\Sigma} & \mathbf{\Sigma}_* \\ \mathbf{\Sigma}_*^T & \mathbf{\Sigma}_{**} \end{bmatrix}. \tag{A-5}$$

After defining the probability density functions in Equations (A-3) and (A-4), the posterior is calculated by filling in Equation (2-16) as follows:

$$p\left(f_* \mid \mathcal{D}, \mathbf{\Sigma}, \mathbf{x}_*\right) = \frac{|\mathbf{\Sigma}|^{\frac{1}{2}}}{(2\pi)^{\frac{1}{2}} \left|\mathbf{\Sigma}_{N+1}\right|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\left[\mathbf{y}^T, f_*\right]\mathbf{\Sigma}_{N+1}^{-1}\left[\mathbf{y}^T, f_*\right]^T - \mathbf{y}^T\mathbf{\Sigma}^{-1}\mathbf{y}\right)}. \tag{A-6}$$

In Equation (A-6) there is observed that an inverse should be calculated of the covariance matrix $\mathbf{\Sigma}_{N+1}$. One theory that is able to calculate the inverse of block matrices is the Schur complement [89]. In this case, the Schur complement is the following:

$$\begin{aligned}
\mathbf{\Sigma}_{N+1}^{-1} &= \begin{bmatrix} \mathbf{\Sigma} & \mathbf{\Sigma}_* \\ \mathbf{\Sigma}_*^{\mathrm{T}} & \mathbf{\Sigma}_{**} \end{bmatrix}^{-1} \\
&= \left(\begin{bmatrix} I & 0 \\ \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1} & I \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma} & 0 \\ 0 & \mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_* \end{bmatrix} \begin{bmatrix} I & \mathbf{\Sigma}^{-1}\mathbf{\Sigma}_* \\ 0 & I \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} I & -\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_* \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}^{-1} & 0 \\ 0 & \left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -\mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1} & I \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{\Sigma}^{-1} + \mathbf{\Sigma}^{-1}\mathbf{\Sigma}_* \left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1} & -\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_* \left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} \\ -\left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1} & \left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} \end{bmatrix}
\end{aligned} \tag{A-7}$$

After calculating the inverse of the block matrix, one can write out the exponent of Equation (A-6). This is done as follows:

$$\begin{aligned}
&-\frac{1}{2}\left(\left[\mathbf{y}^{\mathrm{T}}, f_*\right]\mathbf{\Sigma}_{N+1}^{-1}\left[\mathbf{y}^{\mathrm{T}}, f_*\right]^{\mathrm{T}} - \mathbf{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{y}\right) = \\
&-\frac{1}{2}\Big(\boldsymbol{y}^{\mathrm{T}}\left(\mathbf{\Sigma}^{-1} + \mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1}\mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\right)\boldsymbol{y} - \\
&\boldsymbol{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} f_* - f_*\left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1}\mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{y} + \\
&f_*\left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)^{-1} f_* - \mathbf{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{y}\Big)
\end{aligned} \tag{A-8}$$

Introducing the variable $\sigma_{f_*}^2 = \left(\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right)$, the problem can be rewritten as:

$$\begin{aligned}
&-\frac{1}{2}\Big(\boldsymbol{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\left(\sigma_{f_*}^2\right)^{-1}\mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\left(\sigma_{f_*}^2\right)^{-1} f_* - f_*\left(\sigma_{f_*}^2\right)^{-1}\mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{y} + f_*\left(\sigma_{f_*}^2\right)^{-1} f_*\Big) \\
&= -\frac{1}{2}\left(f_* - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{y}\right)^{\mathrm{T}}\left(\sigma_{f_*}^2\right)^{-1}\left(f_* - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{y}\right)
\end{aligned} \tag{A-9}$$

The solution is recognized when looking at the standard definition of a Gaussian probability density function of Equation (A-2). The mean of the posterior distribution is found within the brackets as $\mu_{f_*} = \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{y}$, and the variance as $\sigma_{f_*}^2 = \mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*$ which is the same result as in Equation (2-19). The variance could have also been obtained by looking at the scaling of Equation (A-6). As there is assumed that the block matrix $\mathbf{\Sigma}_{N+1}$ is invertible due to the feasibility of the Gaussian Process, one can rewrite the scaling term as:

$$\frac{|\mathbf{\Sigma}|^{\frac{1}{2}}}{(2\pi)^{\frac{1}{2}}\,|\mathbf{\Sigma}_{N+1}|^{\frac{1}{2}}} = \frac{|\mathbf{\Sigma}|^{\frac{1}{2}}}{(2\pi)^{\frac{1}{2}}\left(|\mathbf{\Sigma}|\left|\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right|\right)^{\frac{1}{2}}} = \frac{1}{(2\pi)^{\frac{1}{2}}\left|\mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*\right|^{\frac{1}{2}}}, \tag{A-10}$$

which also retrieves that the variance is defined as $\sigma_{f_*}^2 = \mathbf{\Sigma}_{**} - \mathbf{\Sigma}_*^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{\Sigma}_*$.

A-2 Numerical stable implementation of the log likelihood function

The marginal log likelihood (MLL) function of Equation (2-14) is in its normal form a numerical unstable objective function due to the calculation of the inverse. This fact is widely known. Therefore, the matrix inverse and determinant in the MLL function are rewritten. It is also obtained that the objective function is cheaper to evaluate in terms of computational complexity. The MLL of Equation (2-14) is restated as follows:

$$\ell(\boldsymbol{\theta}) = \ln p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) = -\overbrace{\frac{1}{2}\ln\left(|\mathbf{\Sigma}|\right)}^{\text{complexity term}} - \overbrace{\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{y}}^{\text{data-fit term}} - \overbrace{\frac{N}{2}\ln(2\pi)}^{\text{normalisation const.}}. \tag{A-11}$$

The numerical stable implementation is achieved by introducing a mathematical tool of linear algebra which is called the Cholesky decomposition of a matrix. The Cholesky decomposition creates an upper triangular matrix which can be interpreted as the square root operator for matrices. The Cholesky decomposition of the kernel matrix Σ is the following:

$$\text{chol}(\mathbf{\Sigma}) = \boldsymbol{L} \rightarrow \mathbf{\Sigma} = \boldsymbol{L}^T\boldsymbol{L}, \tag{A-12}$$

with L an upper triangular matrix. This decomposition is used in order to calculate the inverse as follows:

$$\mathbf{\Sigma}\alpha = \mathbf{y} \rightarrow \boldsymbol{L}^T\boldsymbol{L}\alpha = \mathbf{y} \rightarrow \alpha = (\boldsymbol{L})^{-1}\left(\boldsymbol{L}^T\right)^{-1}\mathbf{y}. \tag{A-13}$$

The logarithm of the determinant of the Cholesky decomposition is calculated as follows:

$$\begin{aligned}
\log\left(\det(\boldsymbol{\Sigma})\right) &= \log\left(\det\left(\boldsymbol{L}\boldsymbol{L}^T\right)\right) \\
&= \log\left(\det(\boldsymbol{L})\det\left(\boldsymbol{L}^T\right)\right) \\
&= \log\left(\det\left(\boldsymbol{L}\right)^2\right) && \triangleright \textit{ Using property: } \det\left(\boldsymbol{L}^T\right) = \det\left(\boldsymbol{L}\right) \\
&= 2\log\left(\prod_{i=1}^{n} L_{ii}\right) && \triangleright \textit{ Triangular matrix determinant is the product of all diagonal entries} \\
&= 2\sum_{i=1}^{n}\log\left(L_{ii}\right) && \text{(A-14)}
\end{aligned}$$

The simplified MLL is obtained by filling in Equations (A-13) and (A-14) in Equation (A-15) as follows:

$$\ell(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\ln\left(L_{ii}\right) - \frac{1}{2}\mathbf{y}^{\mathrm{T}}\alpha - \frac{N}{2}\ln(2\pi). \tag{A-15}$$

Also the gradient of the MLL of Equation (2-15) can be simplified by the derived equations. This is done as follows:

$$\begin{aligned}
\nabla(\ell(\boldsymbol{\theta})) &= -\frac{1}{2}\operatorname{trace}\left(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{y}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\theta_i}\boldsymbol{\Sigma}^{-1}\mathbf{y} \\
&= -\frac{1}{2}\operatorname{trace}\left(\mathbf{L}^{-1}\left(\mathbf{L}^T\right)^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\theta_i}\right) + \frac{1}{2}\alpha^T\frac{\partial\boldsymbol{\Sigma}}{\partial\theta_i}\alpha.
\end{aligned} \tag{A-16}$$

A-3 Numerically stable implementation PEP algorithm

The MLL of the power expectation propagation (PEP) sparse method in Equation (2-48) also suffers from numerical instabilities due to the availability of a matrix inverse. Therefore, the MLL should also be rewritten in a stable form where the involved calculations are explained in this section. The MLL of Equation (2-48) is restated as follows:

$$\begin{aligned}
\mathcal{L}_{PEP} = &-\frac{n}{2}\log 2\pi - \frac{1}{2}\log\mid \alpha K_{FITC} + \boldsymbol{Q}_{\boldsymbol{ff}} + \sigma_\epsilon^2\boldsymbol{I}\mid \\
&-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\left(\alpha K_{FITC} + \boldsymbol{Q}_{\boldsymbol{ff}} + \sigma_\epsilon^2\boldsymbol{I}\right)^{-1}\boldsymbol{y} - \frac{1-\alpha}{2\alpha}\operatorname{tr}\left[\log\left(I + \frac{\alpha}{\sigma_\epsilon^2}\left(\boldsymbol{K}_{\boldsymbol{ff}} - \boldsymbol{Q}_{\boldsymbol{ff}}\right)\right)\right]
\end{aligned} \tag{A-17}$$

With the following matrices:

$$\begin{aligned}
K_{FITC} &= \operatorname{diag}\left[K_{\boldsymbol{f},\boldsymbol{f}} - Q_{\boldsymbol{f},\boldsymbol{f}}\right] \\
Q_{\boldsymbol{f},\boldsymbol{f}} &= K_{\boldsymbol{f},u}K_{u,u}^{-1}K_{u,f}
\end{aligned} \tag{A-18}$$

In order to rewrite the MLL it is observed firstly that there are two diagonal matrix terms available in Equation (A-17). These terms are rewritten to one diagonal matrix in order to simplify notations:

$$K_d = \alpha \operatorname{diag}\left[K_{\boldsymbol{f},\boldsymbol{f}} - Q_{\boldsymbol{f},\boldsymbol{f}}\right] + \sigma_\epsilon^2 \boldsymbol{I} \tag{A-19}$$

Then, the inverse in the data-fit term is rewritten to the Woodbury Identity matrix [83]:

$$\left(K_d + K_{\boldsymbol{f},u} K_{u,u}^{-1} K_{u,f}\right)^{-1} = K_d^{-1} - K_d^{-1} K_{\boldsymbol{f},u} \left(K_{u,u} + K_{u,f} K_d^{-1} K_{\boldsymbol{f},u}\right)^{-1} K_{u,f} K_d^{-1} \tag{A-20}$$

In order to obtain a numerically stable solution, the inverses in the Woodbury Identity matrix are rewritten to Cholesky decompositions. This calculation step is the same as the one used in the numerical stable representation of the exact MLL of Equation (A-15). However, in this case, there are more steps involved. The first matrix that is rewritten to a Cholesky decomposition is the kernel matrix of the inducing inputs. This decomposition is represented as follows:

$$\operatorname{chol}(\boldsymbol{K}_{u,u}) = \boldsymbol{L}_{u,u} \tag{A-21}$$

Now that this Cholesky decomposition is defined, it is used in the inverse in the Woodbury matrix. But before doing this there is another variable introduced for notational convenience. This variable uses also the Cholesky decomposition of Equation (A-21) as follows:

$$\boldsymbol{L}_{u,u}^T \boldsymbol{A} = K_{u,f} K_d^{-0.5} \rightarrow \boldsymbol{A} = \left(\boldsymbol{L}_{u,u}^T\right)^{-1} K_{u,f} K_d^{-0.5} \tag{A-22}$$

Note that an inverse square root operation is applied to the diagonal matrix K_d. This is applied to every single diagonal element in the matrix. After introducing the new variable $\boldsymbol{A}$ the term in the inverse in the Woodbury identity is rewritten as follows:

$$\begin{aligned} K_{u,u} + K_{u,f} K_d^{-1} K_{\boldsymbol{f},u} &= \boldsymbol{L}_{u,u}^T \boldsymbol{L}_{u,u} + \boldsymbol{L}_{u,u}^T \boldsymbol{A}\boldsymbol{A}^T \boldsymbol{L}_{u,u} \\ &= \boldsymbol{L}_{u,u}^T \left(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^T\right) \boldsymbol{L}_{u,u} \\ &= \boldsymbol{L}_{u,u}^T \boldsymbol{B} \boldsymbol{L}_{u,u} && \triangleright \textit{ Using: } \boldsymbol{B} = \boldsymbol{I} + \boldsymbol{A}\boldsymbol{A}^T \\ &= \boldsymbol{L}_{u,u}^T \boldsymbol{L}_B^T \boldsymbol{L}_B \boldsymbol{L}_{u,u} && \triangleright \textit{ Using: } \operatorname{chol}(\boldsymbol{B}) = \boldsymbol{L}_B \end{aligned} \tag{A-23}$$

In order to use this result in the Woodbury matrix identity, it is useful to introduce another variable for notational convenience:

$$\boldsymbol{L}_{u,u}^T \boldsymbol{L}_B^T \boldsymbol{C}^T = K_{u,f} K_d^{-1} \tag{A-24}$$

$$\boldsymbol{C}^T = \left(\boldsymbol{L}_B^T\right)^{-1} \left(\boldsymbol{L}_{u,u}^T\right)^{-1} K_{u,f} K_d^{-1} \tag{A-25}$$

$$= \left(\boldsymbol{L}_B^T\right)^{-1} \boldsymbol{A} K_d^{-0.5} \tag{A-26}$$

This variable is now used in the Woodbury identity of Equation (A-20) to obtain the following:

$$\left(K_d + K_{\boldsymbol{f},u} K_{u,u}^{-1} K_{u,f}\right)^{-1} = K_d^{-1} - \boldsymbol{C}\boldsymbol{C}^T \tag{A-27}$$

Using this result and the introduced variables, the MLL of Equation (A-17) is rewritten as follows;

$$\begin{aligned}
\mathcal{L}_{PEP} =& -\frac{n}{2}\log 2\pi - \frac{1}{2}\log \mid \alpha K_{FITC} + \boldsymbol{Q}_{\boldsymbol{ff}} + \sigma_\epsilon^2 \boldsymbol{I} \mid \\
& - \frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\left(\alpha K_{FITC} + \boldsymbol{Q}_{\boldsymbol{ff}} + \sigma_\epsilon^2 \boldsymbol{I}\right)^{-1}\boldsymbol{y} - \frac{1-\alpha}{2\alpha}\operatorname{tr}\left[\log\left(I + \frac{\alpha}{\sigma_\epsilon^2}\left(\boldsymbol{K}_{\boldsymbol{ff}} - \boldsymbol{Q}_{\boldsymbol{ff}}\right)\right)\right] \\
=& -\frac{n}{2}\log 2\pi - \frac{1}{2}\log \mid \boldsymbol{B} K_d \mid -\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\left(K_d^{-1} - \boldsymbol{C}\boldsymbol{C}^T\right)\boldsymbol{y} \\
& - \frac{1-\alpha}{2\alpha}\operatorname{tr}\left[\log\left(I + \frac{\alpha}{\sigma_\epsilon^2}\left(\boldsymbol{K}_{\boldsymbol{ff}} - K_d \boldsymbol{A}^T\boldsymbol{A}\right)\right)\right] \\
=& -\frac{n}{2}\log 2\pi - \sum_{i=1}^{m}\log\left(L_{Bii}\right) - \frac{1}{2}\operatorname{tr}\left[log\left(K_d\right)\right] - \frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\left(K_d^{-1}\right)\boldsymbol{y} + \boldsymbol{y}^{\mathrm{T}}\left(\boldsymbol{C}\boldsymbol{C}^T\right)\boldsymbol{y} \\
& - \frac{1-\alpha}{2\alpha}\operatorname{tr}\left[\log\left(I + \frac{\alpha}{\sigma_\epsilon^2}\left(\boldsymbol{K}_{\boldsymbol{ff}} - K_d \boldsymbol{A}^T\boldsymbol{A}\right)\right)\right]
\end{aligned} \tag{A-28}$$

Appendix B

Dynamic model derivations

B-1 Double pendulum

A model of Figure 3-1 is derived by using Lagrange's equation of motion which is defined as follows [90]:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = -\frac{D}{\partial \dot{q}} + F, \tag{B-1}$$

where $q \in \mathbb{R}^n$ indicate the generalized displacement coordinates, $\dot{q} \in \mathbb{R}^n$ the flow coordinates in the system, $\mathcal{L}$ the Lagrangian which is the difference between kinetic K and potential energy P in the system, i.e. $\mathcal{L} = K - P$, D the dissipated energy and F the external forces that are acting on the system.

Before being able to calculate Lagrange's equation of motion for the double pendulum, the generalized displacement coordinates are defined by using an upward Cartesian coordinate system in which the x and y axis coincides with the center of the motor. The following coordinates are identified:

$$\begin{aligned} x_1 &= \frac{1}{2} l_1 sin(\theta_1) \\ y_1 &= -\frac{1}{2} l_1 cos(\theta_1) \end{aligned} \tag{B-2}$$

$$\begin{aligned} x_2 &= l_1 sin(\theta_1) + \frac{1}{2} l_2 sin(\theta_2) \\ y_2 &= -l_1 cos(\theta_1) - \frac{1}{2} l_2 cos(\theta_2) \end{aligned} \tag{B-3}$$

$$\begin{aligned} x_3 &= l_1 sin(\theta_1) + l_2 sin(\theta_2) \\ y_3 &= -l_1 cos(\theta_1) - l_2 cos(\theta_2), \end{aligned} \tag{B-4}$$

where l_i indicates the length of the i-th beam and θ_i the angular rotation of the i-th beam.

The flow coordinates in the system are obtained by taking the derivatives of the generalized displacement coordinates with respect to time. The following flow coordinates are obtained:

$$\begin{aligned} \dot{x}_1 &= \frac{1}{2} l_1 \dot{\theta}_1 cos(\theta_1) \\ \dot{y}_1 &= \frac{1}{2} l_1 \dot{\theta}_1 sin(\theta_1) \end{aligned} \tag{B-5}$$

$$\begin{aligned} \dot{x}_2 &= l_1 \dot{\theta}_1 cos(\theta_1) + \frac{1}{2} l_2 \dot{\theta}_2 cos(\theta_2) \\ \dot{y}_2 &= l_1 \dot{\theta}_1 sin(\theta_1) + \frac{1}{2} l_2 \dot{\theta}_2 sin(\theta_2) \end{aligned} \tag{B-6}$$

$$\begin{aligned} \dot{x}_3 &= l_1 \dot{\theta}_1 cos(\theta_1) + l_2 \dot{\theta}_2 cos(\theta_2) \\ \dot{y}_3 &= l_1 \dot{\theta}_1 sin(\theta_1) + l_2 \dot{\theta}_2 sin(\theta_2), \end{aligned} \tag{B-7}$$

where $\dot{\theta}_i$ is the angular velocity of the i-th beam.

All the defined coordinates are now included in the Lagrangian $\mathcal{L}$. In order to derive the Lagrangian, one should identify the amount of energy that is available in the system in the form of kinetic (K) and potential (P) energy. The potential energy is derived as follows:

$$P = m_1 g y_1 + m_2 g y_2 + m_3 g y_3, \tag{B-8}$$

where m_i indicates the mass of the i-th element and g the gravitational constant.

Filling in the y-coordinates of Equation (B-2), (B-3) and (B-4) gives:

$$\begin{aligned} P &= -\frac{1}{2} m_1 g l_1 cos(\theta_1) + m_2 g(-l_1 cos(\theta_1) - \frac{1}{2} l_2 cos(\theta_2)) + m_3 g(-l_1 cos(\theta_1) - l_2 cos(\theta_2)) \\ &= -g l_1 cos(\theta_1)(\frac{1}{2} m_1 + m_2 + m_3) - g l_2 cos(\theta_2)(\frac{1}{2} m_2 + m_3), \end{aligned} \tag{B-9}$$

The kinetic energy is derived as follows:

$$\begin{aligned} K &= \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 + \frac{1}{2} m_3 v_3^2 + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 \dot{\theta}_2^2 \\ &= \frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2} m_2 (\dot{x}_2^2 + \dot{y}_2^2) + \frac{1}{2} m_3 (\dot{x}_3^2 + \dot{y}_3^2) + \frac{1}{2} I_1 \dot{\theta}_1^2 + \frac{1}{2} I_2 \dot{\theta}_2^2, \end{aligned} \tag{B-10}$$

Where I_i indicate the inertia of the i-th beam.

Filling in the flow coordinates of Equation (B-5), (B-6) and (B-7) gives:

$$\begin{aligned}
K &= \frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 + \frac{1}{2}m_3v_3^2 + \frac{1}{2}(\frac{1}{12}m_1l_1^2)\dot{\theta}_1^2 + \frac{1}{2}(\frac{1}{12}m_2l_2^2)\dot{\theta}_2^2 \\
&= \left[\frac{1}{2}m_1\left(\frac{1}{4}l_1^2\dot{\theta}_1^2\overbrace{\left(cos(\theta_1)^2 + sin(\theta_1)^2\right)}^{=1}\right)\right] + \\
&\quad \left[\frac{1}{2}m_2\left(l_1^2\dot{\theta}_1^2\left(cos(\theta_1)^2 + sin(\theta_1)^2\right) + \frac{1}{4}l_2^2\dot{\theta}_2^2\left(cos(\theta_2)^2 + sin(\theta_2)^2\right) + \cdots\right.\right. \\
&\quad \left.\left. l_1l_2\dot{\theta}_1\dot{\theta}_2\overbrace{(sin(\theta_1)sin(\theta_2) + cos(\theta_1)cos(\theta_2))}^{=cos(\theta_1-\theta_2)}\right)\right] + \\
&\quad \left[\frac{1}{2}m_3\left(l_1^2\dot{\theta}_1^2\left(cos(\theta_1)^2 + sin(\theta_1)^2\right) + l_2^2\dot{\theta}_2^2\left(cos(\theta_2)^2 + sin(\theta_2)^2\right) + \cdots\right.\right. \\
&\quad \left.\left. 2l_1l_2\dot{\theta}_1\dot{\theta}_2\left(sin(\theta_1)sin(\theta_2) + cos(\theta_1)cos(\theta_2)\right)\right)\right] + \\
&\quad \frac{1}{24}m_1l_1^2\dot{\theta}_1^2 + \frac{1}{24}m_2l_2^2\dot{\theta}_2^2 \\
&= \frac{1}{2}\left(\frac{1}{3}m_1l_1^2 + m_2l_1^2 + m_3l_1^2\right)\dot{\theta}_1^2 + \frac{1}{2}\left(\frac{1}{3}m_2l_2^2 + m_3l_2^2\right)\dot{\theta}_2^2 + \cdots \\
&\quad \frac{1}{2}\left(m_2l_1l_2 + 2m_3l_1l_2\right)\dot{\theta}_1\dot{\theta}_2cos(\theta_1 - \theta_2)
\end{aligned} \tag{B-11}$$

Filling in kinetic and potential energy in the Lagrangian provides:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\left(\frac{1}{3}m_1l_1^2 + m_2l_1^2 + m_3l_1^2\right)\dot{\theta}_1^2 + \frac{1}{2}\left(\frac{1}{3}m_2l_2^2 + m_3l_2^2\right)\dot{\theta}_2^2 + \cdots \\
&\quad \frac{1}{2}\left(m_2l_1l_2 + 2m_3l_1l_2\right)\dot{\theta}_1\dot{\theta}_2cos(\theta_1 - \theta_2) + gl_1cos(\theta_1)(\frac{1}{2}m_1 + m_2 + m_3) + \cdots \\
&\quad gl_2cos(\theta_2)(\frac{1}{2}m_2 + m_3) \\
&= \frac{1}{2}J_1\dot{\theta}_1^2 + \frac{1}{2}J_2\dot{\theta}_2^2 + J_3\dot{\theta}_1\dot{\theta}_2cos(\theta_1 - \theta_2) + \mu_1cos(\theta_1) + \mu_2cos(\theta_2),
\end{aligned} \tag{B-12}$$

with:

$$
\begin{aligned}
J_1 &= \frac{1}{3}m_1 l_1^2 + m_2 l_1^2 + m_3 l_1^2 \\
J_2 &= \frac{1}{3}m_2 l_2^2 + m_3 l_2^2 \\
J_3 &= \frac{1}{2}m_2 l_1 l_2 + m_3 l_1 l_2 \\
\mu_1 &= g l_1 (\frac{1}{2}m_1 + m_2 + m_3) \\
\mu_2 &= g l_2 (\frac{1}{2}m_2 + m_3)
\end{aligned} \tag{B-13}
$$

Next to kinetic and potential energy, there is also dissipative energy D available in the system due to friction. There is assumed that there is friction in the links of the beams as follows:

$$D = \frac{1}{2}b_1\dot{\theta}_1^2 + \frac{1}{2}b_2(\dot{\theta}_1^2 - \dot{\theta}_2^2) \tag{B-14}$$

Also, it is possible to excite the system with an external input F. The external input is provided by a motor that is able to exert a torque T at the base of the double pendulum that rotates the first beam of the system directly and the second beam indirectly.

After defining all the elements that are needed in the calculation of the equations of motion (EoMs) of Equation (B-23), the EoMs are calculated as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -J_3\dot{\theta}_1\dot{\theta}_2 sin(\theta_1 - \theta_2) - \mu_1 sin(\theta_1) \tag{B-15}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = J_1\dot{\theta}_1 + J_3\dot{\theta}_2 cos(\theta_1 - \theta_2) \tag{B-16}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) = J_1\ddot{\theta}_1 + J_3\ddot{\theta}_2 cos(\theta_1 - \theta_2) - J_3\dot{\theta}_2 cos(\theta_1 - \theta_2)(\dot{\theta}_1 - \dot{\theta}_2) \tag{B-17}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = J_3\dot{\theta}_1\dot{\theta}_2 sin(\theta_1 - \theta_2) - \mu_2 sin(\theta_2) \tag{B-18}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = J_2\dot{\theta}_2 + J_3\dot{\theta}_1 cos(\theta_1 - \theta_2) \tag{B-19}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right) = J_2\ddot{\theta}_2 + J_3\ddot{\theta}_1 cos(\theta_1 - \theta_2) - J_3\dot{\theta}_1 sin(\theta_1 - \theta_2)(\dot{\theta}_1 - \dot{\theta}_2) \tag{B-20}$$

$$\frac{\partial D}{\partial \dot{\theta}_1} = (b_1 + b_2)\dot{\theta}_1 \tag{B-21}$$

$$\frac{\partial D}{\partial \dot{\theta}_2} = -b_2\dot{\theta}_2 \tag{B-22}$$

Filling in Equation (B-15) – (B-22) in the following equation to obtain the EoMs:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} - \frac{\partial \mathcal{L}}{\partial \theta_i} = -\frac{D}{\partial \dot{\theta}_i} + F_i, \tag{B-23}$$

Which provides the following EoMs:

$$J_1\ddot{\theta}_1 + J_3\ddot{\theta}_2 cos(\theta_1 - \theta_2) + J_3\dot{\theta}_2^2 sin(\theta_1 - \theta_2) + \mu_1 sin(\theta_1) = -(b_1 + b_2)\dot{\theta}_1 + T \tag{B-24}$$

$$J_2\ddot{\theta}_2 + J_3\ddot{\theta}_1 cos(\theta_1 - \theta_2) - J_3\dot{\theta}_1^2 sin(\theta_1 - \theta_2) + \mu_2 sin(\theta_2) = b_2\dot{\theta}_2 \tag{B-25}$$

For numerical purposes the EoMs are rewritten to state space (SS) form as follows:

$$\begin{bmatrix} J_1 & J_3 cos(\theta_1 - \theta_2) \\ J_3 cos(\theta_1 - \theta_2) & J_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = - \begin{bmatrix} J_3\dot{\theta}_2^2 sin(\theta_1 - \theta_2) \\ -J_3\dot{\theta}_1^2 sin(\theta_1 - \theta_2) \end{bmatrix} - \begin{bmatrix} \mu_1 sin(\theta_1) \\ \mu_2 sin(\theta_2) \end{bmatrix} - \begin{bmatrix} (b_1 + b_2)\dot{\theta}_1 \\ -b_2\dot{\theta}_2 \end{bmatrix} + \begin{bmatrix} T \\ 0 \end{bmatrix} \tag{B-26}$$

Appendix C

Results

C-1 Performance of the GP-MPC in simulation

The model predictive control (MPC) controller in the simulation phase is analyzed with respect to its mean tracking error, the number of inducing points, and the length of the prediction horizon. However, the mean of the tracking error and the mean of the calculational time is missing a standard deviation in figures 4-8 and 4-9 which prevents the figures to be chaotic. These figures are restated in the following to provide the variance bounds.

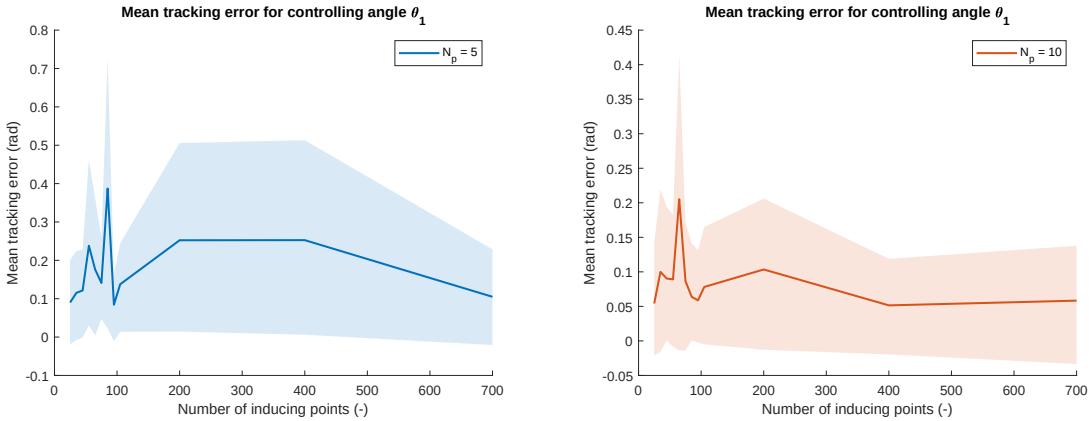


Figure C-1: Mean tracking error with standard deviations of the model predictive controller using sparse GP models with different amounts of inducing points. The left figure indicates the tracking error for controlling θ_1 with a prediction horizon of $N_\mathrm{p} = 5$ and the right figure indicates the same for a prediction horizon of $N_\mathrm{p} = 10$.

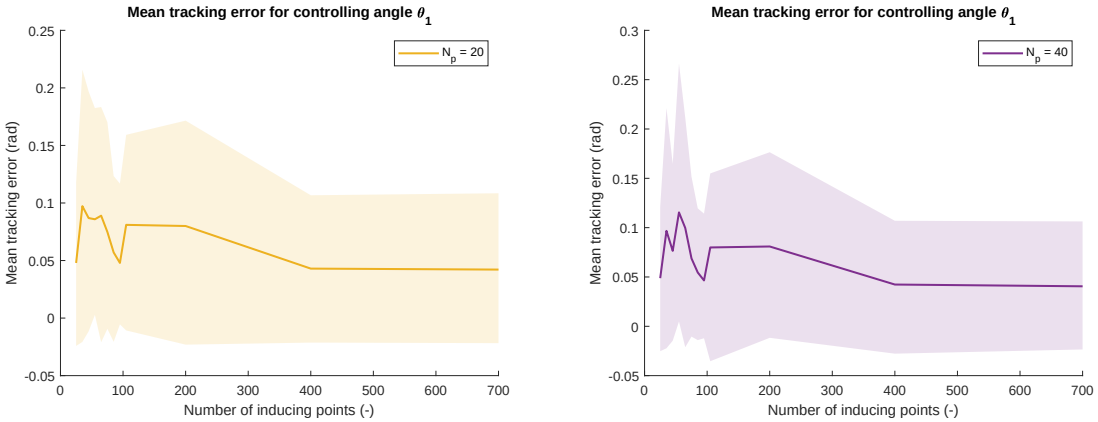


Figure C-2: Mean tracking error with standard deviations of the model predictive controller using sparse GP models with different amounts of inducing points. The left figure indicates the tracking error for controlling θ_1 with a prediction horizon of $N_\mathrm{p} = 20$ and the right figure indicates the same for a prediction horizon of $N_\mathrm{p} = 40$.

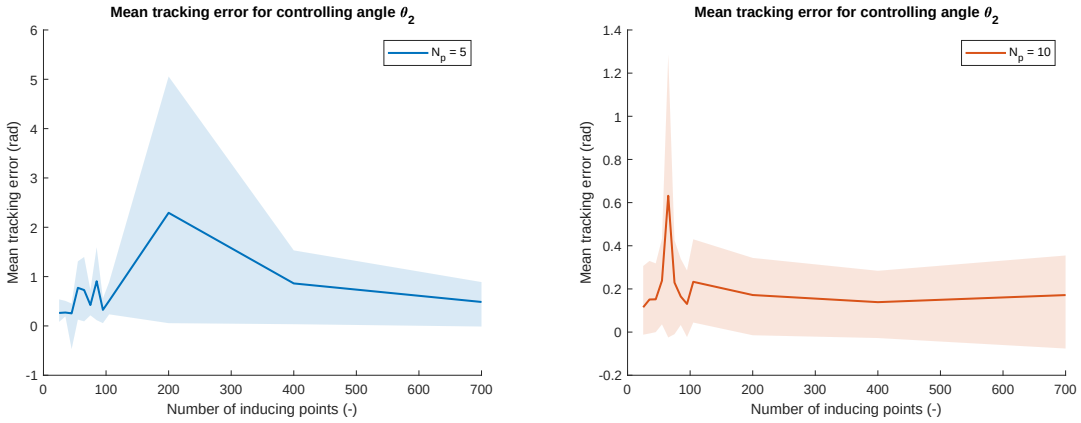


Figure C-3: Mean tracking error with standard deviations of the model predictive controller using sparse GP models with different amounts of inducing points. The left figure indicates the tracking error for controlling θ_2 with a prediction horizon of $N_\mathrm{p} = 5$ and the right figure indicates the same for a prediction horizon of $N_\mathrm{p} = 10$.

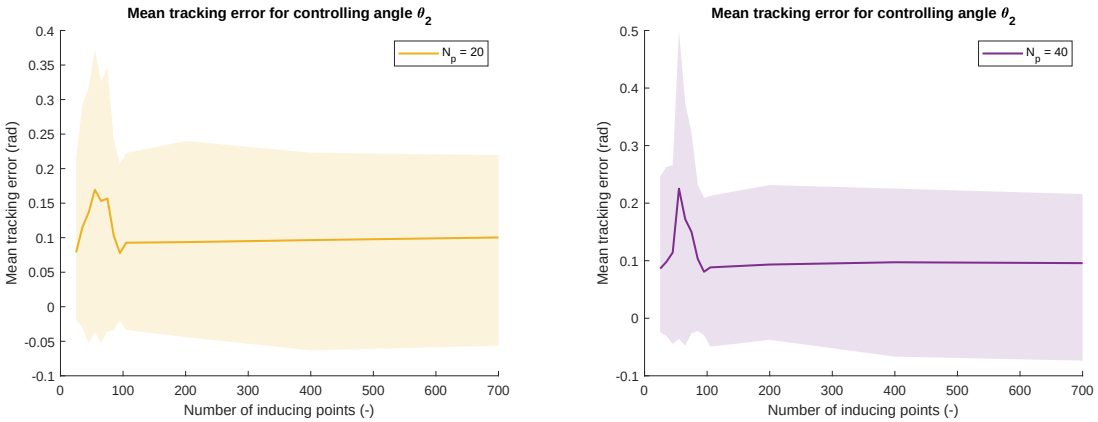


Figure C-4: Mean tracking error with standard deviations of the model predictive controller using sparse GP models with different amounts of inducing points. The left figure indicates the tracking error for controlling θ_2 with a prediction horizon of $N_\mathrm{p} = 20$ and the right figure indicates the same for a prediction horizon of $N_\mathrm{p} = 40$.

Bibliography

[1] S. A. Ajwad, J. Iqbal, M. I. Ullah, and A. Mehmood, "A systematic review of current and emergent manipulator control approaches," *Frontiers of mechanical engineering*, vol. 10, no. 2, pp. 198–210, 2015.

[2] K. Zhou, C. Fu, and S. Yang, "Big data driven smart energy management: From big data to big insights," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 215–225, 2016.

[3] S. Xu and H. Peng, "Design, analysis, and experiments of preview path tracking control for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 48–58, 2019.

[4] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, "Learning and control using Gaussian processes," in *2018 ACM/IEEE 9th international conference on cyber-physical systems (ICCPS)*, IEEE, 2018, pp. 140–149.

[5] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[6] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[7] A. Jain, "Methods for data-driven model predictive control," Ph.D. dissertation, University of Pennsylvania, 2020.

[8] U. Rosolia, X. Zhang, and F. Borrelli, "Data-driven predictive control for autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 259–286, 2018.

[9] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[10] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks, fuzzy models, and Gaussian processes*. Springer Nature, 2020.

[11] F. Smarra, A. Jain, T. De Rubeis, D. Ambrosini, A. D'Innocenzo, and R. Mangharam, "Data-driven model predictive control using random forests for building energy optimization and climate control," *Applied energy*, vol. 226, pp. 1252–1272, 2018.

[12] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[13] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[14] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[15] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[16] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable gps," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[17] E. Bradford, L. Imsland, M. Reble, and E. A. del Rio-Chanona, "Hybrid Gaussian process modeling applied to economic stochastic model predictive control of batch processes," in *Recent Advances in Model Predictive Control*, Springer, 2021, pp. 191–218.

[18] J. Matschek, T. Gonschorek, M. Hanses, N. Elkmann, F. Ortmeier, and R. Findeisen, "Learning references with Gaussian processes in model predictive control applied to robot assisted surgery," in *2020 European Control Conference (ECC)*, IEEE, 2020, pp. 362–367.

[19] K. Haninger, C. Hegeler, and L. Peternel, "Model predictive control with Gaussian processes for flexible multi-modal physical human robot interaction," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 6948–6955.

[20] S. Mosharafian, M. Razzaghpour, Y. P. Fallah, and J. M. Velni, "Gaussian process based stochastic model predictive control for cooperative adaptive cruise control," in *2021 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2021, pp. 17–23.

[21] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio Chanona, "Stochastic data-driven model predictive control using Gaussian processes," *Computers & Chemical Engineering*, vol. 139, p. 106 844, 2020.

[22] M. Maiworm, D. Limon, J. M. Manzano, and R. Findeisen, "Stability of Gaussian process learning based output feedback model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 455–461, 2018.

[23] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proceedings of the 2004 American control conference*, IEEE, vol. 3, 2004, pp. 2214–2219.

[24] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of unknown non-linear systems," *IET Control Theory & Applications*, vol. 11, no. 5, pp. 703–713, 2017.

[25] K. J. Keesman and K. J. Keesman, *System identification: an introduction*. Springer, 2011, vol. 2.

[26] J. Kocijan, *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[27] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse Gaussian process approximations," *Advances in neural information processing systems*, vol. 29, 2016.

[28] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," in *2014 American control conference*, IEEE, 2014, pp. 2413–2419.

[29] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, *Gaussian processes and kernel methods: A review on connections and equivalences*, 2018. arXiv: 1807.02582 [stat.ML].

[30] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[31] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[32] R. M. Neal, "Monte Carlo implementation of gaussian process models for Bayesian regression and classification," *arXiv preprint physics/9701026*, 1997.

[33] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[34] L. Ljung, "System identification," in *Signal analysis and prediction*, Springer, 1998, pp. 163–173.

[35] N. I. Chaudhary and M. A. Z. Raja, "Design of fractional adaptive strategy for input nonlinear Box-Jenkins systems," *Signal Processing*, vol. 116, pp. 141–151, 2015.

[36] R. de Oliveira Teloli, L. G. Villani, S. da Silva, and M. D. Todd, "On the use of the GP-NARX model for predicting hysteresis effects of bolted joint structures," *Mechanical Systems and Signal Processing*, vol. 159, p. 107 751, 2021.

[37] S. Särkkä, "The use of Gaussian processes in system identification," *arXiv preprint arXiv:1907.06066*, 2019.

[38] S. Rannen, C. Ghorbel, and N. Braiek, "NARMAX structure and identification of coupled mass-spring-damper system," in *3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16)*, 2016, pp. 475–480.

[39] C. Rasmussen and Z. Ghahramani, "Occam's razor," *Advances in neural information processing systems*, vol. 13, 2000.

[40] H. Oyama and M. Yamakita, "Online learning of automotive gasoline engine model using robust recursive Gaussian process," in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 3975–3980.

[41] K. Worden, W. Becker, T. Rogers, and E. Cross, "On the confidence bounds of Gaussian process NARX models and their higher-order frequency response functions," *Mechanical Systems and Signal Processing*, vol. 104, pp. 188–223, 2018.

[42] E. R. Ackermann, J. P. De Villiers, and P. Cilliers, "Nonlinear dynamic systems modelling using Gaussian processes: Predicting ionospheric total electron content over south africa," *Journal of Geophysical Research: Space Physics*, vol. 116, no. A10, 2011.

[43] Y. Wang, Z. Wang, K. Han, P. Tiwari, and D. B. Work, "Personalized adaptive cruise control via Gaussian process regression," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 1496–1502.

[44] R. H. Shumway, D. S. Stoffer, and D. S. Stoffer, *Time series analysis and its applications*. Springer, 2000, vol. 3.

[45] S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman, "Identification of Gaussian process state space models," *Advances in neural information processing systems*, vol. 30, 2017.

[46] T. Beckers and S. Hirche, "Stability of Gaussian process state space models," in *2016 European Control Conference (ECC)*, IEEE, 2016, pp. 2275–2281.

[47] K. Berntorp, "Online Bayesian inference and learning of Gaussian-process state–space models," *Automatica*, vol. 129, p. 109 613, 2021.

[48] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle mcmc," *arXiv preprint arXiv:1306.2861*, 2013.

[49] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational Gaussian process state-space models," in *Advances in neural information processing systems*, 2014, pp. 3680–3688.

[50] R. Turner, M. Deisenroth, and C. Rasmussen, "State-space inference and learning with Gaussian processes," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 868–875.

[51] A. McHutchon and C. Rasmussen, "Gaussian process training with input noise," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[52] M. Titsias and N. D. Lawrence, "Bayesian Gaussian process latent variable model," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 844–851.

[53] A. Girard, C. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," *Advances in neural information processing systems*, vol. 15, 2002.

[54] A. Damianou and N. D. Lawrence, "Semi-described and semi-supervised learning with Gaussian processes," *arXiv preprint arXiv:1509.01168*, 2015.

[55] A. Girard, *Approximate methods for propagation of uncertainty with Gaussian process models*. University of Glasgow (United Kingdom), 2004.

[56] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[57] A. Melkumyan and F. T. Ramos, "A sparse covariance function for exact Gaussian process inference in large datasets," in *Twenty-first international joint conference on artificial intelligence*, 2009.

[58] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," *The Journal of Machine Learning Research*, vol. 11, pp. 1865–1881, 2010.

[59] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse Gaussian process approximations," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016. [Online]. Available: `https://proceedings.neurips.cc/paper/2016/file/7250eb93b3c18cc9daa29cf58af7a004-Paper.pdf`.

[60] B. W. Silverman, "Some aspects of the spline smoothing approach to non-parametric regression curve fitting," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 47, no. 1, pp. 1–21, 1985.

[61] G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. E. Klein, "The bias-variance tradeoff and the randomized GACV.," in *NIPS*, Citeseer, 1998, pp. 620–626.

[62] A. J. Smola and P. L. Bartlett, "Sparse greedy Gaussian process regression," in *Advances in neural information processing systems*, 2001, pp. 619–625.

[63] L. Csató and M. Opper, "Sparse on-line Gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[64] M. W. Seeger, C. K. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *International Workshop on Artificial Intelligence and Statistics*, PMLR, 2003, pp. 254–261.

[65] T. D. Bui, J. Yan, and R. E. Turner, "A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 3649–3720, 2017.

[66] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, p. 1257, 2006.

[67] M. K. Titsias, "Variational model selection for sparse Gaussian process regression," *Report, University of Manchester, UK*, 2009.

[68] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between Gaussian mixture models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, IEEE, vol. 4, 2007, pp. IV–317.

[69] T. P. Minka, "A family of algorithms for approximate Bayesian inference," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[70] B. Kouvaritakis and M. Cannon, "Model predictive control," *Switzerland: Springer International Publishing*, vol. 38, 2016.

[71] N. A. Seco Rodrigues, "Gaussian process regression for data-driven model predictive control," 2021.

[72] A. T. Schwarm and M. Nikolaou, "Chance-constrained model predictive control," *AIChE Journal*, vol. 45, no. 8, pp. 1743–1752, 1999.

[73] T. X. Nghiem and C. N. Jones, "Data-driven demand response modeling and control of buildings with Gaussian processes," in *2017 American Control Conference (ACC)*, IEEE, 2017, pp. 2919–2924.

[74] L. Hewing and M. N. Zeilinger, "Stochastic model predictive control for linear systems using probabilistic reachable sets," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 5182–5188.

[75] E. Bradford, L. Imsland, and E. A. del Rio-Chanona, "Nonlinear model predictive control with explicit back-offs for Gaussian process state space models," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 4747–4754.

[76] C. Vuik, F. J. Vermolen, M. B. Gijzen, and M. Vuik, *Numerical methods for ordinary differential equations (2nd ed.)* DAP, Delft Academic Press., 2018.

[77] *Sc52035 info*. [Online]. Available: `http://homepage.tudelft.nl/b3k1c/sc52035/setups/rotating-pendulum.html`.

[78] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification*, Springer, 2001, pp. 547–577.

[79] D. J. MacKay *et al.*, "Introduction to Gaussian processes," *NATO ASI series F computer and systems sciences*, vol. 168, pp. 133–166, 1998.

[80] M. . Verhaegen and V. . Verdult, *Filtering and System Identification*. Cambridge, Verenigd Koninkrijk: Cambridge University Press, 2012.

[81] MATLAB, *9.12.0.1884302 (R2022a)*. Natick, Massachusetts: The MathWorks Inc., 2022.

[82] N. J. Higham, "Cholesky factorization," *Wiley interdisciplinary reviews: computational statistics*, vol. 1, no. 2, pp. 251–254, 2009.

[83] A. W. Max, "Inverting modified matrices," in *Memorandum Rept. 42, Statistical Research Group*, Princeton Univ., 1950, p. 4.

[84] T. Galy-Fajou and M. Opper, "Adaptive inducing points selection for Gaussian processes," *arXiv preprint arXiv:2107.10066*, 2021.

[85] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019. DOI: `10.1007/s12532-018-0139-4`.

[86] T. X. Nghiem, T.-D. Nguyen, and V.-A. Le, "Fast gaussian process based model predictive control with uncertainty propagation," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2019, pp. 1052–1059.

[87] M. Ellis, J. Liu, and P. D. Christofides, "Economic model predictive control," *Springer*, vol. 5, no. 7, p. 65, 2017.

[88] B. Pluymers, J. Rossiter, J. Suykens, and B. De Moor, "A simple algorithm for robust mpc," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 257–262, 2005.

[89] F. Zhang, *The Schur complement and its applications*. Springer Science & Business Media, 2006, vol. 4.

[90] H. Vallery and A. L. Schwab, *Advanced Dynamics*. TU Delft, 2018.

List of acronyms

APRBS	amplitude modulated pseudo random binary sequence
ARD	automatic relevance determination
BFGS	Broyden–Fletcher–Goldfarb–Shanno
DCSC	Delft Center for Systems and Control
TU Delft	Delft University of Technology
DIC	deterministic inducing conditional
DTC	deterministic training conditional
ELBO	evidence lower bound
EM	expectiation maximization
EoMs	equations of motion
EP	expectation propagation
FIC	fully independent conditional
FITC	fully independent training conditional
GP	Gaussian process
GP-LVM	Gaussian process latent variable model
GPR	Gaussian process regression
GPs	Gaussian processes
GPSS	Gaussian process state space
KL	Kullback–Leibler
MCMC	Markov chain Monte Carlo
ML	machine learning
MLL	marginal log likelihood
MPC	model predictive control
MSLL	mean standardised log loss
NARMAX	nonlinear autoregressive moving average with exogenous inputs

NARX	nonlinear autoregressive model with exogenous input
NBJ	nonlinear Box-Jenkins
NFIR	nonlinear finite imulse response
NIGP	noisy input gaussian process
NMLL	negative marginal log likelihood
NMPC	nonlinear model predictive controller
NOE	nonlinear output error
NSS	nonlinear state space
PEM	prediction error method
PEP	power expectation propagation
PITC	partially independent training conditional
PMCMC	particle Markov chain Monte Carlo
PLV	projected latent variables
PPA	projected process approximation
PRBS	pseudo random binary sequence
RK-4	Runge-Kutta of the fourth order
SE	squared exponential
SIMO	single input multiple output
SoD	subset of data
SS	state space
SSGP	sparse spectrum Gaussian process
SoR	subset of regressors
SGPP	sparse Gaussian processes using pseudo-inputs
VAF	variance accounted for
VFE	variational free energy