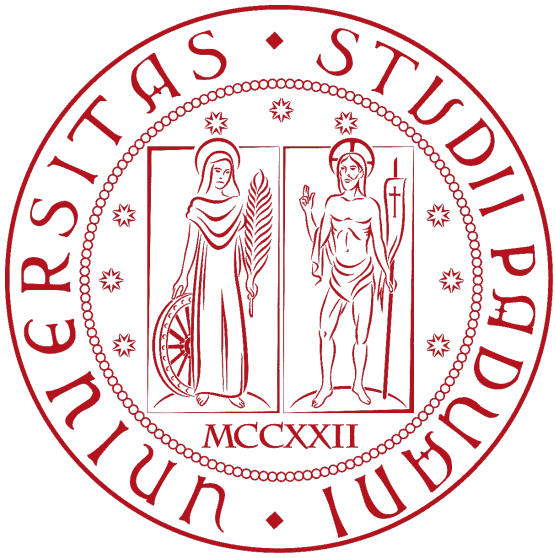
Università degli Studi di Padova



Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in
Ingegneria dell'Automazione

Application of Generalized Kernels to Learning-Based Non-Linear Model Predictive Control

Laureando:
Diego Fernandez
Galindo

Relatore:
Prof. Mattia
Bruschetta

Correlatore:
Dott. Enrico Picotti

ANNO ACCADEMICO 2022-2023

13 Dicembre 2022

A mia madre.

Abstract

Gaussian Processes are a powerful tool that is finding successful applications in the Control Systems field due to their high flexibility and ability to identify dynamical systems. In a general context, gaussian process regression is a bayesian non-parametric method which makes predictions based on the posteriori probability density given training data. The behavior of the predictions (regressed function) is completely characterized by the covariance function, whose properties are directly inhereted by the regressed function. In many problems, such function is chosen *ad hoc* and depends on prior knowledge of the underlying phenomena, i.e. depends on human intervention.
Learning-Based Non-Linear Model Predictive Control (LB-NMPC) scheme is a data-driven implementation of the MPC used to control non-linear dynamical systems while satisfying plant constraints, it is aimed at improving the nominal model available to the controller by approximating underlying missing dynamics with a gaussian process using past plant observations as training data. We implement the so-called Generalized Kernels to the Gaussian process within the LB-NMPC, exploiting the spectral representation of a positive-definite function that is modelled as mixture of a basis function. Such basis function can have desired properties such as smoothness or degree of differentiabilty, generating richer model classes that can be used to identify more complex dynamical systems. This procedure automatizes the selection of the kernel function, while still maintaining an analytic expression of the regressed function, allowing its implementation on the MPC scheme and, in particular, it has a strong link with RKHS framework that is useful to analyze the model class rigorously.

Contents

Chapter 1

Introduction

The past few decades have been marked by the growing computational power available to us, giving possibility to the development fast algorithms to solve complex optimization problems. Model Predictive Control is an approach which solves iteratively an optimization problem to finish a control task even when the system's behavior presents non-linear dynamics, guaranteeing high-performance with respect to other control techniques. However, the quality of this control technique depends on the accuracy of the mathematical description of the physical system, making the identification procedure for the physical system a key aspect. In the case of linear systems, System Identification is a well established field explaining detailed techniques based on the reconstruction of the system's impulse response given observations, obtaining excellent results. However, the case of non-linear system identification is a still under deveplopment by the most part, mainly because of the huge diversity of non-linearities that can be present in a given system, making most of the standard identification methods not well-suited for the task. Although difficult, in most of the cases there's the core idea of using a basis function k that can approximate arbitrarily well any function if enough basis functions are added together [12],[24]. This is the case for neural networks, these simple structures can approximate any function and they have been successfully implemented in some real-time applications to infer patterns from data or to learn dynamical systems, achieving incredible results that make them the most used model structures [10]. Although their great capabilities to learn from data, their complex structures could make their application within a MPC scheme rather difficult. Another similar technique, which has been neglected in the past given the heavy computational burden of its training, are Gaussian Processes. This approach, which falls into the non-parametric Bayesian framework[14], also models the identification of the physical system as a sum of basis functions k but their locations on the operational space depends on the observed data. Moreover, the kernel function is actually meaningfull for regression, i.e. its properties are inhereted by the regressed function, and its analytical expression can be quite simple, making it an attractive method for MPC schemes. Gaussian processes have been readily proposed for joint learning and control tasks [11], moreover there's a vast literature available that describes their flexibility for learning and

prediction capabilities [25] in different machine learning tasks. In this thesis, we are interested in a class of kernels which have been rarely implemented to learn non-linear dynamical systems. This class of kernels are flexible by design, and should be able to infer much more complex behavior, outperforming the standard kernel choices. Our goal is to apply gaussian processes to the Model Predictive Control scheme, in order to achieve high performance during a complex control task. This joint effort between learning and control is relatively new within NMPC applications[6], mainly because now we are able to efficiently introduce data-driven techniques for tasks that require a high-computational speed. Some of these learning techniques are based on the introduction of a Gaussian process aimed at improving different aspects of the overall controller setup, either tunning directly the objective function [15] or improving the mathematical models[17] for solving the optimization problem. Not only that, gaussian processes were also implemented withing a NMPC scheme to improve the performance, pushing the controller to take less conservative actions while keeping the controlled system within a safety region to avoid possible damages [7].

1.1 Improving physical models from data

Through out the thesis we will focus on the improvement of the mathematical description of the system by means of Gaussian processes, aimed to improve the solution of the following optimal control problem:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u}}{\operatorname{argmin}} \quad & \int_{t_0}^{t_f} l(x(t), u(t))dt + l_n(x(t_f)) \\ s.t \quad & \dot{x}(t) = f_t(x(t), u(t), t; p) \end{aligned}$$

where f_t are the system's dynamics. This problem has a huge history behind, many techniques have been developed ranging from Calculus of Variations to Dynamic Programming. Here we focus on a more pragmatic way, based on the discretization of the problem which allows the use of the heavy computational machinery we have at hand. Moreover, we will focus on the behavior of the system which can be influenced by the presence of external disturbances or uncertainty present in some parameters $p \in \mathbb{R}^{n_p}$ of its mathematical description, directly affecting the resulting optimal control trajectory $\mathbf{u}(t)^*$. Leaving aside disturbances, our goal is to improve this mathematical description, at least locally, based on the available data of the system's behavior. That is, given a nominal description f_n we would like to improve this description based on the prediction errors observed in previous control tasks. More explicitly, the *true* dynamics $f_t(x, u)$ of the model can be decomposed:

$$f_t(x, u) = f_n(x, u) + \psi(x, u)$$

where ψ is called the mismatch model. Depending on the knowledge available to us, we can set two kind of models:

- **Grey-Box**: We have a mathematical description from physical principles, the mismatch model will be formulated based prediction errors of the nominal model and the measured variables (i.e. velocity prediction).

- **Black-Box**: We don't have a mathematical description, the mismatch model will be formulated from differences between observations (i.e. velocity differences).

These representations are general, many system identification techniques have these qualities (i.e. ARMAX models in the linear case [13]), but we make a clear distinction: we won't use a parametric representation of the models. That is, the regressed functions will live in an infinite-dimensional space. This is the case of Gaussian Process, where the estimated functions are just an infinite collection of random gaussian variables. By modelling ψ as a Gaussian process, the prediction values at the particular test point (x, u) have a conditional probability with respect to the observed data $\{(x_i, u_i)\}_{i=1}^{N}$. At first sight this seems a rather complicated way of making estimation, but the predicted values will have a neat expression of a weigthed sum

$$\psi(x, u) = \sum_{i=1}^{N} k(x, u; x_i, u_i)\alpha_i$$

allowing its implementation within the NMPC scheme while mainting its powerful prediction performance. As can be noticed, this prediction depends on the chosen basis function called *kernel*. The flexibility of Gaussian process comes from this function, selecting the right kernel can result in a drastic change of behavior as we will discuss later. Our focus will be on an analytical approximation of this kernel function which should lead to a kernel that best represents data, hence improving performance in the NMPC. Most of the earlier work about *approximating* a gaussian process was directed at reducing the computational training or prediction time given the necessity to invert an $N \times N$ covariance matrix which has $O(N^3)$ computational complexity: adding more data-points to the prediction model meant recomputing the covariance matrix. This made applying gaussian processes unappealing for high-amounts of data, techniques about approximating or reducing the complete inversion of the covariance matrices were developped [19] by taking artificious inputs, inducing inputs, that reduce the overall computational burden while keeping the prediction accuracy of the gaussian process. This lead to approximating the kernel function itself, which completely characterizes the covariance matrix, based on properties about its frequency components [21] particularly to fasten the predictions. This approach was later generalized to obtain a highly flexible class of kernels capable of approximating any kernel called *Generalized Spectral Kernels*. Within this thesis we analyze the behavior of these kernel approximations in the NMPC, trying to establish if their flexibility leads to an overall improvement of the closed-loop performance and if they are suitable for non-linear identification. Our model benchmark will be the Cart-Pendulum system, we will compare the generelized kernel with the Squared Exponential

kernel, widely used in the machine learning field, moreover we will establish their generalization capabilities by employing them in different control tasks.

1.1.1 Organization of the thesis

The main programming tools we used in this work are based on Algorithmic Differentiation, which is a recent programming paradigm. The core idea is that complex functions in a computer program are compositions of elementary arithmetic sequences, using a symbolic framework combined with a graph structure makes it possible to track these sequences and compute their derivative either in a forward or backward fashion, this makes their numerical evaluation much faster and suitable for complex optimization problems . In this work, we use the open-source software Lb-MATMPC which has two components:

- **MATMPC**: A MATLAB-based fast NMPC solver which uses on CasADi[3], a general-tool for gradient-based numerical optimization based on AD in symbolic framework, heavily focused in optimal control.
- **gpr-torch**: An open-source Python library for Gaussian Processes which uses Pytorch, an optimized tensor library based on AD for creating and training of deep neural networks.

The thesis will be structured as follows:

- **Chapter 2**: Here we will give the main components of Non-Linear Model Predictive Control and the techniques used to solve them.
- **Chapter 3**: We will introduce Gaussian Processes and their main properties, linking its learning properties to standard system identification and their implementation on the NMPC scheme.
- **Chapter 4**: The focus will be on the study and generalization of the spectral kernel, where the ideas come from and why it can have great capabilities for non-linear identification.
- **Chapter 5**: Simulation results of the generalized kernel, focusing on the training procedures and later confronting it with the Squared Exponential kernel. We will test the gaussian models with two different nominal models, in two different control tasks, discussing the results of the generalization properties of the kernels to determine the overall performance.
- **Conclussions** We conclude by describing the results obtained with this work, highlighting the main aspects and giving a general direction for further investigations.

Chapter 2

Non-Linear Model Predictive Control

In this chaper we introduce the Model Predictive Control formulation, focusing on its application to Non-Linear dynamical systems. We give a brief introduction to the main ingredients for the linear case, highlighting the crucial aspects that make this approach exceptional for control of complex systems while achieving high-performance and later focusing on the applied tecniques for non-linear systems throughout the thesis. This interest arises from the growing computational capabilities of modern computers, which allows to solve in real-time control tasks at high-frequency sampling using fast algorithms based on approximations of the original problem. Moreover, we will also focus on cases that can give rise to unsatisfactory performance, mainly parametric errors present during the modelling of the physical system. We will set the optimal control problem for the cart-pendulum system, which is our benchmark model to test the integration of Gaussian processes for the learning of the mismatch model. Given the high accuracy of modern computers, it is correct to assume that simulations of the inverted pendulum are accurate enough to validate our proposed learning techniques.

2.1 Model Predictive Control

The roots of Model Predictive Control [16] are based on optimization, mainly the study of the Linear Quadratic Regulator problem:

$$\min_{\mathbf{u}} \quad \sum_{k=0}^{N-1} \frac{1}{2}(x_k^T Q x_k + u_k^T R u_k) + x_N^T Q_N x_N \tag{2.1}$$

$$\text{s.t.} \quad \begin{cases} x_{k+1} = Ax_k + Bu_k \\ x_0 = \hat{x}_0 \end{cases} \tag{2.2}$$

where $x_i \in \mathbb{R}^{n_x}$, $u_i \in \mathbb{R}^{n_u}$ are state and control variable, $\hat{x}_0$ is the initial observed state, the matrices $Q, R, Q_N > 0$ are assumed positive definite and $\mathbf{u} = [u_1, \dots, u_{N-1}]$ are the decision variables to optimize. Under general assumptions about the system, this problem admits a (unique) optimal solution, i.e. a

sequence $\mathbf{u}^*$ that minimizes the objective function and drives the state variable to 0, moreover the control sequence has a feedback form $u_i^* = K(i)x(i)$ where $K(i)$ comes from solving a Discrete Riccati Equation in a backward fashion. In a perfect setting, this optimal control trajectory could be applied in an open-loop fashion and complete the control task, nevertheless the limitations of this approach is that the true system's behavior may have some discrepancies with respect to the predicted solution $\mathbf{x}^*$ from the initial state $\hat{x}_0$, either from parametric errors within the nominal model system or measurements errors of the state, blindly applying the optimal solution could result unsatisfactory performance. Another key point is the physical limitations of the control trajectory, i.e. actuators limits, or confinement of state vector to be within some regions for safety reasons. That is:

$$x_{\min} \leq \mathbf{x} \leq x_{\max}$$
$$u_{\min} \leq \mathbf{u} \leq u_{\max}$$

The MPC formulation is capable of handling these difficulties by solving the optimal control problem at each iteration of the control task. At stage k, given the available state measurement $x_k = \hat{x}_k$, it solves (2.2) for the prediction horizon $[k, \ldots, k+N-1]$ and applies the first element of the control sequence $u_k(0) = u_0^*$ to the real system, shifting the prediction horizon $[k+1, \ldots, k-N]$ and re-iterating the procedure once the measurement of x_{k+1} is available, as illustrated in figure (2.1).

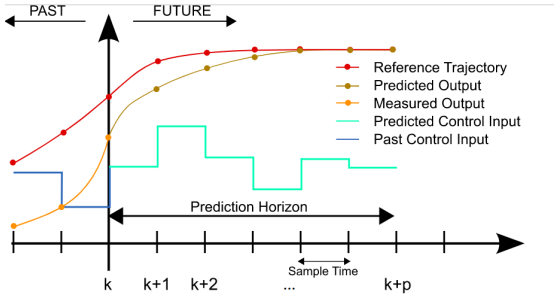


Figure 2.1: Model Predictive Control scheme.

Constraints can be introduced within the optimization problem, given a convex cost, these can be solved efficiently using Quadratic Programming (QP) solvers. In the end, we have the main ingredients for the MPC formulation:

- **Cost function**: Usually convex function for efficient solving, it has to reflect the goals of the control task.
- **System dynamics**: It's the mathematical description of the physical system. The accuracy of the modelling heavily influences the computed optimal control trajectory.
- **Constraints**: These are conditions that must be satisfied throughout the control task, imposed either on the state or the control variables. They can

be hard, i.e. imposed by the physical system, or soft, conditions we'd like to be satisfied but are not necessarily achievable.

2.2 Non-Linear Model Predictive Control

We focus now on the extension of the MPC paradigm to the non-linear dynamics case, in particular on the Direct Optimal Control approach, which is based on a particular discretization of the state and control variables, allowing the approximation of the original problem into a Non-Linear Programming (NLP) problem. This NLP formulation can be solved with the implementation of fast algorithms to compute the optimal decision variables making the integration of moving prediction horizon an appealing technique, even in cases where the dynamics of the physical system are highly complex, maintaining high performance throughout all the stages of the control task. This comes with a *caveat*, we must have a high sampling frequency and the mathematical description of the system has to be accurate enough to make the predictions representative of what is happening in real-time. From first principles, we can obtain a mathematical description of the system at hand as

$$\dot{x}(t) = f(x(t), u(t), t; p) \tag{2.3}$$

where $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ are the state and control variables, $p \in \mathbb{R}^{n_p}$ are parameters which characterize the system. Knowledge of the values of p is important to guarantee high-accuracy prediction, which is sometimes not the case, we will focus later on that problem.

2.2.1 Setting up the Non-Linear Programming problem

Assume now that the system dynamics are smooth enough, i.e. Lipschitz, we can pose now pose the continuous control problem as

$$\min_{x(\cdot),u(\cdot)} \quad \int_{t_0}^{t_f} l(x(t), u(t))dt + l_f(x(t_f)) \tag{2.4}$$

$$\text{s.t} \quad \begin{cases} x(t_0) = \hat{x}_0 \\ \dot{x}(t) = f(x(t), u(t), t; p) \\ r(x(t), u(t)) \leq 0, \forall t \in [t_0, t_f] \\ r_f(x(t_f)) \leq 0 \end{cases} \tag{2.5}$$

where l and l_f are the objective function, $r(\cdot)$ and $r_f(\cdot)$ are the path and terminal inequality constraints. The optimization problem depends on the value of $\hat{x}_0$ which is the initial condition of the system. Many solving techniques are available, we put our focus on the Direct Multiple Shooting approach [20], where we divide the prediction horizon $[t_0, t_f]$ into N *shooting intervals* $[\tau_k, \tau_{k+1}]$ with $k = 0, \ldots, N-1$, obtaining $N+1$ grid points $t_0 = \tau_0 < \tau_1 \cdots < \tau_N = t_f$.

The control trajectory is parametrized such that it remains constant during an interval:

$$u(t) = u_k, \forall t \in [\tau_k, \tau_{k+1})$$

This control trajectory parametrization, together with the multiple shooting grid, results in the representation of state trajectory as N coupled initial value problems

$$\begin{cases} \dot{x}(t) = f(x(t), u_i, t; p), \forall t \in [\tau_i, \tau_{i+1}) \\ x(\tau_i) = x_i \end{cases}$$

In multiple shooting, $N+1$ *shooting points* $(x_0, \ldots, x_N)$ are defined exactly on each grid point such that these variables satisfy $x_i = x(\tau_i)$. To take into consideration the system's dynamics, the *continuity constraints* are imposed

$$x_{i+1} = \Xi(\tau_i, x_i, u_i), \quad i = 0, \ldots, N-1$$

where $\Xi(\cdot)$ is an integration operator which solves the IVP in the interval $[\tau_i, \tau_{i+1})$ and returns the solution at the grid point τ_{i+1}. Throughout this thesis we use a uniform grid, and apply the same integration operator. We obtain the following discrete-time autonomous system dynamics:

$$x_{i+1} = \phi(x_i, u_i), \quad i = 0, \ldots, N-1$$

where $\phi(\cdot)$ is the resulting function from calling the integration operator. Regarding the path constraints, they are imposed directly on the shooting grid points:

$$\begin{cases} r(x_i, u_i) \leq 0, & i = 0, \ldots, N-1 \\ r_f(x_N) \leq 0 \end{cases}$$

Given these parametrizations, we can reformulate the original objective function as:

$$\sum_{i=0}^{N-1} \int_{\tau_i}^{\tau_{i+1}} l(x_i, u_i) dt + l_f(x_N)$$

which can be approximated as a discrete sum:

$$\sum_{i=0}^{N-1} l(x_i, u_i) + l_f(x_N)$$

where we omit the grid points, we can formulate the NLP problem as

$$\min_{\mathbf{x},\mathbf{u}} \sum_{i=0}^{N-1} l(x_i, u_i; p) + l_f(x(t_f)) \tag{2.6}$$

$$s.t \begin{cases} x_0 = \hat{x_0} \\ x_{i+1} = \phi(x_i, u_i), & i = 1, \ldots, N-1 \\ r(x_i, u_i) \leq 0, & i = 0, \ldots, N-1 \\ r_f(x_N) \leq 0 \end{cases} \tag{2.7}$$

where $\mathbf{x} = [x_0, \ldots, x_N]$ and $\mathbf{u} = [u_0, \ldots, u_{N-1}]$ are the discretized state and control variables. In the case of Non-Linear MPC, at every instant k of the control task, a similar NLP problem is formulated over the prediction horizon and the initial optimal input u_0^* of the control sequence is applied. However, finding the optimal solution of the NLP is not an easy task, mainly when we want real-time solutions in highly non-convex problems. We have to resort to another approximation within the NLP formulation to convexify the problem at hand. This results in a multi-hierarchy scheme[4], where fast algorithms have to be parallelized or find clever ways to avoid repeating computations, one of those is Sequential Quadratic Programming.

2.2.2 Sequential Quadratic Programming

To solve the original NLP problem, we use a local quadratic approximation of the original objective function and linearized constraints at each stage. In general, a NLP problem has the following form:

$$\min_{\mathbf{z}} \quad a(\mathbf{z}) \tag{2.8}$$

$$s.t \quad b(\mathbf{z}) = 0 \tag{2.9}$$

$$c(\mathbf{z}) \leq 0 \tag{2.10}$$

where $\mathbf{z} \in \mathbb{R}^{n_z}$ are the decision variables, $a(\mathbf{z}) : \mathbb{R}^{n_z} \to \mathbb{R}$ is the objective function, $b(\mathbf{z}) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_b}$ and $c(\mathbf{z}) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_c}$ are the equality and inequality constraints respectively. Given the non-convexity of the problem, it is hard to find global solutions and we rely only on local optimality. Still, many arguments can be made such that we can analyze those local optimal points in a similar fashion for global solutions. Those solutions are based on the minimization of the Lagrangian Functional:

$$\mathcal{L}(\mathbf{z}, \lambda, \mu) = a(\mathbf{z}) + \lambda^T b(\mathbf{z}) + \mu^T c(\mathbf{z})$$

Admissible solutions have to satisfy the celebrated Karush-Kuhn-Tucker conditions, assume that $\mathbf{z}^*$ is a local minimizer for NLP problem then there exist Lagrange multipliers λ^* and μ^* such that:

$$\nabla_{\mathbf{z}}\mathcal{L} : \qquad \nabla_z a(\mathbf{z}^*) + \nabla_z b(\mathbf{z}^*)^T \lambda^* + \nabla_z c(\mathbf{z}^*)^T \mu^* = 0 \tag{2.11}$$

$$\nabla_{\lambda}\mathcal{L} : \qquad b(\mathbf{z}^*) = 0 \tag{2.12}$$

$$\nabla_{\mu}\mathcal{L} : \qquad c(\mathbf{z}^*) \leq 0 \tag{2.13}$$

$$\mu_i^* c_i(\mathbf{z}^*) = 0 \quad i = 1, \ldots, n_c \tag{2.14}$$

A primal-dual solution $(\mathbf{z}^*, \lambda^*, \mu^*)$ is called a KKT point, more details on the conditions for a local minimizer can be found [20]. Assume to have an initial guess $(\mathbf{z}_i^*, \lambda_i^*, \mu_i^*)$ at iteration i, it's possible to linearize the NLP problem to find a primal increment $\Delta\mathbf{z} = \mathbf{z} - \mathbf{z}_i^*$ from solving the following Quadratic Programming

(QP) subproblem:

$$\min_{\Delta \mathbf{z}} \quad \frac{1}{2}\Delta \mathbf{z}^T H \Delta \mathbf{z} + \nabla_{\mathbf{z}} a(z^*)^T \Delta \mathbf{z} \tag{2.15}$$

$$\text{s.t.} \quad b(z^*) + \nabla_{\mathbf{z}} b(z^*)^T \Delta \mathbf{z} = 0 \tag{2.16}$$

$$c(z^*) + \nabla_{\mathbf{z}} c(z^*)^T \Delta \mathbf{z} \leq 0 \tag{2.17}$$

where $H = \nabla^2_{\mathbf{z}} \mathcal{L}(z_i^*, \lambda_i^*, \mu_i^*)$ denotes the Hessian of the Lagrangian. The solution of the subproblem gives the optimal increment $\Delta \mathbf{z}^*$ and new multipliers λ_{i+1}, μ_{i+1}, resulting in the update:

$$\begin{aligned}
\mathbf{z}^*_{i+1} &= \mathbf{z}^*_i + \alpha \Delta \mathbf{z}^* \\
\lambda^*_{i+1} &= (1-\alpha)\lambda^*_i + \alpha \lambda_{i+1} \\
\mu^*_{i+1} &= (1-\alpha)\mu^*_i + \alpha \mu_{i+1}
\end{aligned}$$

where α is the step-size, this procedure is iterated until the KKT conditions are satisfied up to a given accuracy. There are many other schemes to obtain solutions of the NLP, based on different strategies of posing the problem or parametrization, which are out of the scope of the thesis.

2.2.3 Solving the Optimal Control Problem

Given this brief introduction to NLP, we can apply it directly into an optimal control problem. The classical discretized objective function for control tasks has the form

$$\frac{1}{2}\sum_{k=0}^{N-1} \|h(x_k, u_i)\|^2_W + \frac{1}{2}\|h_N(x_N)\|^2_{W_N} \tag{2.18}$$

where h is usually a non-linear function, it can embed tracking or regulation tasks. Recall the discretize OCP (2.7), given an initial state and control trajectory guess, defining $\mathbf{z} = [z_0, \ldots, z_{N-1}, x_N]^T \in \mathbb{R}^{n_z}$ where $z_i = [x_i^T, u_i^T]^T \in \mathbb{R}^{n_x + n_u}$ and the discretized constraints:

$$b(\mathbf{z}) = \begin{bmatrix} x_0 - \hat{x}_0 \\ x_1 - \phi(x_0, u_0) \\ \vdots \\ x_N - \phi(x_{N-1}, u_{N-1}) \end{bmatrix} \quad c(\mathbf{z}) = \begin{bmatrix} r(x_0, u_0) \\ \vdots \\ r(x_{N-1}, u_{N-1}) \\ r_N(x_N) \end{bmatrix}$$

where the inequaly constraints have the form

$$\begin{aligned}
\underline{r}^l \leq r(x_i, u_i) \leq \overline{r}^l \\
\underline{r}^l_N \leq r_N(x_N) \leq \overline{r_N}^l
\end{aligned}$$

we can setup the usual NLP. At the l iterate of the SQP, given the state and control trajectory $\mathbf{z}^l$ guess, we have the following QP subproblem

$$\begin{aligned}
\min_{\Delta\mathbf{x},\Delta\mathbf{u}} \quad & \sum_{i=0}^{N-1} \Big(\frac{1}{2} \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix}^T H_i^l \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix} + g_i^{l^T} \Delta x_i \Big) \\
& + \frac{1}{2} \Delta x_N^T H_N^l \Delta x_N + g_N^{l^T} \Delta x_N \\
\text{s.t} \quad & \Delta x_0 = x_0 - \hat{x}_0 \\
& \Delta x_{i+1} = A_i^l \Delta x_i + B_i^l \Delta u_i + a_i^l, \quad i = 0, \dots, N-1 \\
& \underline{c}_i^l \leq c_i^l + C_i^{l^T} \Delta x_i + D_i^{l^T} \Delta u_i \leq \overline{c}_i^l, \quad i = 0, \dots, N-1 \\
& \underline{c}_N^l \leq c_N^l + C_N^{l^T} \Delta x_N \leq \overline{c}_N^l
\end{aligned} \tag{2.19}$$

where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^l$, $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}^l$, defining with $\mathbf{x}^l = [x_0^l, \dots, x_N^l]$ and $\mathbf{u}^l = [u_0^l, \dots, u_{N-1}^l]$. The respective Hessian matrices H_i,H_N are approximated via Gauss-Newton method and g_i,g_N are first-order terms of the Lagrangian. The linearization matrices are

$$\begin{aligned}
& A_i^l = \frac{\partial \phi}{\partial x_i}, \quad B_i^l = \frac{\partial \phi}{\partial u_i}, \quad a_i^l = \phi(x_i, u_i) - x_{i+1}^l \\
& C_i^l = \frac{\partial r}{\partial x_i}, \quad D_i^l = \frac{\partial r}{\partial u_i}, \quad C_N^l = \frac{\partial r_N}{\partial x_N} \\
& \underline{c}_i^l = \underline{r}_i^l - r(x_i, u_i), \quad \overline{c}_i^l = \overline{r}_i^l - r(x_i, u_i) \\
& \underline{c}_N^l = \underline{r}_N^l - r_N(x_N), \quad \overline{c}_N^l = \overline{r}_N^l - r_N(x_N)
\end{aligned} \tag{2.20}$$

From the solution of the QP, we can update the state and control trajectory:

$$\begin{aligned}
\mathbf{x}^{l+1} &= \mathbf{x}^l + \alpha \Delta \mathbf{x}^l \\
\mathbf{u}^{l+1} &= \mathbf{u}^l + \alpha \Delta \mathbf{u}^l
\end{aligned} \tag{2.21}$$

and iterate until the KKT conditions are satisfied, we apply the first control input of the converged solution and shift the trajectory for the next sampling instant.

2.2.4 Parametric uncertainties on the NMPC

Given the previous formulation, we focus on our benchmark problem. Consider the inverted pendulum which has the following nonlinear dynamics

$$\begin{aligned}
\ddot{p} &= \frac{-ml \sin(\theta)\dot{\theta}^2 + mg \cos(\theta) \sin(\theta) + F}{M + m - m \cos^2(\theta)} \\
\ddot{\theta} &= \frac{F \cos(\theta) - ml \cos(\theta) \sin(\theta)\dot{\theta}^2 + (m + M) g \sin(\theta)}{l(M + m - m \cos^2(\theta))}
\end{aligned}$$

where p is the cart's position, θ is the pendulum's angular position and F is the control force, depicted in Figure 2.2. From now on, each simulation will have as correct parameters the ones in Table 2.2.4.

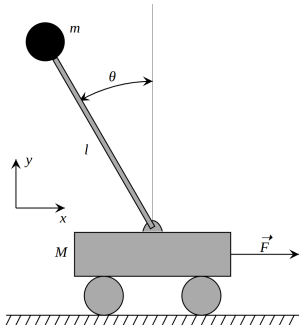


Figure 2.2: Cart-pendulum system where M is the cart mass, m and l are the pendulum mass and angular position.

Parameter	Value
M	0.1 kg
m	1 kg
l	0.5 m
g	9.81 m/s^2

Table 2.1: Correct parameters of the cart-pendulum system.

The state variables will be $x = [p, \theta, v, \omega]^T$, the control input will be $u = F$, we have the discretize OCP

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{i=0}^{N-1} \left\| \begin{bmatrix} x_i \\ u_i \end{bmatrix} - \begin{bmatrix} x_{ref} \\ u_{ref} \end{bmatrix} \right\|_Q^2 + x_N^T Q_N x_N \tag{2.22}$$

$$s.t \quad x_0 = \hat{x}_0 \tag{2.23}$$

$$x_{i+1} = \phi(x_i, u_i) \quad \forall i = 0, \ldots, N-1 \tag{2.24}$$

$$-2 \leq p_i \leq +2 \quad \forall i = 0, \ldots, N-1 \tag{2.25}$$

$$-50 \leq u_i \leq +50 \quad \forall i = 0, \ldots, N-1 \tag{2.26}$$

where $Q = diag[10, 10, 0.1, 0.1, 0.01]$ and $Q_N = diag[10, 10, 0.1, 0.1]$ are the weight matrices, $\hat{x}_0 = [0, \pi, 0, 0]^T$ is the initial state of the system, we solve this problem using SQP. While solving the QP problem, the main cause of the prediction error comes from the parametric error, this a static error present in the non-linear nominal model. Our goal is to compensate this error from the measurements of the variables $\dot{p}$ and $\dot{\theta}$ in the discretized model that comes from the integration operator $\phi(x_k, u_k)$. We will have two models to evaluate, based on the knowledge of the mathematical description we will compensate the predictions using:

- **Grey-Box**: Mathematical description is available, we will define the mismatch model on velocity prediction errors: $\psi(x_k, u_k) = \dot{q}_{k+1} - \hat{\dot{q}}_{k+1}$, where $\dot{q}_{k+1}$ are the measured velocities of the sytem (i.e. cart velocity or pendulum angular velocity) while $\hat{\dot{q}}_{k+1}$ is our velocity prediction based on the nominal model.

- **Black-Box**: No mathematical description is available, we will define the missmatch model on velocity increments: $\psi(x_k, u_k) = \dot{q}_{k+1} - \dot{q}_k$. In this case $\dot{q}_{k+1}$ and $\dot{q}_k$ are velocity measured values.

In particular, these models will be implemented by the means of a guassian process -pun intented- trained to learn the mismatch model, with the hope of compensating these parametric uncertainties.

Chapter 3

Gaussian Processes for System Identification

The goal of this chapter is to introduce the general framework of system identification of non-liner dynamical systems using Gaussian processes. There's a vast literature and many approaches have been proposed in the past decades, from neural networks to fuzzy logic models, where one the main ideas is the application of a set of basis functions[23], which are either local or global, that approximate the underlying system. Recall that any model about the physical system is wrong and we are not trying to infer the true model, instead we want to find useful models that are able to complete the task in different working conditions of the physical system. This paradigm was introduced by Ljung [13], which put on the shelf different kinds of black-box models for linear systems and evaluated their performance by minimizing some penalty function (i.e. the quadratic prediction error). Such unifiying framework for linear system identification remains valid for non-linear systems. Nonetheless, it is useful to give preliminaries concepts on linear system and later apply the key ideas to non-linear models. We will focus on the Bayesian framework, introducing Gaussian processes and their main properties. Moreover, we will give their connection with the Reproducing Kernel Hilbert Spaces, making the application of Gaussian processes for dynamical systems theoretically sounding. Later, we will introduce the implementation of the gaussian process into the NMPC scheme, showing the reformulation of the discretized optimal control problem.

3.1 Bayesian framework for System Identification

The main goal of system identification is to learn models from observed data, which allows for prediction of properties or behaviors of the phenomena at hand. Most models result in a mathematical expression, which describe relationships between variables present in the system. There could a huge amount of variables which play a role in the observed output and different models of the same model

class could describe arbitrary well the data, raising the necessity to select the best model by using some criterion. There should be also a measure of the complexity of the model classes, their size or flexibility. Here we introduce the basic example for linear systems using the Bayesian approach to inference, remarking the key concepts that will be also applied for non-linear cases.

3.1.1 The Linear Case: Some key concepts

Assume to have N datapoints $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, where $x \in \mathbb{R}^d$ is the measured input (or feature) and $y_i \in \mathbb{R}$ is the measured output. A bayesian measure model is defined as:

$$y_i = \mathbf{x}_i^T \mathbf{w} + \epsilon_i \tag{3.1}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the regressor, modelled as random variables with probability density $p(\mathbf{w})$. Output measurements are always affected by imprecisions which are unpredictable, modelled usually as $\epsilon \sim \mathcal{N}(0, \sigma_0^2)$ and assumed independently and identically distributed (i.i.d) for all the measurements. Our goal is to infer values of the regressor that best describe the dataset $\mathcal{D}$ under some criterium. In the Bayesian framework we don't have a punctual estimation but rather a probability density from which we infer some quantities. In the spirit of introducing the methodology followed in the thesis, for now we pursue the Maximum Likelihood approach to obtain a punctual estimate of the regressor. Assume that the model parameters are normally distributed $\mathbf{w} \sim \mathcal{N}(0, \Sigma)$. By stacking measurements, we obtain the more general regression model:

$$\mathbf{y} = \Phi \mathbf{w} + \boldsymbol{\epsilon} \tag{3.2}$$

where $\Phi = [\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T]^T \in \mathbb{R}^{N \times d}$, $\epsilon \sim \mathcal{N}(0, \sigma_0^2 \mathbf{I}_N)$ and we have that

$$\mathbf{y} \sim \mathcal{N}(0, \Phi \Sigma \Phi^T + \sigma_0^2 \mathbf{I}_N)$$

Consider now the likelihood function:

$$\begin{aligned} L : \mathbb{R}^d &\longrightarrow \mathbb{R}_+ \\ \mathbf{w} &\longmapsto L(\mathbf{w}; y) = p(\mathbf{w}, y) \end{aligned}$$

where y is assumed fixed and the variable $\mathbf{w}$ can change, we find the Maximum Likelihood estimate as:

$$\hat{\mathbf{w}} = \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmax}} \, L(\mathbf{w}, y) \tag{3.3}$$

Equivalently, we can define the negative log-likelihood:

$$l(\mathbf{w}; y) = -\log(p(\mathbf{w}; y)) \tag{3.4}$$

and set the optimization problem as:

$$\hat{\mathbf{w}} = \underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} \, l(\mathbf{w}, y) \tag{3.5}$$

The joint distribution can be computed as: $p(\mathbf{w}, y) = p(y|w; \sigma_0^2)p(\mathbf{w})$, where the conditional is the model error distribution. We obtain:

$$l(\mathbf{w}, y) = -\log(p(y|w; \sigma_0^2)) - \log(p(\mathbf{w}))$$

where:

$$p(y|w; \sigma_0^2) \sim \mathcal{N}(\Phi\mathbf{w}, \sigma_0^2\mathbf{I}_N)$$
$$p(\mathbf{w}) \sim \mathcal{N}(0, \Sigma)$$

obtaining the final optimization problem:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2\sigma_0^2}\|\mathbf{y} - \Phi\mathbf{w}\|^2 + \frac{1}{2}\mathbf{w}^T\Sigma^{-1}\mathbf{w} + C \tag{3.6}$$

where C is a constant. This is a simple quadratic optimization problem, and admits solution:

$$\hat{\mathbf{w}}_{ML}(\mathbf{y}) = \Phi(\Phi^T\Phi + \sigma_0^2\Sigma^{-1})^{-1}\Phi^T\mathbf{y} \tag{3.7}$$

Notice how the objective function can be decomposed in two terms: the first term is the data-fit and the second term is the complexity penalty. This decomposition generates a natural trade-off between interpretability of the data and flexibility of the model, something that will be recurrent throughout the thesis. If we'd like to make predictions using a new test point $\mathbf{x}^*$, we can simply use our punctual estimate:

$$\hat{y}(\mathbf{x}_*) = \mathbf{x}_*^T\hat{\mathbf{w}}_{ML} = \mathbf{x}_*^T\Phi(\Phi^T\Phi + \sigma_0^2\Sigma^{-1})^{-1}\Phi^T\mathbf{y}$$

We can find an alternative expression for our estimate by appying the inversion lemma to the inverse matrix:

$$\hat{\mathbf{w}}_{ML} = \Sigma\Phi^T(\Phi\Sigma\Phi^T + \sigma_0^2 I_N)^{-1}\mathbf{y} \tag{3.8}$$

One key aspect of the Bayesian framework is that it gives us confidence on our estimates. In general, given the probabilistic nature of our model, we can apply the Bayes' formula to find the parameters $\mathbf{w}$ posterior distribution:

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|\mathbf{w}, X)p(\mathbf{w})}{p(\mathbf{y}|X)} \tag{3.9}$$

where

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{w}, X)p(\mathbf{w})d\mathbf{w}$$

is a normalization constant. It is known that $p(\mathbf{w}|\mathbf{y}, X)$ is still a gaussian density, with mean and variance:

$$\mathbb{E}[\mathbf{w}|\mathbf{y}] = \Sigma\Phi(\Phi\Sigma\Phi^T + \sigma_0^2 I_N)^{-1}\mathbf{y} \tag{3.10}$$
$$\operatorname{var}(\mathbf{w}|\mathbf{y}) = \Phi\Sigma\Phi^T - \Sigma\Phi(\Phi\Sigma\Phi^T + \sigma_0^2 I_N)^{-1}\Phi^T\Sigma) \tag{3.11}$$

where the mean of the posterior has the same expression of the ML estimate. This is a known result, in case of Gaussian densities the maximum likelihood estimate coincides with the mean of the posterior density of the regressor, i.e. the maximum a posteriori estimate (MAP):

$$\mathbb{E}[\mathbf{w}|\mathbf{y}] = \hat{\mathbf{w}}_{ML}$$

and the variance is the remaining uncertainty on the parameters given that we've observed the dataset $\mathcal{D}$.

3.1.2 Non-Parametric Bayesian Regression

Following the linear regression case, we can generalize our measurement model to non-linear model as:

$$y(x) = g(x;\theta) + \epsilon \tag{3.12}$$

where $g(x;\theta)$ is a non-linear mapping, $x \in \mathbb{R}^d$ and θ are some parameters that characterize such function. Non-linear identification is based on trying to infere $g(x;\theta)$ from data, which can be extremely challenging. It is possible to apply some approximations and redefine the problem as a linear regression. We can get to this general result by first defining a non-linear mapping, called basis function:

$$\begin{aligned} \phi_\gamma : \mathbb{R}^d &\longmapsto \mathbb{R}^k \\ x &\longmapsto \phi_\gamma(x) \end{aligned}$$

where γ are parameters we can modify. Then, we can set the measurement model as:

$$y(x) = \phi_\gamma(x)^T\mathbf{w} + \epsilon \tag{3.13}$$

where $w \in \mathbb{R}^k$ is a random vector and ϵ is white noise. We can set again $\Phi(X) = [\phi(x_1), \ldots, \phi(x_N)]^T$, and obtain the linear regression model:

$$Y = \Phi(X)\mathbf{w} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_0^2 I_N)$$

This is the one of the most general black-box models[12], think of neural networks. Choosing the basis function represents the most challenging part, it has to be set a priori based on the knowlegde of the underlying system together with the parameters γ that best describe the data. Assuming again $\mathbf{w}$ as normally distributed $\mathcal{N}(0, \Sigma)$, we can apply the ML estimator to obtain punctual estimates:

$$\hat{\mathbf{w}} = \Sigma\Phi(X)^T\Big(\Phi(X)\Sigma\Phi^T(X) + \sigma_0^2 I_n\Big)^{-1}\mathbf{y} \tag{3.14}$$

In particular, we can make a prediction based on a new test input x_*:

$$y(x_*) = \phi(x_*)^T\hat{\mathbf{w}} \tag{3.15}$$

$$= \phi(x_*)^T\Sigma\Phi^T(X)\Big(\Phi(X)\Sigma\Phi^T(X) + \sigma_0^2 I_n\Big)^{-1}\mathbf{y} \tag{3.16}$$

Recall that we have a probabilistic model, encoding our priors assumption about the system. Indeed given the input x, the measurement model has a gaussian density:

$$p(y(x)) \sim \mathcal{N}(0, \phi_\gamma(x)^T \Sigma \phi_\gamma(x) + \sigma_0^2) \tag{3.17}$$

If we consider another input x', noting that the error between output measurements are independent, we obtain their covariance (correlation) as:

$$\mathbb{E}[y(x)y(x')^T] = \phi_\gamma(x)^T \Sigma \phi_\gamma(x') \tag{3.18}$$

That is, the map $\phi_\gamma(x)$ induces a covariance function in the measurement model, which directly influences our estimate and prediction (look closely at (3.14) and (3.16)). Analyzing the properties of that function, choosing the basis function ϕ_γ that best suits our needs and devicing a way to compute the (sub)-optimal parameters γ will be discussed next, where we introduce the Gaussian processes framework.

3.2 Gaussian Processes for Regression

In this section, we introduce the tool that will be our cornerstone through the thesis. It is flexibile as a function approximator, analogous to that of Neural Networks with an intimate link between both of them, and its intuitive way to construct classes of functions that represent our a priori knowledge of the model gives us a powerful tool for system identification. Most of its properties are well known in the machine learning field [25], our interest here is their capabilities as non-linear approximator. One of the main approaches for non-linear system identification is based on using a set of basis functions, which tries to infer the manifold generated by the observed inputs and variables, think of reconstructing a static image from some of its pixels. This can be generalized for Gaussian Process using the Reproducing Kernel Hilbert Spaces formalism, where these manifolds are not other that sums of linear combinations of the basis function.

3.2.1 Gaussian Processes

Before introducing gaussian processes, we need to introduce the notion of a stochastic process. This is an important mathematical tool applied in many science fields when trying to describe uncertain phenomena. Many complex physical behaviors can be described with simple models based on stochastic processes, describing its formalism and main properties is crucial to the understanding of gaussian processes.

Definition 3.2.1 (Stochastic process)**.** *Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a non-empty set, $\mathcal{F}$ is a σ-algebra of Ω and $\mathbb{P}$ is a measure function. A stochastic process is a collection of random variables $\{X_t\}$, indexed by a set T, defined on Ω:*

$$X(t, \omega) : T \times \Omega \to \mathbb{R} \tag{3.19}$$

The stochastic process is characterized by its cummulative distribution function $F(\mathbf{x})$ for a finite-dimensional collection of random variables $\{X_{t_1}, \ldots, X_{t_N}\}$, which is defined as:

$$F_{t_1,\ldots,t_k}(x_1, \ldots, x_k) \triangleq \mathbb{P}(X_{t_1} \leq x_1, \ldots, X_{t_N} \leq x_N) \tag{3.20}$$

It has to satisfy the standard conditions such as right-continuity and monoticity. Moreover, if it's absolutely continuous it admits a density function $p(\mathbf{x})$ defined as:

$$p_{t_1,\ldots,t_k}(x_1, \ldots, x_k) = \frac{\partial^k F_{t_1,\ldots,t_k}(x_1, \ldots, x_k)}{\partial x_1 \ldots \partial x_k} \tag{3.21}$$

In most cases is rather difficut to explain the behavior of the stochastic process for each realization of ω (sample path), in general we can describe its behavior with the moment functions.

Expectation and Covariance

We are interested on the mean behavior and correlation between different random variables of the process, usually referred as the first and second moments. We define the mean function $m(t)$ as:

$$m(t) = \int_\Omega X_t(\omega) d\mathbb{P}(\omega) = \int_{\mathbb{R}} x \; dF_t(x) \tag{3.22}$$

and the covariance function $C(t, s)$ as:

$$C(t,s) = \int\int_\Omega (X_t(\omega) - m(t))(X_s(\omega) - m(s)) d\mathbb{P}(\omega) \tag{3.23}$$

$$= \int\int_{\mathbb{R}^2} xy \; d^2 F_{t,s}(x, y) - m(t)m(s) \tag{3.24}$$

$$\tag{3.25}$$

We have that the variance of the random variable X_t is:

$$\sigma^2(t) = C(t, t) \tag{3.26}$$

and the correlation function with another r.v. X_s is defined as:

$$\rho(t, s) = \frac{C(t, s)}{\sigma(t)\sigma(s)} \tag{3.27}$$

We will see that mainly the study of the covariance (correlation) function is the study of the behavior of the stochastic process itself. A key property is its positive definiteness, a covariance function $C(t, s)$ is:

- **(Semi-)Positive definite**: Consider the random variable $X = \alpha_1 X_{t_1} + \cdots + \alpha_k X_{t_k}$, we have:

$$\text{Var}(X) = \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_i \alpha_j C(t_i, t_j) \geq 0, \forall \alpha_i \in \mathbb{R}, t_i \in T, k > 0 \tag{3.28}$$

  which is the definition of a semi-positive definite function.

This is a fundamental property for stochastic processes. In Abrahamsen [1], they noted that the class of covariance functions coincide with the class of positive definite functions and that the class of correlation functions coincide with the class of positive definite functions where $C(t,t) = 1$. This can be seen also as a restriction, the modelling of some phenomena with a stochastic processes must be done with a valid covariance (correlation) function. In such review, they mainly focused on the analitycal properties that a positive function must have to obtain specific behaviors of the stochastic processes. To simplify the analysis, some invariance properties must be embedded on C or ρ, a general assumption is stationarity.

Stationarity

The behavior of a stochastic process strongly depends on the cummulative distribution $F(\mathbf{x})$, which depends on the index set T. Making assumptions on the covariance functions means making assumptions on the index set, which are generally properties that such set must contain. A classic assumption is that the index set T is a linear space (i.e. $t, s \in T \Rightarrow t + s \in T$, e.g $\mathbb{R}^d$).

Definition 3.2.2. *A stochastic process is called* ***strongly stationary*** *if the finite-dimensional distributions are invariant to translations on the index set T:*

$$F_{t_1,\dots,t_k}(x_1,\dots,x_k) = F_{t_1+s,\dots,t_k+s}(x_1,\dots,x_k), \forall s \in T \tag{3.29}$$

This property is in general hardly verify. We can define a lesser condition, which acts not on the cummulative distribution but rather on the moments of the process.

Definition 3.2.3. *A stochastic process is called* ***weakly stationary*** *process if:*

$$\begin{cases} m(t) & = m \\ C(t,s) & = C(t-s) \end{cases} \tag{3.30}$$

That is, the mean is constant and the covariance between random variables depends only on the difference of the indexes. For the correlation function:

$$\rho(\tau) = \frac{C(\tau)}{\sigma_0^2} \tag{3.31}$$

with $\sigma_0^2 = C(0)$ *and* $\tau = t - s$.

We will treat interchangeably the covariance and correlation function, differing only by the scale constant σ_0^2. In most of the non-linear identification approaches, stationarity is implicitely assumed as the basis function depends only on the distance between the input variables. We follow this direction, mainly because most of the data is collected in a small region and it simplifies the training and implementation of the gaussian processes.

Differentiability

Another interesting property to us is the derivative of a stochastic process. We can define the derivative of a scalar process $X(\mathbf{t})$, with $\mathbf{t} \in \mathbb{R}^n$ as:

$$\dot{X}_i(\mathbf{t}, \omega) = \frac{\partial X(\mathbf{t})}{\partial t_i} = \lim_{\Delta \to 0} \frac{X(\mathbf{t} + \mathbf{\Delta e_i}, \omega) - X(\mathbf{t}, \omega)}{\Delta} \tag{3.32}$$

where ω is a fixed and $\mathbf{e}_i$ is a unit vector in the i-th. Under some conditions on the covariance function of the process [1], if the derivative $\dfrac{\partial^2 C(\mathbf{t}, \mathbf{s})}{\partial t_i \partial s_j}$ exists and it's finite for all $i = 1, \ldots, n$ at $(\mathbf{t}, \mathbf{t})$, the derivatives of process are mean-square continuous. That is, realizations of $\dot{X}_i(\mathbf{t})$ are generally continuous, its moments defined as:

$$\begin{cases} \mathbb{E}[\dot{X}_i(\mathbf{t})] = \dfrac{\partial m(\mathbf{t})}{\partial t_i} = \dot{m}_i(\mathbf{t}) \\ \ddot{C}_{ij}(\mathbf{t}, \mathbf{s}) = Cov(\dot{X}_i(\mathbf{t}), \dot{X}_j(\mathbf{s})) = \dfrac{\partial^2 C(\mathbf{t}, \mathbf{s})}{\partial t_i \partial s_j} \end{cases}$$

Given that we will use derivatives of gaussian processes, this property is useful because we'll have well defined linearizations within the NMPC and it will make the proposed metric (5.4) a valid one.

Gaussian processes

We are now ready to introduce the definition of Gaussian processes and its main characteristics.

Definition 3.2.4 (Gaussian process)**.** *It's a stochastic process which has a multivariate normal distribution as finite-dimensional distribution. We focus on the scalar case, for any collection of r.v.* $\{X_{t_1}, \ldots, X_{t_n}\}$*, where* $X_t \in \mathbb{R}$*, it admits density function:*

$$p(\mathbf{x}) = \frac{1}{\left(\sqrt{2\pi\, det(\Sigma)}\right)^n} exp\Big\{-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\Big\} \tag{3.33}$$

where $\mathbf{x} = [x_1, ..., x_n]^T$*,* $\mu = [m(t_1), \ldots, m(t_n)] \in \mathbb{R}^n$ *and* $\Sigma \in \mathbb{R}^{n \times n}$ *are the mean and the covariance matrix with* $\Sigma_{ij} = C(t_i, t_j)$*, defined by its two moments* $m(t)$ *and* $C(t, s)$*.*

Without loss of generality, we can consider zero-mean gaussian processes. In regression problems, the gaussian process is denoted with f and the index set is the Euclidean space $\mathbb{R}^d$. We will denote a scalar gaussian process as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'; \theta)),$$

where we use θ to indicate that the covariance function $k(x, x')$ depends on some parameters. The function k is also called the kernel function of the gaussian process, we will use interchangebly the term kernel and covariance. For a collection

of indexed random variables $\{f(x_1), \ldots, f(x_n)\}$, according to the definition of Gaussian process, we have that their joint distribution is:

$$\begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k(x_1, x_1) & \ldots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \ldots & k(x_n, x_n) \end{bmatrix}\right) \tag{3.34}$$

Some classical examples are:

- **Exponential Kernel**:

$$k_{exp}(x, x') = \sigma^2 \exp\Big(- \gamma \|x - x'\| \Big)$$

- **Squared Exponential Kernel**:

$$k_{SE}(x, x') = \sigma^2 \exp\left(- \frac{\|x - x'\|^2}{2l^2} \right)$$

- **Matérn Kernel**:

$$k_{MA}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x - x'\|}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\|x - x'\|}{l} \right)$$

  where $\Gamma(\nu)$ is the gamma function and $K_\nu(\cdot)$ is the modified bessel function of the second kind [2].

- **Polynomial Kernel**:

$$k_{pol}(x, x') = (1 + \gamma x^T x')^d$$

- **Trigonometric Kernel**:

$$k_{trig}(x, x') = \sigma^2 \cos(\omega(x - x'))$$

In Algorithm 1 we report the simple procedure to generate sample-paths of a Gaussian process with covariance $k(x, x')$. Note that these are a priori realizations, no inference has been done. As stated, there are many parameters that control the behavior of the stochastic process, in Figure 3.1 we can see properties from periodicity to smoothness, etc. Notice how also setting different parameters change the resulting function within a model class (in particular for the Matérn kernel). Therefore, we could naivily (not so much) say that setting these covariance functions means choosing a space of functions with properties inhereted from the covariance function, i.e. we are choosing an hypothesis, and within that same class there are more intrinsicate behaviors determined by the so called hyperparameters. We will later formalize this initial intuition.

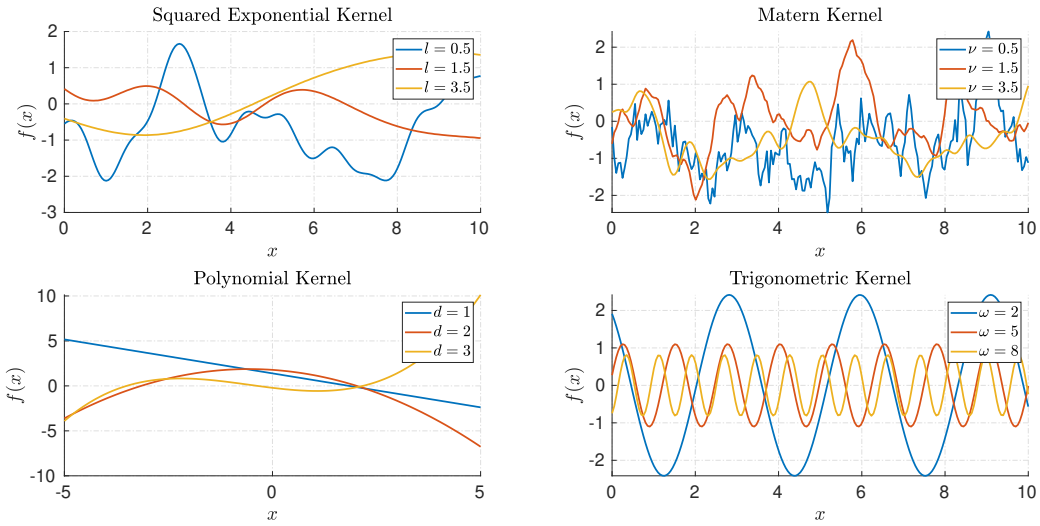


Figure 3.1: Realizations on uniform grids with SE, Matérn, Polynomial and Trigonometric kernels.

3.2.2 Regression with Gaussian processes

Now, consider that we have N datapoints $\{(x_i, y_i)\}$, we are interested in finding the relationship between input $x \in \mathbb{R}^d$ and the measured output $y \in \mathbb{R}$. We model their relation with a Gaussian process f indexed by the input variables x. The setup for measurement model is:

$$y_i(x_i) = f(x_i) + e, \text{ where } \begin{cases} f(x) \sim \mathcal{GP}(0, k(x, x'; \theta) \\ e \sim \mathcal{N}(0, \sigma_n^2 \delta_{ij}) \end{cases} \tag{3.35}$$

where e is white gaussian noise representing the uncertainty in our measurements. Collecting all the measurements, we have that:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} \tag{3.36}$$

From now on, we indicate the input collection as X and $\mathbf{e}$ the collection of measurement errors:

$$\mathbf{y} = \mathbf{f}(X) + \mathbf{e} \tag{3.37}$$

Algorithm 1 Generate sample paths on a n-points uniform-grid

procedure SAMPLEPATH($k, \{x_1, \ldots, x_n\}$)
  $[K(X, X)]_{ij} \leftarrow k(x_i, x_j)$
  $L \leftarrow$ CholeskyDecomposition$(K(X, X))$
  $\mathbf{e} \leftarrow n$ samples from $\mathcal{N}(0, 1)$
  $f \leftarrow L^T e$
end procedure

We have that $\mathbf{y}$ is a normally distributed vector with probability density:

$$\mathbf{y} \sim \mathcal{N}(0, K(X,X) + \sigma_n^2 I_N) \tag{3.38}$$

where $[K(X,X)]_{ij} = k(x_i, x_j)$. Given that we have a probabilistic model, when we have to make predictions in another index point x^*, we need to compute the conditional probability of $f(x^*)$ with respect to the available data $\mathbf{y}$, where the joint probability for $[f(x^*), \mathbf{y}^T]^T$ is available:

$$\begin{bmatrix} f(x^*) \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(x^*,x^*) & K(X,x^*) \\ K(x^*,X) & K(X,X) + \sigma_n^2 \end{bmatrix}\right) \tag{3.39}$$

It's a known result that conditioning jointly gaussian variables results in a gaussian variable:

$$f(x^*|y,X) \sim \mathcal{N}(\mu(x^*|y,X), \mathrm{var}(x^*|y,X)) \tag{3.40}$$

with

$$\begin{cases} \mu(x^*|y,X) & = K(x^*,X)\Big(K(X,X) + \sigma_n^2\Big)^{-1}\mathbf{y} \\ \mathrm{var}(x^*|y,X) & = k(x^*,x^*) - k(x^*,X)\Big(K(X,X) + \sigma_n^2\Big)^{-1}K(X,x^*) \end{cases} \tag{3.41}$$

where $K(x^*,X) = \Big[k(x^*,x_1),\ldots,k(x^*,x_n)\Big]$ and $K(X,x^*) = K(x^*,X)^T$. To make a punctual estimate of $f(x^*)$, we could apply the ML estimator, which in the Gaussian case coincides with the mean of the conditional density:

$$\hat{f}(x^*) = \mu(x^*|y,X) \tag{3.42}$$

As can be noticed, the prediction strongly depends on the choosen covariance function. Another great property of the Bayesian framework is that it allows to define the uncertainty in our predictions, but we won't use these information in the MPC scheme.

3.2.3 Learning the Hyperparameters

The quality of the regression depends on the parameters we assign to the kernel function. There are many methodoligies to carry out the selection of the hyperparameters such as cross-validation or grid-like evaluations. One of the most applied is the marginal likelihood maximization when we are dealing with continuous hyperparameters $\theta \in \mathbb{R}^n$. In general, the observed variables $\mathbf{y}$ are distributed according to:

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)d\mathbf{f} \tag{3.43}$$

where we integrate out the random functions $\mathbf{f}$ distributed according to the gaussian process, while $p(\mathbf{y}|\mathbf{f}, X)$ is the measurement error distribution. In the Gaussian case, we have a closed-form expression for the marginal likelihood:

$$\mathbf{y}|X \sim \mathcal{N}(0, K(X,X) + \sigma_n^2 I_n) \tag{3.44}$$

We can apply the *log* transform to $p(\mathbf{y}|X)$ and try to maximize it with respect to the hyperparameters:

$$\log(p(\mathbf{y}|X;\theta)) = -\frac{n}{2}\mathbf{y}^T(K_\theta(X,X) + \sigma_n^2 I_n)^{-1}\mathbf{y} - \frac{n}{2}\log(2\pi\det(K_\theta(X,X) + \sigma_n^2 I_n) \tag{3.45}$$

Notice how can we divide the objective function as a data-fit term (first-term) and a complexity-term (second-term), which makes this function suitable for regularization within the selection procedure. Define $K = K_\theta(X,X) + \sigma_n^2 I_n$, we can set the Marginal Log-Likehilihood optimization problem:

$$\theta^* = \underset{\theta \in \mathbb{R}^n}{\operatorname{argmax}} \log(p(\mathbf{y}|X;\theta))$$

This is in general a non-convex optimization problem with respect to the hyper-parameters, in some cases there are also constraints that must be satisfied (e.g. for SE kernel $l \in \mathbb{R}_+$). The objective function, under some conditions such as the differentiability of the kernel fucntion k with respect to the hyper-parameters, admits a gradient:

$$\frac{\partial \log(\mathbf{y}|X;\theta)}{\partial \theta_j} = \frac{1}{2}\mathbf{y}^T K_\theta^{-1}\frac{\partial K_\theta}{\partial \theta_j}K_\theta^{-1}\mathbf{y} - \frac{1}{2}\operatorname{tr}(K_\theta^{-1}\frac{\partial K_\theta}{\partial \theta_j}) \tag{3.46}$$

$$= \frac{1}{2}\operatorname{tr}\Big((\alpha\alpha^T - K_\theta^{-1})\frac{\partial K_\theta}{\partial \theta_j}\Big) \tag{3.47}$$

where $\alpha = K_\theta^{-1}\mathbf{y}$. This procedure is also known as Empirical Bayes or Type-2 ML, we can apply gradient descent algortihms to this optimization problem to find the hyper-parameters. Given the non-covexity of the objective function, the solutions may be locally stationary and different initializations for the hyper-parameters may lead to drastical different solutions. With a relative low number of parameters this phenomena can be disregarded, but once we start constructing more complex kernels and the dimensionality of the hyper-parameters grows, some solutions can lead to overfitting.

3.2.4 Reproducing Kernel Hilbert Spaces

We denoted earlier a Gaussian Process as a function $f(\cdot)$ taking as input $x \in \mathbb{R}^d$. Indeed, any gaussian process can be seen as assigning a probability to a function where its measure is given by the infinite-dimensional distribution. In the Figure 3.1, we used the different covariance functions to see how the behavior

of the associated Gaussian process changed. Assume that we have a set of basis functions $\{\phi_d(x)\}_{d=1}^n$ and we construct a function as:

$$f(x) = \sum_{d=1}^{n} w_d \phi_d(x)$$

where the random variables $w_d \sim \mathcal{N}(0, \sigma^2)$ are independent between each other. If we compute the covariance between different inputs:

$$\begin{aligned}\mathbb{E}[f(x)f(x')] &= \mathbb{E}\Big[\big(\sum_{d=1}^{n} w_d\phi_d(x)\big)\big(\sum_{d=1}^{n} w_d\phi_d(x')\big)\Big] \\ &= \sigma^2 \sum_{d=1}^{n} \phi_d(x)\phi_d(x')\end{aligned}$$

This is a covariance function constructed from the set of basis functions. Any realization of $(w_1, \ldots, w_n)$ will assign a linear combination of the set of basis functions, hence we can assign a probability to any linear combination of such functions based on the joint distribution of $(w_1, \ldots, w_n)$. Of course, in rare cases we precisely know such functions for the regression problem and we rely on the kernel function where there's no trace of a set of basis functions. Moreover, we could pose the question of what happens if we impose an infinite number of basis functions weighted by the random variables, does it admit a kernel representation? Or viceversa, if any fixed kernel admits a set of basis functions weighted by some random variables. The answer is positive and comes from Mercer's Theorem, but to understand it we have to introduce the notion of Reproducing Kernel Hilbert Spaces (RKHS) which gives us a powerful tool for regression problems [8].

Definition 3.2.5 (RKHS)**.** *Let $\mathcal{X}$ be a non-empty set and $k(\cdot,\cdot)$ a positive definite kernel function defined on $\mathcal{X}$. A Hilbert space $\mathcal{H}_k$ of functions defined on $\mathcal{X}$ with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}_k}$ is called a reproducing kernel Hilbert space (RKHS) with kernel k if it satisfies the properties:*

1. *For all $x \in \mathcal{X}$, the kernel section $k(\cdot, x) \in \mathcal{H}_k$.*
2. *For all $x \in \mathcal{H}_k$ and for all $f \in \mathcal{H}_k$:*

   $f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{H}_k}$ *(Reproducing Property or Evaluation operator)*

From this general definition, we'd like to characterize the relationship between a kernel $k(x, x')$ and a Hilbert space $\mathcal{H}_k$. Indeed, there is a one-to-one relationship between them, as stated by Moore-Aronszajn theorem:

Theorem 3.2.1. *Moore-Aronszajn: For a positive definite kernel k there exists a unique Hilbert space $\mathcal{H}_k$ such that k is the reproducing kernel, and viceversa.*

Now, for purpose of understanding better what is the role of k, we introduce the formal construction of a reproducing kernel Hilbert space. Consider a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ positive-definite. We can define the space of linear combinations:

$$\mathcal{H}_0 = span\big\{k(\cdot, x) : x \in \mathcal{X}\big\} = \Big\{f = \sum_{i=1}^{n} a_i k(\cdot, x_i) : n \in \mathbb{N}, a_i \in \mathbb{R}, x_i \in \mathcal{X}, i = 1, \cdots, n\Big\}$$

Moreover, we can define as inner-product between $f = \sum_{i=1}^{n} a_i k(\cdot, x_i)$ and $g = \sum_{j=1}^{m} b_j k(\cdot, y_j)$:

$$\langle f, g\rangle_{\mathcal{H}_0} \coloneqq \sum_{i=1}^{n}\sum_{j=1}^{m} a_i b_j k(x_i, y_j)$$

which satisfies all properties of an inner-product by the positive-definiteness of k. The induced norm is $\|f\|^2_{\mathcal{H}_0} = \langle f, f\rangle_{\mathcal{H}_0}$ and we can obtain the Reproducing Kernel Hilbert Space of k by taking the closure of $\mathcal{H}_0$:

$$\mathcal{H}_k = \overline{span}\mathcal{H}_0 \coloneqq \Big\{f = \sum_{i=1}^{\infty} a_i k(\cdot, x_i) : (a_1, \dots) \subset \mathbb{R}, (x_1, \dots) \subset \mathcal{X}, \text{ such that}$$
$$\|f\|^2_{\mathcal{H}_k} \coloneqq \lim_{n\to\infty} \Big\| \sum_{i=1}^{n} a_i k(\cdot, x_i)\Big\|^2_{\mathcal{H}_0} = \sum_{i,j}^{\infty} a_i a_j k(x_i, x_j) < \infty\Big\}$$

From this construction, it can be noticed that properties of any function $f \in \mathcal{H}_k$ are inhereted from the properties of k. That is, when we set a covariance function on the gaussian process f, we are implicitly making assumptions on the space on which realizations may belong.
Now, let μ be a finite Borel measure defined on $\mathcal{X}$ with $\mathcal{X}$ being its support (e.g. Lebesgue measure). Let $L_2(\mu)$ be the Hilbert space of square-integrable functions with respect to μ. We can define the integral operator with k defined as before, we have:

$$T_k f \coloneqq \int k(\cdot, x) f(x) d\mu(x), \;\; f \in L_2(\mu)$$

This is a self-adjoint, positive and compact linear operator with a countable system of non-negative eigenvalues $\{\lambda_k\}_{k=1}^{\infty}$ such that $\sum_{i=1}^{\infty} \lambda_i^2 < +\infty$. This implies that there exists an eigen-decomposition $\{\phi_i\}_{i=1}^{+\infty}$ of T_k such that:

$$T_k f = \sum_{i=1}^{+\infty} \lambda_i \langle \phi_i, f\rangle_{L_2(\mu)} \phi_i$$

The eigenfunctions $\{\phi_i\}_{i=1}^{+\infty}$ form an orthonormal system in $L_2(\mu)$, i.e. $\langle \phi_i, \phi_j\rangle_{L_2(\mu)} = \delta_{ij}$. Mercer's Theorem says that the kernel function can be described by this set of eigen-pairs.

Theorem 3.2.2 (Mercer's Theorem)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$, μ a positive Borel measure on $\mathcal{X}$, $k(\cdot,\cdot)$ a symmetric continous function defined on $X \times X$ and positive definite. Moreover, k such that:*

$$\int_{\mathcal{X}} \int_{\mathcal{X}} K(x,y)^2 d\mu(x) d\mu(y) < +\infty$$

Then

$$k(x,x') = \sum_{i=1}^{+\infty} \lambda_i \phi_i(x) \phi_i(x')$$

where the series converge absolutely and uniform over $x, x' \in \mathcal{X}$.

This means that for each absolutely integrable kernel we choose, there's a basis function from which we could carry a Bayesian linear regression where the weights are associated to a measure μ. This representation, although abstract, gives us the intuition that we could construct any kernel if we were to assign the correct measure, in particular for stationary kernels as we will see in the next chapter. In a similar fashion to the gaussian regression, we could ask if given N datapoints $\{(f_i, x_i)\}_{i=1}^N$, there exists a function in $\mathcal{H}_k$ that minimizes:

$$\hat{f} = \underset{f \in \mathcal{H}_k}{\operatorname{argmin}} \sum_{i=1}^{N} (f_i - f(x))^2 + \gamma \|f\|_{\mathcal{H}_k}^2 \tag{3.48}$$

where $\mathcal{H}_k$ has kernel k. The answer is positive, and intimately linked to gaussian regression.

Theorem 3.2.3 (The Representer Theorem)**.** *If $\mathcal{H}_k$ is a RKHS, the minimizer of 3.48 has the form:*

$$\hat{f}(\cdot) = \sum_{i=1}^{N} \alpha_i k(\cdot, x_i)$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ are given by

$$\boldsymbol{\alpha} = (K(X,X) + \gamma \mathbf{I}_n)^{-1} \mathbf{f} \tag{3.49}$$

where $\mathbf{f} = [f_1, \ldots, f_2]^T$ and $K(X,X)$ is the Gram matrix given by $k(x_i, x_j)$.

Notice the familiarity of the solution, indeed it coincides with the mean of the posterior of a GP. Now we know that choosing the kernel means choosing a set of basis functions, and the prediction depends again on such basis functions. Therefore, it is reasonable to try to approximate in some way the basis functions to improve the overall performance of the GP in the regression task.

3.3 Gaussian Processes in the NMPC

In this section we describe the implementation of gaussian processes in the NMPC scheme. We'll give a brief introduction on the training procedure, from collecting the data and setting up the optimal control problem. The intent of implementing the gaussian process is to achieve better predictions, in particular during the generation of the QP subproblem which relies on the accuracy of the nominal model, leading to a better closed-loop behavior.

3.3.1 Training the Gaussian process

The training of a gaussian process is a straightforward procedure. Collect data from the system, model the input variables that will be form the feature space and the relative variable to predict. For dynamical systems, the feature space usually relates to past-present inputs and state variables, and the quantity to model with the gaussian process is generally some partially observed state variable. In our case, the input for the GP are physical variables that come from the same instant k and its output is the relative prediction error at the next instant $k+1$. This is a general approach, recall again the discretized model that comes from the integration operator:

$$\mathbf{x}_{k+1} = \phi(\mathbf{x}_k, \mathbf{u}_k)$$

where we can divide the state variable as $\mathbf{x}_k = [\mathbf{q}_k, \dot{\mathbf{q}}_k]^T$ where $\mathbf{q}_k \in \mathbb{R}^{n_q}$ and $\dot{\mathbf{q}}_k \in \mathbb{R}^{n_q}$ are the positions and velocities of the physical system. We assume that we can observe the state variable, so we don't deal with state estimation.

Collecting the data

Given N time instants, we can group the observed states and inputs:

$$\mathcal{X} = \{\mathbf{x}_0, \dots, \mathbf{x}_{N-1}\} \qquad \mathcal{U} = \{\mathbf{u}_0, \dots, \mathbf{x}_{N-1}\}$$
$$\dot{\mathcal{Q}} = \{\dot{\mathbf{q}}_1, \dots, \dot{\mathbf{q}}_N\} \qquad \hat{\dot{\mathcal{Q}}} = \{\hat{\dot{\mathbf{q}}}_1, \dots, \hat{\dot{\mathbf{q}}}_N\}$$

where $\hat{\dot{\mathbf{q}}}$ are the predicted velocities by the nominal model and $\dot{\mathbf{q}}$ are the actual measured velocities. The mismatch model will be defined on the velocities, depending on the kind of model setup, we define it as

$$\begin{aligned} \text{Grey-Box}: &\quad \Delta\dot{\mathbf{q}}_k = \dot{\mathbf{q}}_{k+1} - \hat{\dot{\mathbf{q}}}_{k+1} && \text{(Velocity prediction errors)} \\ \text{Black-Box}: &\quad \Delta\dot{\mathbf{q}}_k = \dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k && \text{(Velocity increments)} \end{aligned}$$

Each mismatch velocity component will be modelled independently as a gaussian process. Defining the index points as $\tilde{\mathbf{x}} = [\mathbf{x}^T, \mathbf{u}^T]^T$, we define the relative gaussian process for the i-th mismatch velocity

$$\Delta\dot{q}^i \coloneqq \hat{\psi}^i_{\dot{q}}(\tilde{\mathbf{x}}) \sim \mathcal{GP}(0, k^i(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'))$$

where $k^i(\cdot,\cdot)$ is its kernel function. We obtain the measurement model

$$\mathbf{y}^i = \begin{bmatrix} \Delta\dot{q}_1^i \\ \vdots \\ \Delta\dot{q}_N^i \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} = \Delta\dot{\mathbf{q}}^i + \mathbf{e}$$

where $\mathbf{e} \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I}_N)$ is the measurement noise with variance σ_n^2. The dataset doesn't need to be ordered in a sequential manner, it's enough to have the pairs $(\tilde{\mathbf{x}}_k, \mathbf{y}_k^i)$, rather it is heavily recommended to have a shuffled dataset. This is because, from the manifold view-point, we are completing the velocity models of the system and the prediction model isn't based on causality.

Learning procedure

For each GP, we will have the dataset $\mathcal{D} = \{(\tilde{\mathbf{x}}_j, \mathbf{y}_j^i)\}_{j=1}^N$. Each kernel will have its own hyper-parameters $\theta \in \mathbb{R}^{n_\theta}$, these define the overall behavior of the process. To optimize them, we minimize the Negative Log-Likehood by gradient-descent – this is the key part of the learning procedure for GP-based models. There is another important aspect of the learning that has to be treated, focused on the generalization properties of the gaussian process. Given that we are trying to infere the model mismatch, within a control task where the nominal NMPC controller is active, our generalization capabilities could be limited within the state trajectory obtained by the sub-optimal control trajectory, i.e. the generalization properties are task-focused, or limited within a region of the operational space. A way to avoid this bias within the learning is to apply an external control excitation, this is a general procedure to explore different operational regions. Defining $\mathbf{u}_{ext}$ the external input, this leads to the dataset

$$\Delta\dot{\mathbf{q}} = \dot{\mathbf{q}}_{k+1} - \phi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_{MPC} + \mathbf{u}_{ext})$$

where $\mathbf{u}_{MPC}$ is the control input from the MPC controller, needed to establish safety and achievement of the control task. To estimulate the system, we will use sum of decaying sinusoids in different regions of the control task. We will talk more in-depth about the hyperparameters and excitation of the system in the simulation results.

Making predictions

We can use the trained process to make prediction on a new test point $\tilde{\mathbf{x}}^*$, the conditional probability density of the process $\hat{\psi}_{\dot{q}}^i(\tilde{\mathbf{x}}^*)$ given the N datapoints is still gaussian and its Maximum Likelihood estimate coincides with its mean. We obtain the closed-form expression for the prediction as:

$$\begin{aligned} \mathbb{E}[\hat{\psi}_{\dot{q}}^i(\tilde{\mathbf{x}}^*)|\mathcal{D}] \coloneqq \psi_{\dot{q}}^i(\tilde{\mathbf{x}}^*) &= \mathbf{k}^i(\tilde{\mathbf{x}}^*)\boldsymbol{\alpha}^i \\ &= \sum_{j=1}^N k^i(\tilde{\mathbf{x}}^*, \tilde{\mathbf{x}}_j)\alpha_j^i \end{aligned}$$

where $\boldsymbol{\alpha}^i = (K^i(X,X) + \sigma_n^2 \mathbf{I}_N)^{-1}\mathbf{y}^i$ and $\mathbf{k}^i(\tilde{\mathbf{x}}^*) = [k^i(\tilde{\mathbf{x}}^*, \tilde{\mathbf{x}}_1), \ldots, k^i(\tilde{\mathbf{x}}^*, \tilde{\mathbf{x}}_N)]$. Finally, we have the complete mismatch model prediction at the test point $\tilde{\mathbf{x}}^*$

$$\boldsymbol{\psi}_{\dot{q}}(\tilde{\mathbf{x}}^*) = \begin{bmatrix} \psi_{\dot{q}}^1(\tilde{\mathbf{x}}^*) \\ \vdots \\ \psi_{\dot{q}}^{n_q}(\tilde{\mathbf{x}}^*) \end{bmatrix} \tag{3.50}$$

Given the model setup, we will have different models implementation in the NMPC scheme. Under constant control input during the sampling interval T_s, we have

- **Grey-Box**:

$$\begin{aligned} \mathbf{x}_{k+1} &= \phi(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\psi}(\mathbf{x}_k, \mathbf{u}_k) \\ \begin{bmatrix} q(\mathbf{x}_k, \mathbf{u}_k) \\ \dot{q}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix} &= \begin{bmatrix} \phi_q(\mathbf{x}_k, \mathbf{u}_k) \\ \phi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix} + \begin{bmatrix} \psi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_k) \cdot \frac{T_s}{2} \\ \psi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix} \end{aligned}$$

  where ϕ is the nominal model description.

- **Black-Box**:

$$\begin{aligned} \mathbf{x}_{k+1} &= \phi(\mathbf{x}_k) + \boldsymbol{\psi}(\mathbf{x}_k, \mathbf{u}_k) \\ \begin{bmatrix} q(\mathbf{x}_k, \mathbf{u}_k) \\ \dot{q}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix} &= \begin{bmatrix} q_k + \dot{q}_k \cdot T_s \\ \dot{q}_k \end{bmatrix} + \begin{bmatrix} \psi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_k) \cdot \frac{T_s}{2} \\ \psi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix} \end{aligned}$$

  that is, our nominal velocity prediction model is the identity operator $\phi_{\dot{q}}(\mathbf{x}_k, \mathbf{u}_k) = \mathbb{I}_{n_q}$.

3.3.2 Learning-based NMPC

We are ready to implement the improved models within the NMPC scheme to solve the OCP. Recalling the discretized OCP, the QP subproblem to solve iteratively will be:

$$\min_{\Delta\mathbf{x}, \Delta\mathbf{u}} \quad \sum_{i=0}^{N-1} \Big(\frac{1}{2} \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix}^T H_i^l \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix} + g_i^{l^T} \Delta x_i \Big) \quad + \frac{1}{2} \Delta x_N^T H_N^l \Delta x_N + g_N^{l^T} \Delta x_N \tag{3.51}$$

$$\begin{aligned} \text{s.t} \quad & \Delta x_0 = x_0 - \hat{x}_0 \\ & \Delta x_{i+1} = (A_i^l + A_{i,GP}^l)\Delta x_i + (B_i^l + B_{i,GP}^l)\Delta u_i + a_{i,GP}^l, \quad i = 0, \ldots, N-1 \end{aligned} \tag{3.52}$$

$$\begin{aligned} & \underline{c}_i^l \leq c_i^l + C_i^{l^T} \Delta x_i + D_i^{l^T} \Delta u_i \leq \overline{c}_i^l, \quad i = 0, \ldots, N-1 \\ & \underline{c}_N^l \leq c_N^l + C_N^{l^T} \Delta x_N \leq \overline{c}_N^l \end{aligned}$$

where we have new linearization matrices

$$A_{i,GP}^l = \frac{\partial \psi}{\partial x_i}, \quad B_{i,GP}^l = \frac{\partial \psi}{\partial u_i}, \quad a_{i,GP}^l = \phi(x_i, u_i) + \psi(x_i, u_i) - x_{i+1}^l$$

and the other matrices are defined as in (2.20), notice that the continuity constraints now include the prediction of the GP in the term $a_{i,GP}$. Another key point to take into consideration is that the linearization matrices don't come from the actual linearization of the system but rather from the integration operator, i.e. *first-discretize-then-differentiate*. This is possible because we've implemented the optimization problem with the CasADi library and we're able to compute the derivatives with AD after the integration operator to improve computational performance, we don't digress further into that matter. These matrices can be evaluated directly within Lb-MATMPC toolbox throughout the simulation, they are of interest because their quality of linearization depends on the accuracy of the original model. If the GP model has indeed learned the mismatch model, we should get that $A_i + A_{i,gp}$ is close to the linearized matrices of the original system. This is possible because the kernel function encodes a priori information about differentiability, in paricular $\dfrac{\partial \psi}{\partial x_i}$ is a well defined quantity if we use kernels that satisfy the continuity property for the derivative process (3.32) - at least twice-differentiable at 0 ([25] Chapter 9).

Chapter 4

Generalization of Stationary Kernel Functions

In this section we introduce the family of spectral representations of kernel functions, a class of kernels flexible enough to describe our data. As we've seen before, the quality of the regression using Gaussian processes strictly depends on the selected kernel. If the phenomena that we are trying to describe is not compatible with the kernel, i.e. the analytical properties of the kernel don't resemble that of the phisycal system we are trying to infer, the generalization will be unsatisfying. An interesting idea is that of approximating the kernel function itself by adding up different kernels to give the gaussian process more flexibility which should model different behaviors of the phenomena we are trying to regress, obtaining an overall improved prediction performance – this is not a new idea by itself and it's actually a standard approach for gaussian regression. The main difficulty with such method is that the kernels must encode a priori information, therefore it's up to the operator to select which kernels to use leading to tuning the hyperparameters by trial-and-error, making the overall training procedure longer and difficult. Moreover, if we were to implement different kernels within the NMPC, the overall computational speed could increase depending on the amount of kernels added, it would go against our goals of achieving fast predictions in real-time, something the standard Squared Kernel has proved to excel. The SE kernel has this nice flexibility property while maintaining a simple form, a key aspect we would like to preserve in the implementation of gaussian process for NMPC. Here, rather than imposing different classes of functions, we introduce a general classes of kernels which from a simple basis kernel can approximate any stationary kernel function[22]. That is, this class of kernels allows us to improve the learning performance while implementing only one class of kernel and adding little computational burden to the NMPC scheme. The main theoretical underpinning is based on a Fourier decomposition for positive-definite functions, Bochner's Theorem, which has been at the center of many procedures with the goal of approximating kernel functions [21],[9],[26],[5]. Before introducing the implemented kernel, we describe the main ideas that lead up to it, with the hope of making clear why we selected it.

4.1 Spectral Representation of Kernel Functions

The main property of a kernel function is that it is positive-definite, Bochner's theorem uses this property to state an alternative representation of the kernel function:

Theorem 4.1.1 (Bochner's Theorem)**.** *A real-valued function on $\mathbb{R}^d$ is the covariance function of a weakly stationary mean square continuous real valued process defined on $\mathbb{R}^d$ if and only if it can be represented as*

$$k(x - x') = \int_{\mathbb{R}^d} e^{j\omega^T(x-x')}\mu(d\omega)$$

where μ is a positive finite measure and $x \in \mathbb{R}^d$.

This is main theoretical underpinning for many approaches that try to generalize the kernel function. For now, focus in absolutely continuous measures with respect to the Lebesgue's measure, admitting density function $S(\omega)d\omega = \mu(d\omega)$ such that:

$$k(x - x') = \int_{\mathbb{R}^d} e^{j\omega^T(x-x')}S(\omega)d\omega.$$

Setting $\tau = x - x'$, we have that $S(\omega)$ and $k(\tau)$ form a Fourier pair, by the duality of the Fourier transform, we can compute the spectral density as:

$$S(\omega) = \int_{\mathbb{R}^d} k(\tau)e^{-j\omega^T\tau}d\tau$$

Recall that we can normalize the kernel function $\hat{k}(x, x') = \frac{k(x,x')}{\sigma^2}$ where $k(0) = \sigma^2$, such that $\hat{k}(0) = 1$. This is the correlation function of the stochastic process, it can be interpreted as the expectation of the random variable $e^{j\omega^T(x-x')}$, applying Bochner's Theorem:

$$\hat{k}(x, x') = \int e^{j\omega^T(x-x')}\frac{S(\omega)}{\sigma^2}d\omega \quad \text{and} \quad \hat{k}(0) = \int \frac{S(\omega)}{\sigma^2}d\omega = 1$$
$$\Rightarrow \hat{k}(x, x) = \mathbb{E}_{\hat{S}(\omega)}\left[(e^{j\omega^T x})(e^{j\omega^T x'})^*\right]$$

where $\hat{S}(\omega) = \frac{S(\omega)}{\sigma^2}$ is a valid probability density function. Both of these representations are usually referred as the Power Spectral Density of the stochastic process, and have the following properties:

- Symmetry: $S(\omega) = S(-\omega)$
- Positivity: $S(\omega) > 0, \forall\omega \in \mathbb{R}^d$
- Normalization: If k is the correlation function, then $k(0) = \int S(\omega)d\omega = 1$

which follow from the properties of the kernel function. Approaches based on this decomposition to approximate the kernel function can be divided in probabilistic or deterministic, depending on how they interpret the power spectral density. In the following we describe their main characteristics and differences.

4.1.1 Probabilistic and Deterministic Fourier Features

The probabilistic view states that the kernel is the mean of a function of the random variable $\omega \sim S(\omega)$:

$$k(x, x') = \mathbb{E}_{\omega \sim S(\omega)}\left[\left(e^{j\omega^T x}\right)\left(e^{j\omega^T x'}\right)^*\right]$$

where $*$ denotes the conjugate. Given that we are considering only real functions, the imaginary part of the inner product can be discarded (more generally we have that, by the symmetry of $S(\omega)$, the integration of the immaginary part is equal to 0), allowing us to rewrite:

$$k(x, x') = \mathbb{E}_{\omega \sim S(\omega)}\Big[\cos(\omega^T(x - x'))\Big]$$

One of the first approaches using Fourier features[21] used this expression to approximate the resulting kernel by Monte-Carlo integration. That is, they extracted M i.i.d. samples from $S(\omega)$ (it had to be known apriori) and computed:

$$k(x, x') \approx \hat{k}(x, x') = \frac{1}{M}\sum_{i=1}^{M}\cos(\omega_i^T(x - x')), \qquad \omega_i \sim S(\omega)$$

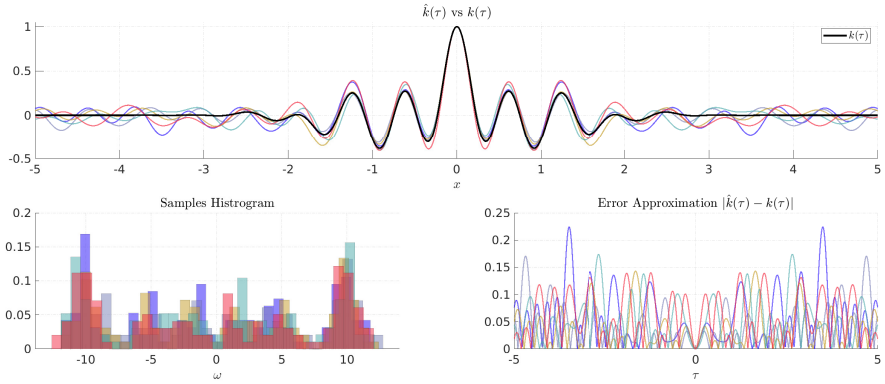
Moreover, they introduced a probabilistic bound to the error of the approximation:

Theorem 4.1.2 (Uniform convergence of Fourier Features)**.** *Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with diameter $diam(\mathcal{M})$. Then, for the mapping $\hat{k}$, we have that:*

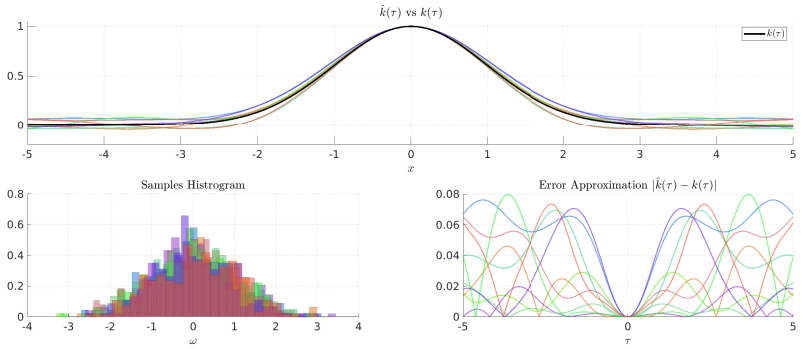
$$\boldsymbol{Pr}\left[\sup_{x,y\in\mathcal{M}}|\hat{k}(x, x') - k(x, x')| \geq \epsilon\right] = C\left(\frac{\sigma_p diam(\mathcal{M})}{\epsilon}\right)^2 exp\left(-\frac{D\epsilon^2}{4(d+2)}\right)$$

where $\sigma_p^2 \triangleq E_{S(\omega)}[\omega^T\omega]$ is the second moment of the Fourier transform of k, D is the number of samples and C is a constant.

We can see this bound in figure 4.4, where we extract M i.i.d samples from $S(\omega)$ and generate $\hat{k}(x, x')$. In this example we consider spectral densities from the SE Kernel and Genereralized Kernel with $K = 3$ clusters, which will be introduced later. Notice how the generated kernel function closely resembles the original kernel near the origin, with a relative low number of samples, intuitively this is caused by how many frequencies we extract near the peaks of the distribution, as can be seen by the histograms. This is desirable because it leads local correlation values, although the negative effect is at the tails of this approximation where we can see oscillations that can cause erroneous correlation values between distant datapoints. This approximation was mainly used to decrease the training time while preserving some accuracy within SVM regression by computing the kernel mapping from samples of $\omega \sim S(\omega)$, the spectral density remained the same so it wasn't actually infering the kernel structure. Let's now elaborate the same result



(a) Random Generalized Kernel, $K = 3$.



(b) Random SE Kernel.

Figure 4.1: Random Fourier Feature kernels using $N = 128$ samples, Samples Histogram and Error Approximation.

but in a deterministic manner, in particular we focus on the function-view of the gaussian processes. Defining with $z_i(x) = [\cos(\omega_i x), \sin(\omega_i x)]^T$, we can see that:

$$\hat{k}(x, x') = \frac{1}{M} \sum_{i=1}^{M} cos(\omega_i(x - x')) = \frac{1}{M} \sum_{i=1}^{M} z_i^T(x) z_i(x')$$

More in general, this approximation can be seen as a trigonometric bayesian regression problem where we model the gaussian process $f(x), x \in \mathbb{R}^d$ as:

$$f(x) = \sum_{i=1}^{M} a_i \cos(\omega_i^T x) + b_i \sin(\omega_i^T x)$$

where the frequencies ω_i are deterministic values and the coefficients $\{a_i, b_i\}$ are i.i.d random variables:

$$a_i \sim \mathcal{N}\Big(0, \frac{\sigma_0^2}{M}\Big), \qquad\qquad b_i \sim \mathcal{N}\Big(0, \frac{\sigma_0^2}{M}\Big)$$

with their variance scaled down by the number of basis functions. Defining $\phi(x) = [\cos(\omega_1^T x), \quad \sin(\omega_1^T x), \ldots, \cos(\omega_M^T x), \quad \sin(\omega_M^T x)]^T \in \mathbb{R}^{2M}$, we obtain $f(x)$ as a

zero-mean Gaussian process with covariance function:

$$k(x, x') = \frac{\sigma_0^2}{M}\phi^T(x)\phi(x')$$

which differs to the Random Features kernel by a scale factor σ_0^2. This basis was introduced by Lazarò et al.[9] mainly to reduce the computation complexity of the inverse of the covariance matrix. Consider N datapoints, define the matrix $\Phi(X) = [\phi(x_1), \ldots, \phi(x_N)]$ and assume additive noise $e \sim \mathcal{N}(0, \sigma_n^2)$ to the measurements, we obtain the covariance matrix:

$$K(X, X) = \frac{\sigma_0^2}{M}\Phi(X)\Phi(X)^T + \sigma_n^2 I_N$$

Applying the inversion lemma, we can obtain the equivalent inverse matrix as:

$$K(X, X)^{-1} = \frac{1}{\sigma_n^2}\Big(I_N - \Phi(X)^T A^{-1}\Phi(x)\Big), \quad A = \Phi(X)^T\Phi(X) + M\frac{\sigma_n^2}{\sigma_0^2}I_{2M}$$

where A is a $2M \times 2M$ matrix. If the $2M < N$ we can reduce the computational complexity of the inverse matrix while preserving accuracy of the approximated kernel (recall the uniform bound). The training is done by optimizing the marginal log-likehood with the fixed frequencies treated as hyperparameters, with the hope that it can find resonable approximations of spectral density, in this case we are indeed trying to infere some structure of the kernel. In Lazarò, they state that a stationary gaussian process can be seen as a neural network with one infinite-width hidden layer with weights distributed according to $S(\omega)$ and trigometric activation functions. This is somewhat interesting but also useless, we can't work with infinite samples or hidden-units. Moreover, both of these approaches use only trigonometric approximations, which can be seen as a symmetric Dirac's delta approximation of the spectral density:

$$\begin{aligned} S(\omega) &= \int \frac{1}{M}\sum_{i=1}^{M}\cos(\omega_i^T\tau)e^{-j\omega^T\tau}d\tau \\ &= \frac{1}{M}\sum_{i=1}^{M}\frac{\delta(\omega-\omega_i)+\delta(\omega+\omega_i)}{2} \end{aligned}$$

where the δ's are weighted uniformly. Now, if it was the case that the kernel function that best suits our data admits discrete spectral density, even in the most trivial cases, the discrete representation should look like:

$$S(\omega) = \sum_{i=1}^{+\infty}\frac{a_k}{2}(\delta(\omega-\omega_i)+\delta(\omega+\omega_i))$$

where $a_k \geq 0$ and $\sum_{i=1}^{+\infty} a_k < +\infty$. This is the main cause of overfitting for both approaches: the frequency components are truncated and not scaled down. Even if we optimize them, their weights are associated to a uniform spectral density.

A possible solution could be to associate different variances to the r.v $\{a_i, b_i\}$ adding $2M$ hyperparameters to the optimization problem, which in itself is not infeasible, but considering the non-convex nature of the marginal log-likelihood function, we would incur in many stationary-local points losing interpretability of the solutions. These were the main results that lead to the approximation of kernels for gaussian process, their flexibility and ease to train was of particular interest at the time. This was known as sparsification of the spectrum and it allowed to fasten the computation of either prediction or training of the gaussian processes for high amounts of data. Although simple, they were powerful techniques for the regression task, but their generalization properties were unsatisfactory when applied to more complex problems – mainly because of their persistent oscillatory behavior.

4.1.2 Continuous Spectral Densities

So far we've seen the approximation of the spectral density as a sum of Dirac's deltas. In general, we'd like to use continuous kernel functions with different properties to explain data and sparsification of the spectrum would demise generalization properties of the original kernel. Consider the scalar SE Kernel and compute its Fourier transform:

$$\phi(\omega) \triangleq \mathcal{F}\{k_{SE}\}(\omega) = \int_{-\infty}^{\infty} e^{-\frac{l^2\tau^2}{2}} e^{-j\omega\tau} d\tau \propto e^{-\frac{\omega^2}{2l^2}}$$

where in this case $l_s = 1/l$ is the so called length-scale parameter. The classic interpretation for l_s is that it regulates the strenght of the correlation between datapoints, while using the spectral representation we can deduce that the lengthscale actually controls the frequency components of the kernel inputs and therefore of the regressed functions. We can see this more clearly in Figure 4.2, where we simulate sample paths of a Gaussian process with a Gaussian spectral density in a uniform grid. This is what makes the kernel gaussian a

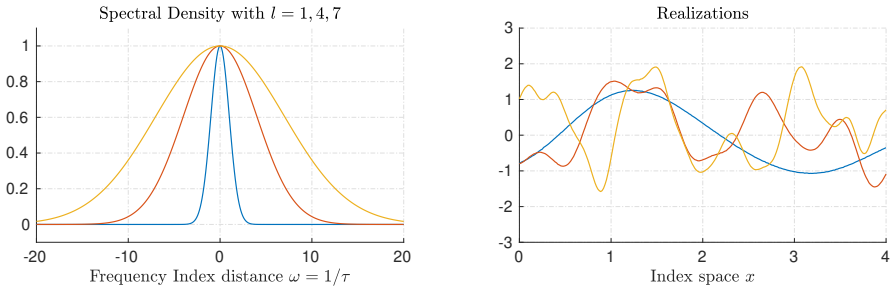


Figure 4.2: Spectral density with different lengthscales $l = 1, 3, 7$ with realizations of the associated gaussian process. The spectral densities weren't normalized for clarity purposes.

really useful tool, by controlling a single parameter we can get drastical different

behaviors. Although very flexible, one of its pitfalls is its generalization properties when applied to more complex phenomena. When it is not able to make good predictions, during the training phase, it tends to increase or anhilated its lengthscale parameters, higher lengthscales means constant behavior while smaller lengthscales means more oscillatory behavior, this can lead to poor generalization performances. The main cause of this unwanted pathology is that the kernel has the same power spectral density for the regression task. To adress this lack of flexibility, in Wilson et al.[26], they used the Bochner's representation to approximate directly the spectrum with a mixture of gaussians. They noticed that the SE Kernel spectral density was still a Gaussian, taking advantage of its universal approximation property, they modelled each spectral dimension as a coupled pair of gaussian distributions (so symmetry would be preserved) and the resulting kernel was obtained by applying the inverse Fourier transform. Their idea was that a high number of Gaussians for each dimension would be able to approximate any spectral density. More exactly, for inputs $\tau \in \mathbb{R}^d$, each single mixture density was modelled as:

$$\phi_i^{(j)}(\omega) = \frac{1}{\sqrt{2\pi(l_i^{(j)})^2}} \exp\Big\{ - \frac{(\omega_i - \mu_i^{(j)})^2}{2(l_i^{(j)})^2} \Big\} \Rightarrow \Phi_q(\omega) = \prod_{i=1}^{d} \left(\frac{\phi_i^{(q)}(\omega) + \phi_i^{(q)}(-\omega)}{2} \right)$$

where $\mu^{(q)} = [\mu_1^{(q)}, \ldots, \mu_d^{(q)}]$ and $l^{(q)} = [l_1^{(q)}, \ldots, l_d^{(q)}]$ are hyperparameters of the q mixture. The resulting spectral approximation S_Q was a weighted mixture, whose Fourier transform was the Spectral Mixture kernel K_Q:

$$S_Q(\omega) = \sum_{q=1}^{Q} w_q \Phi_q(\omega) \Rightarrow K_Q(\tau) = \sum_{q=1}^{Q} w_q \Big(\prod_{i=1}^{d} \exp(-(l_i^{(q)})^2 \tau_i^2) \cos(\mu_j^{(q)} \tau_i) \Big)$$

This is a promising result because we have a closed-form expression of the approximated kernel, in particular it isn't based on sampling and the permanent oscillations are no longer a problem because of the exponential decay as can be seen in Figure 4.3. In this case the trained hyperparameters can be interpreted,

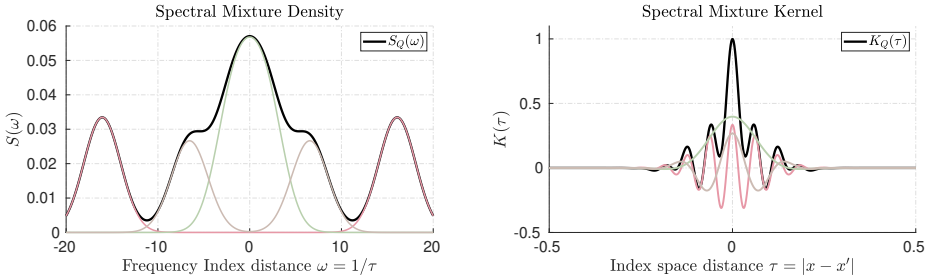


Figure 4.3: Spectral Mixture with $Q = 3$ for the scalar case. Left the spectral density approximation, right resulting kernel.

μ is the center of the spectral distribution and l is the range of feature frequencies which compose the kernel. The problem with this kernel is that the input

features are inpendepent from each other, which is rarely the case for physical systems as the states may appear as linear combinations. This result was later generalized[22], obtaining more general conditions to have a full approximation to any stationary kernel. Before stating their result, we have to formalize what we wanted to achieve: we are looking for a class of kernel functions which is flexible, tractable and can resemble any stationary kernel up to an arbitrary precision. The last condition is the focus of the next part, where we formalize how any kernel is the limit of a class of kernels S_Q. Before that, notice how we have obtained two kind of spectral approximations: discrete dirac's deltas and a continuous mixture. This is a consequence of Lebesgue's Decompostion theorem, which state that a positive finite measure $\mu(d\omega)$ can be divided as:

$$\mu = \mu_{sing.} + \mu_{cont.}$$

where $\mu_{cont.}$ is an absolutely continuous measure which admits density and $\mu_{disc.}$ is a pure-point measure supported on a countable set (i.e. discrete). One causes constant oscillatory behavior while the other has decaying frequency components.

4.2 Generalized Stationary Kernels

We can now present the generalization of the Spectral Mixture kernels, formalized by Samo[22]. This class of kernels is very flexible and it can, in theory, approximate any stationary kernel. This means that if we were to gather enough data of a stationary process, we could precisely reconstruct the covariance function that characterizes such process. In this work we are not trying to infer the original kernel, if the power spectral density $S(\omega)$ of the process was available we wouldn't need to recur to approximations, moreover trying to reconstruct the power spectral density is a whole different problem. Instead, we are interested in having a powerful class of kernel functions that can infer more complex patterns from the available observations and improve the overall prediction performance of the gaussian process, in particular this approximation has a nice interpretion based on Mercer's theorem(3.2.2): adapting $S(\omega)$ based on data actually means changing the basis functions which determine our class model. The generalized spectral kernels can be seen as a general extension of the Sparse Spectrum kernels, where the frequencies are treated as deterministic parameters that have to be optimized, in our case through the Marginal Likelihood, while avoiding its overfitting tendencies by introducing exponential decay term. Another of its key features is that, under simple assumptions about the kernel, we can embbed geometrical properties such as differentiability of the resulting process, this together with its simple analytical expression makes the generalized spectral kernel an attractive implementation on the NMPC scheme.

Notions of convergence and Dense sets

Before introducing the generalized spectral kernel, we briefly introduce their mathematical contruction to highlight its main characteristics.

Convergence notions The approximation of a stationary kernel $k(\tau)$ is based on the convergence of a sequence of functions in the spectral domain, we have the following notions of convergence

- **Point-Wise convergence**: A sequence of functions $\{f_n(x)\}$ converges to a function $f(x)$ in the point-wise sence if $|f_n(x) - f(x)| < \epsilon, \forall x \in \mathbb{R}^d$ as $n \to \infty$.
- L_1**-Convergence**: A sequence of functions $\{F_n(x)\}$ converges to $F(x)$ in the L_1-sense if $\|F_n(x) - F(x)\|_{L_1} \to 0$ as $n \to \infty$, where we considere the canonical L_1-norm $\|g - f\|_{L_1} = \int |g(x) - f(x)|dx$.

Consider now functions that are absolutely integrable, i.e $\int |f(x)|dx < +\infty$, then they admit a Fourier transform. Assume that a sequence of $\{f_n\}$ and f in $\mathbb{R}^d$ are absolutely integable functions, denote with $\{F_n\}$ and F their Fourier Transform. We have that:

$$|f_n(x) - f(x)| \leq \Big| \int_{\mathbb{R}^d} (F_n(\omega) - F(\omega)) e^{j\omega^T x} d\omega \Big| \leq \int_{\mathbb{R}^d} |F_n(\omega) - F(\omega)| d\omega$$

$$\Rightarrow |f_n(x) - f(x)| \leq \|F_n(\omega) - F(\omega)\|_{L_1}, \forall x \in \mathbb{R}^d$$

If we are able to find a sequence $\{F_n\} \in L_1(\mathbb{R}^d)$ that converges to F in the L_1-sense, then we are able to say that the sequence of functions $\{f_n(x)\}$ converges point-wise to $f(x)$. The main problems with this approach is that we don't have an explicit form for the sequences nor the limit, moreover it is difficult to prove that an arbitrary sequence in L_1 converges to an element in L_1. To tackle these difficulties we need to introduce the notion of a dense set.

Family of Dense functions in $L_1(\mathbb{R}^d)$ We say that a set $\mathcal{G}$ is dense in the set $\mathcal{H}$ if any sequence $\{g_n\}$ of elements in $\mathcal{G}$ admits a limit $h \in \mathcal{H}$. We'd like to find a set of spectral densities $\{F_n\}$ that are dense in the spectral domain with respect to the L_1-norm, in that way the family of inverse Fourier Transforms of $\{F_n\}$ will also be dense in the space of integrable kernels (original domain) with respect to the pointwise convergence. To aid such purposes, we state the following theorem:

Theorem 4.2.1 (Wiener's Tauberian Theorem)**.** *If ϕ is a function in $L_1(\mathbb{R}^d)$, a necessary and sufficient condition for the set of all linear combinations of translations of ϕ to be dense in $L_1(\mathbb{R}^d)$(in the L_1-sense) is that the Fourier transform of ϕ:*

$$F(x) \triangleq \mathcal{F}\{\phi\}(x) = \int_{\mathbb{R}^d} \phi(\omega) e^{-jx^T\omega} d\omega$$

has no zeros.

Recall again the spectral density of the SE Kernel:

$$S(\omega) = \frac{1}{\sqrt{2\pi l^2}} e^{-\frac{\omega^2}{2l^2}} \Rightarrow \mathcal{F}\{S\}(x) = e^{-l^2\frac{x^2}{2}} > 0$$

It satisfies all the conditions of Wiener's Tauberian Theorem, it is the known Universal Approximation property of the Gaussian functions and it was brought up by Wilson et al[26].

Generalized Spectral Kernels

Having a function that is dense in L_1 is not enough, some conditions must be imposed to have a dense set of functions for the spectral densities:

- **Positivity**: A valid function $\phi(\omega)$ must be strictly positive.
- **Symmetry**: The basis function $\phi(\omega)$ must be symmetric, moreover to any translation $\phi(\omega - \omega_i)$, we have to add another translation in the opposite direction $\phi(\omega + \omega_i)$ to obtain a valid spectral density.
- **Normalization**: The resulting approximation must integrate to 1. This is required if the kernel represents the correlation function, which differs from the covariance function just by a scale factor σ^2.

These conditions must be satisfied to be a valid approximation of a spectral density of a stationary process, we end up with the following spectral density family:

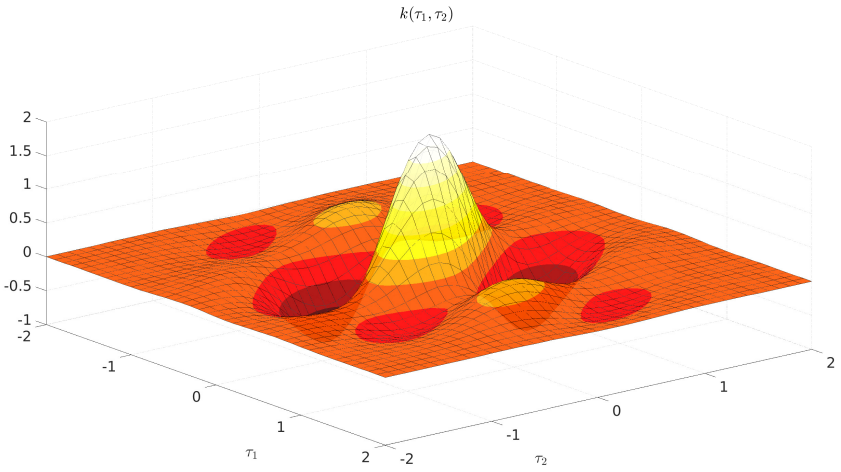
$$\Phi(\omega) = \sum_{i=1}^{K} \frac{\alpha_i}{2}(\phi(\omega - \omega_i) + \phi(\omega + \omega_i)) = \sum_{i=1}^{K} \frac{\alpha_i}{2}(\phi_i^+ + \phi_i^-)$$

where α_i are non-negative coefficients associated to the pair (ϕ^+, ϕ^-) and such that $\sum_i \alpha_i = 1$ for correlations functions, from such approximation the resulting kernel is found by applying the Fourier transform to Φ. These is the idea behind theorem in Samo et al.[22] which we state:

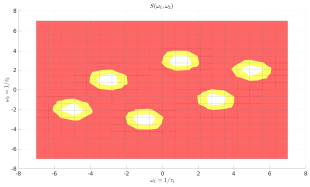
Theorem 4.2.2 (Generalized Stationary Kernels)**.** *Let h be a real-valued positive semidefinite, continuous, and integrable function such that $h(\tau) > 0, \forall \tau \in \mathbb{R}^d$. The family of functions:*

$$k_K(\tau) \triangleq \sum_{i=1}^{K} \alpha_k h(\tau \odot \gamma_k) cos(2\pi\omega_k^T \tau)$$

with $\omega_k, \gamma_k \in \mathbb{R}^{+d}$, $\alpha_k \in \mathbb{R}_+, K \in \mathbb{N}^*$ is dense in the family of stationary real-valued kernels with respect to the pointwise convergence, where $\tau \odot \gamma$ is the entry-wise product.



(a) 2D Generalized Kernel, $K = 3$.



(b) 2D Spectral Density SE, top view to highlight the centers.

Figure 4.4: 2D Generalized Kernel using a SE Kernel as basis function.

With this theorem, it's possible to approximate arbitrarily well any stationary continuous kernel function and any kernels whose spectral measure is a pure-point measure (i.e. they admit Fourier expansion) by the continuity of the basis kernel $h(\tau)$. Another consequence of theorem 4.2.2 is that the resulting kernel inherits all the properties of the basis kernel, that is, if we choose a Matern kernel which is $2p$ times differentiable, the resulting approximation will be $2p$ times differentiable. This is an exciting result, we know that the kernel function is strictly connected to a Reproducing Kernel Hilbert Space and the functions which belong to such space inherit the properties of the kernel: by approximating the spectral measure we are changing the class of functions we can infere, an implicit model selection procedure. Of course it comes with a trade-off, the number of hyperparameters is $O(DK)$ with D the parameters dimension and K the number of clusters. This can lead to overfitting for high numbers of mixtures, moreover their optimization mainly carried by maximizing the Marginal Log-Likelihood which notoriously has many local-stationary points with few hyperparameters, let aside a few hundred.

4.2.1 Catching up with the times

We are interested in the spectral domain approximations because of the resulting kernel analitycal tractability. Other kinds of methods are also available, which mainly focus on the feature space of the gaussian process. The first method is

an sparsification on the input by selecting inducing points on the feature space [19], which have greater influence on the predictions, such that we can reduce the computational complexity of the inversion of the covariance function, we won't discuss these methods because they don't actually approximate the kernel structure. The second method is called Deep Kernel Learning[27], where the input features are modelled by a neural network and then passed to a gaussian process. The later is taking off the ground because of their great generalization capabilities, it is a known fact that neural networks have great approximations properties but their interpretability is somewhat lost. Nonetheless, although they have a high number of parameters to train, modern software renders their implementation rather easy. To overcome the training of millions of parameters, they rely on the techniques such as backprogation and stochastic optimization. The hope in introducing neural networks is to find feature mappings that improve the prediction capabilities: the idea is the same as random fourier features, where we stated that a lower trigonometric representation can be applied to find richer flexibility in the inference. Similarly, in deep kernel learning framework the task to find interesting features is assigned to the neural netkwork, and given that a gaussian process can be seen as a hidden layer with infinite-units, the last layer of the network is an infinite expansion which depends on the learned features. As

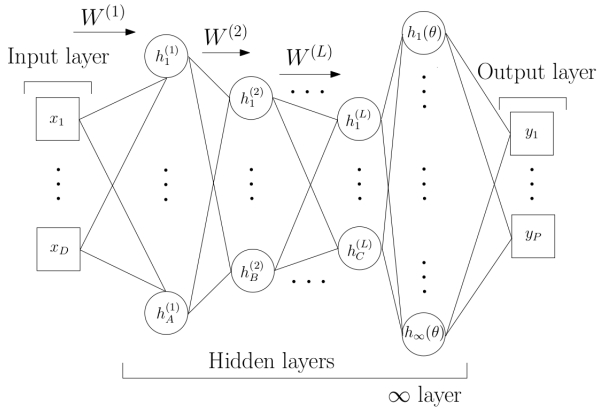


Figure 4.5: Deep Kernel Learning model structure.

stated in Ober et al. [18], although these structures are able to generalize well, the can suffer from overfitting. In such work they demonstrate how the MLL optimization for the hyperparameters becomes a dull goal function. Indeed, they apply the standard decomposition for the marginal likelihood and observed that by applying the neural network, the data-fit term can always be minimized and the only factor which influences the minimization is the complexity-term. They state that such minimization occurs by correlating all the data-points and hence becomes uninformative, we have encountered the same problem for the generalized spectral kernel. Still, the DKL gives satisfactory results even in the presence of this bad behavior – they attribute this feat to the mini-batch stochastic learning applied during the training phase. Moreover, they state that a full Bayesian

approach is enough to make DKL robust to these saturation of learning. This is the main curse of the generilized spectral kernel: its flexibility, unless constrained in some way by some additional regularization functions, can result in overconfident models and its application on the MPC scheme could give unsatisfacotry performances. Either way, its approximation capabilities can be compared to that of Neural Networks, and its simple analytic form cannot be disregarded in an iterative optimization problem.

Chapter 5

Simulation Results

In this chapter we present the results of the generalized spectral kernel applied to the NMPC controller. We will compare the overall performance with respect to the Squared Exponential (SE) kernel, widely used in the machine learning field. Recall our goal is to compensate the original model, based on the prediction errors obtained from previous a control tak. We will first discuss the training procedure and its main intricasies, evaluating each trained GP in two datasets: training set and validation set. This will gives us initial hints about the performance of the GP, but in general is not enough to decide which one performs better. We will apply metrics defined on the differences between linearization matrices obtained from the true model and the joint nominal-GP model, computing them on the training and validation set. Although not possible in practice, here we are interested in evaluating the capabilities of the Generalized Spectral Kernels applied to dynamical systems. Later, we will evaluate their generalization properties by comparing the same matrices metrics on the closed-loop trajectory and their relative one-step-ahead prediction errors, which should be in general a good metric to evaluate the overall performance of the trained GP. As mentioned before, given that we are training the GPs in a particular control-task, we'd like to be sure that the trained model is not task-focused. That is, if we were to apply the learnt model in a slightly different control task, we would like to still obtain a satisfactory performance: this indicates that the GP is indeed compensating the parametric uncertainties present in the nominal model. To aid such purposes, we will train the gaussian models in a richer dataset obtained by estimulating the system. We will see that this improves performance and the models obtained outperform the ones trained on datasets with nominal trajectories. Actually, this should be something we expect from the gaussian process: richer data means more information about the system. We will perform these analisis on two nominal models: changing the lengths and mass of the pendulum, as well as the mass cart. We will see that the Gaussian process, with SE and generalized spectral (SP) kernels are able to achieve improvements, however the SP kernel outperforms the SE in many simulations.

5.1 Setting up the models

In this section we describe the training of the prediction models defined in (3.50), implemented in our benchmark dynamical system: the Cart-Pendulum. We compare the following kernels:

$$k_{SE}(\mathbf{x}-\mathbf{x}') = \alpha^2 \exp\Big(-(\mathbf{x}-\mathbf{x}')^T\Sigma^{-1}(\mathbf{x}-\mathbf{x}')\Big) \tag{5.1}$$

$$k_{SP} = \sum_{i=1}^{2} \alpha_i^2 \exp\Big(-(\mathbf{x}-\mathbf{x}')^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{x}')\Big) cos(\mu_i^T(\mathbf{x}-\mathbf{x}')) \tag{5.2}$$

where $\mathbf{x} = [p, \theta, v, \omega, u]^T \in \mathbb{R}^5$ contains states and inputs of the system. The hyperparameters of the respective kernels are the lengthscales $\Sigma = diag\{l_1, \ldots, l_5\}$, feature frequencies $\mu_i \in \mathbb{R}^5$ and scale factors $\alpha^2 \in \mathbb{R}_+$. For the generalized spectral (SP) kernel, we choose as basis function the Squared Exponential to show that adding more flexibility indeed improves the performance and it doesn't lead to overfitting. We decided to implement only 2 mixtures because throughout the learning procedures, we noticed that mixtures more or less converged to the same μ. Notice that we defined μ as feature frequencies because there isn't a physical interpretation of such hyper-parameters.

5.1.1 Compensating parametric uncertainties

Recall again our original control problem for the cart-pendulum system (2.26). In Figure (5.1), we present two simulations, one with correct parameters and another with $l = 0.8\ m$ where we assume that all state variables can be measured. The simulations run the original system, while the control trajectories are computed with the nominal parameters. As can be notice, the behavior is widely different. A more detailed information of what is happening comes from the linearization matrices. Within a control task, at each state initial state $\hat{x}_k$, we evaluate the distance between the linearization matrices of *true* f_t and nominal f_n model. We use the Frobenius norm for A_k and l_2-norm for B_k, obtaining:

$$\|A_{\text{true}} - A_{\text{nom}}\|_F = \sqrt{Tr((A_{\text{true}} - A_{\text{nom}})(A_{\text{true}} - A_{\text{nom}})^T)} \tag{5.3}$$

$$\|B_{\text{true}} - B_{\text{nom}}\|_2 = \sqrt{(B_{\text{true}} - B_{\text{nom}})^T(B_{\text{true}} - B_{\text{nom}}))} \tag{5.4}$$

these will be our metrics to evaluate closed-loop performance between prediction models later into the results. In Figure 5.2, we present the distances of the linearization matrices between the closed-loop trajectories of the nominal model with respect to the true model. Within the simulations of the true model, there are present errors from the application of the integrator operator, which can be neglected. Our goal here is to use the prediction errors obtained from these simulations and try to compensate the nominal model, by modelling the velocity mismatch models with gaussian processes and training them with the available prediction erros.

Nominal Models: To examine the learning properties of both kernels SE and SP, we carried our analisis with two different nominal models which we refer to as:

- **Nominal Length**: Nominal model with a wrong value for the length of the pendulum: $l = 0.8m$.
- **Nominal Mass**: Nominal model with wrong parameters: $l = 0.4\ m$, $M = 0.9\ kg$ and $m = 0.2\ kg$. The nominal trajectory for the pendulum swing-up is depicted in Figure 5.3.

The nominal model has just a slight parametric error, while the second model are much futher from the true ones. By evaluating the generalizion properties of the GPs using these two models, we should be able to compare more effectively which kernel has better properties to learn the mismatch model.

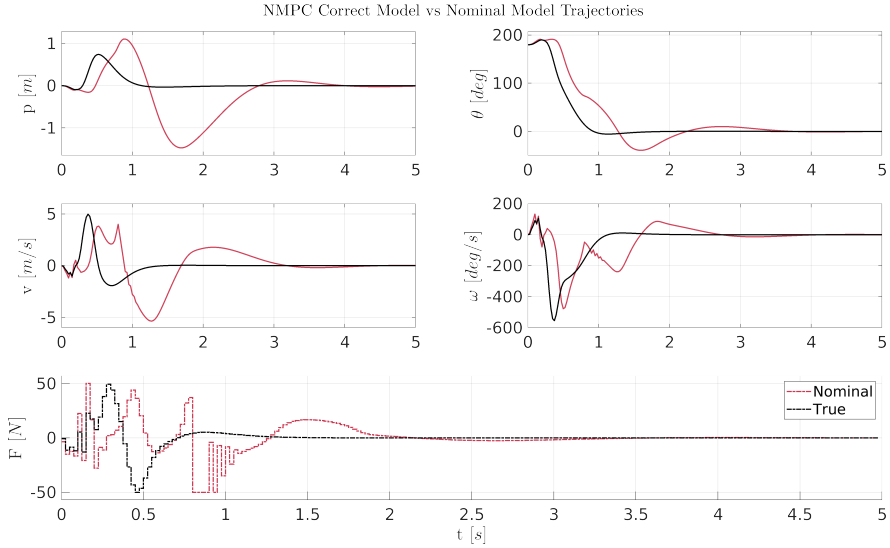


Figure 5.1: Cart-Pendulum simulation using the correct parameter $l = 0.5$ m. Control actions are obtained from the NMPC scheme using the correct model and the nominal model with $l = 0.8$ m.

Types of datasets: We will train the gaussian models with two different of datasets. The aim is to obtain more information about the true system, by applying an external known input added to the nominal input during the swing-up task, we get more erroneous predictions from the nominal model which should gives us more information about the mismatch model. We have the following datasets, which we refer to:

- **Nominal Inputs:** Data is collected from nominal trajectories obtained from a swing-up of the pendulum, as in Figure 5.1.
- **Excited Inputs:** Data is collected from trajectories with an external input applied, in particular we will excite the system with a sum of decaying sinusoids through out the nominal trajectory. An example is in Figure 5.4.

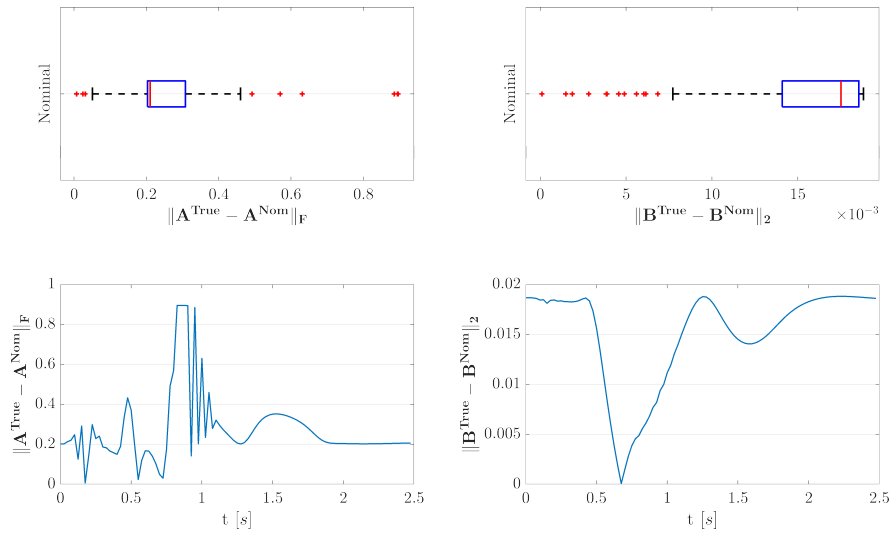


Figure 5.2: Linearization Matrices within the MPC scheme, evaluated within the nominal trajectory at each observed data point $\hat{x}_k, u_k$ with respect to the correct model.

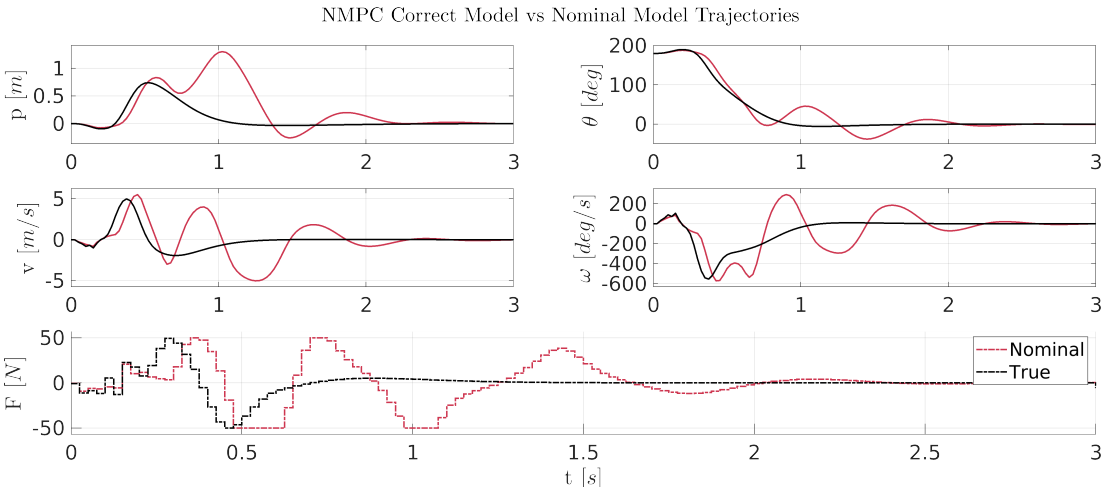


Figure 5.3: Cart-Pendulum simulation with nominal parameters $l = 0.4\ m$, $M = 0.9\ kg$ and $m = 0.2[kg]$.

5.1.2 Learning the hyper-parameters

We briefly discuss the training process and the problems that we encounter. This is the fundamental part of the entire procedure, we trained the models with the usual minimization of the negative marginal log-likelihood (NML) by employing the **gpr-torch** library. During this phase, we initially let all the hyperparameters to be traininable for both SE and SP kernels, noticing that the scale factors were very unstable or got annihilated by the end of the procedure. Although the objective function was indeed being minimized and reaching to local-minima, the performance in the NMPC was either not satisfactory or the solver wouldn't

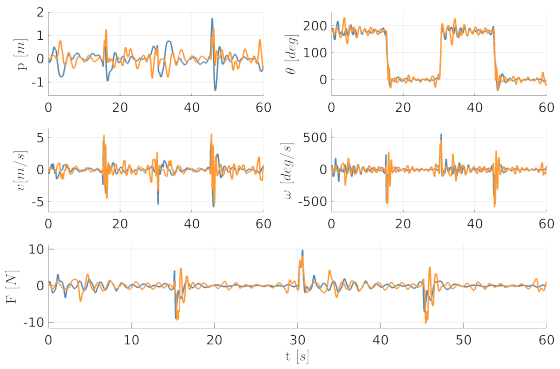


Figure 5.4: System trajectories excited by an external sum of sinusoids.

converge in the QP subproblem. As mentioned in Ober et al. [18], when dealing with kernels with a high amount of parameters, these scale factors tend to become a dull parameter and not improve the performance of the trained GP, we found the same troubles. We decided to fix the scale factors: for the SE we have $\alpha = 1$ and for the SP kernel we set then to be uniform $\alpha_i = 0.5$.

Mini-batch learning: In general, the standard training of the gaussian process uses only the N_{pred} datapoints used later for the prediction to optimize the marginal likelihood. We found out that this could lead into overfitting during the training, given that we have a relative high number of parameters. To improve the training procedure, we applied mini-batch training, stacking data obtained from previous swing-up procedures resulting in N_{tot} datapoints, dividing the whole dataset into random subsets of N_{pred} datapoints and, at each epoch, applying gradient-descent with the random subsets. To improve the numerical stability of the hyper-parameters μ, we scaled them by a constant and applied them into ReLu activations for the μ parameters and soft-plus activations for the length-scales parameters. This lead to an overall improvement of the training procedure, where the hyperparameters would converge to the same values. Indeed, in all the training procedures, the values converged for both grey-box and black-box models to:

$$\mu = [0, 2, 0, 0, 0] \quad \text{(Velocity GP)}$$
$$\mu = [0, 1, 0, 0, 0] \quad \text{(Angular Velocity GP)}$$

while the lengthscales hyperparameters changed from dataset to dataset.

5.2 Training and Validation performance

In this section we evaluate the results of the learning process of the proposed kernel, implemented in both grey-box and black-box models. The data set consists of N_{tot} datapoints, collected from the swing-up task simulated with the nominal models with the relative nominal inputs or excited inputs. Recall that the gaussian process uses N_{pred} points to make the predictions about the mismatch

of the nominal model. By looking at the fitting error on the N_{pred} datapoints, we evaluate how well the gaussian model describes the mismatch of nominal system on known test inputs. The remaining datapoints of the training set are the validation set, evaluating the fitting error on this set gives indication of the generalization properties of the gaussian model – if it indeed has learnt the mismatch model. This is a standard procedure and, in general, it gives good indication of the success of the learning procedure. In particular, we could say that the best performing kernel is the one that achieves lower fitting error in the validation set. However, once we use the chosen kernel to solve the OCP, one can observe that the closed-loop trajectory can wildly differ from the true closed-loop trajectory. That is, the gaussian process hasn't actually learnt the mismatch model – if that was the case, we should obtain similar trajectories. To better compare the generalization properties between kernels, we evaluated the linearization matrices in both the training and validation sets, and evaluated their distance to the linearization matrices obtained from the true model. This was done in order to establish which kernel is indeed learning the mismatch model, we have the modified metric for the matrices:

$$\|A_{\text{true}} - A_{\text{GP}}\|_F = \sqrt{Tr((A_{\text{true}} - A_{\text{GP}})(A_{\text{true}} - A_{\text{GP}})^T)} \tag{5.5}$$

$$\|B_{\text{true}} - B_{\text{GP}}\|_2 = \sqrt{(B_{\text{true}} - B_{\text{GP}})^T(B_{\text{true}} - B_{\text{GP}}))} \tag{5.6}$$

where A_{GP} and B_{GP} are the linearization matrices defined as in (5.1). Grey-box models have the nominal and GP linearizations, black-box models have only GP linearizations. To have improve the comparison, we removed the datapoints where the control inputs where near zero to avoid skewing the analysis – these points belong to parts of the system trajectory where the control task has been completed, i.e. the pendulum is stably upwards. This will give us the first indication of the overall success of the learning procedure for the gaussian models. We will use boxplots to have a better visualization about the distribution of the fitting errors in the training and validation set, moreover we set $N_{pred} = 200$ for the all the models. We are ready to present the results of the learning procedure for each nominal model, each with excited and nominal inputs, examining both the performance of the grey-model and black-box model. Recall that each mismatch velocity component of the nominal model was modelled with a gaussian process, we refer to them as v and ω without making difference between nominal and gaussian models.

5.2.1 Nominal Length model

Here we present the results of the learning of the mismatch model for nominal model with pendulum length $l = 0.8\ m$, where we have the fitting errors and linearization matrices norms of the GP models in the training and validation sets, trained with nominal inputs and excited inputs datasets, respectively.

Grey-Box models: Nominal and Excited inputs

The fitting performance are in favor of the SE kernels in the dataset with nominal inputs, while SP kernel is slightly better in the excited inputs datasets. We can see this more clearly by looking at the linearization matrices norms, which are in favor of the SE kernel in the validation set.

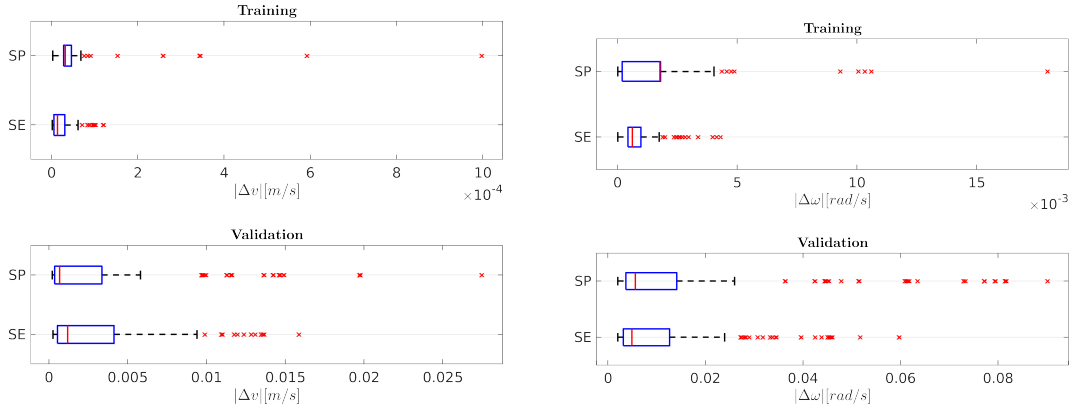


Figure 5.5: Grey-Box training and validation fitting errors for $l = 0.8\ m$ with nominal inputs.

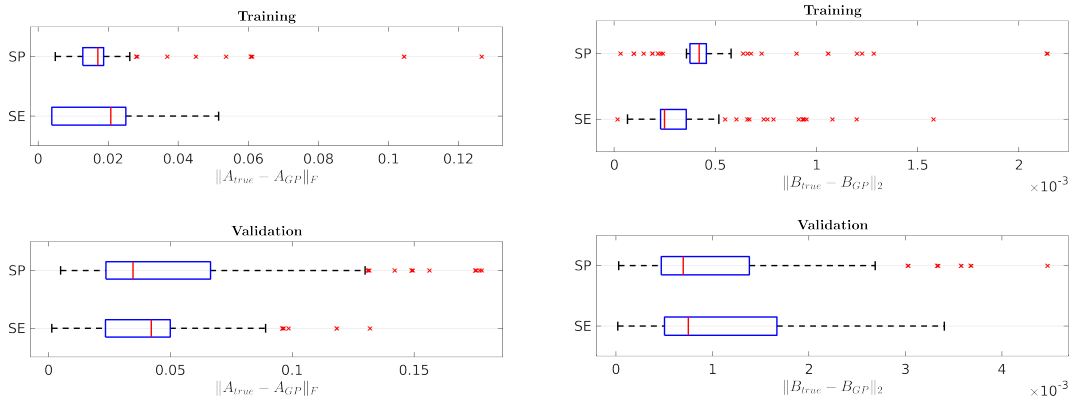


Figure 5.6: Grey-Box matrices norm for $l = 0.8\ m$ with nominal inputs.

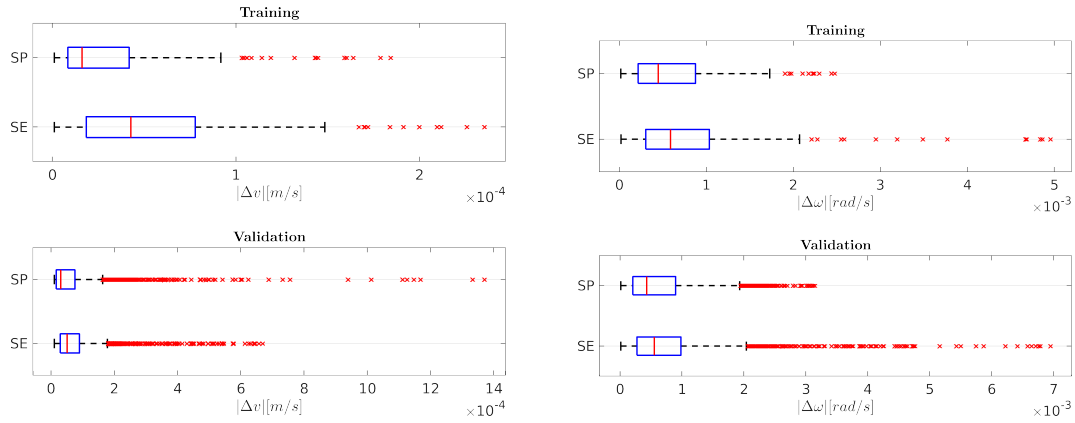


Figure 5.7: Grey-Box training and validation fitting errors for $l = 0.8\ m$ with excited inputs.

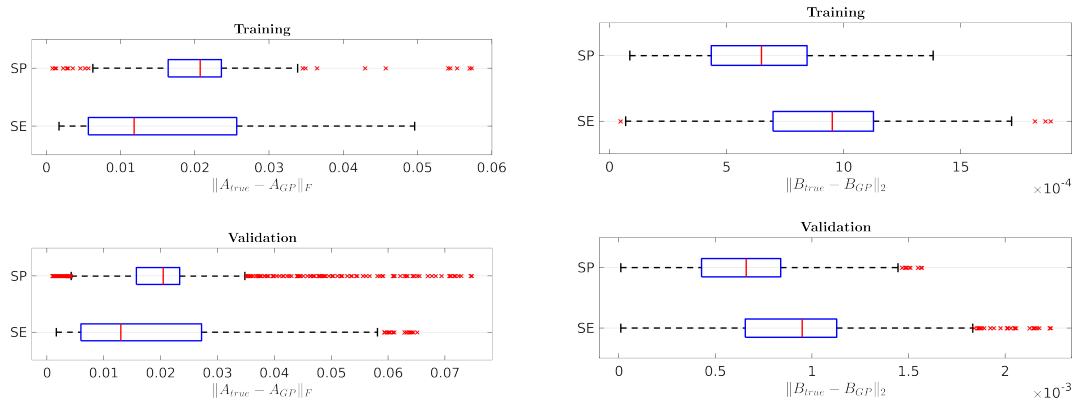


Figure 5.8: Grey-Box matrices norms for $l = 0.8\ m$ with excited inputs.

Black-Box models: Nominal and Excited inputs

Both kernels have similar fitting properties of the velocities increments, but the SP kernel is clearly better in the case of the linearization matrices in both nominal and excited inputs datasets. This can be asserted by looking the performance on the validation set.

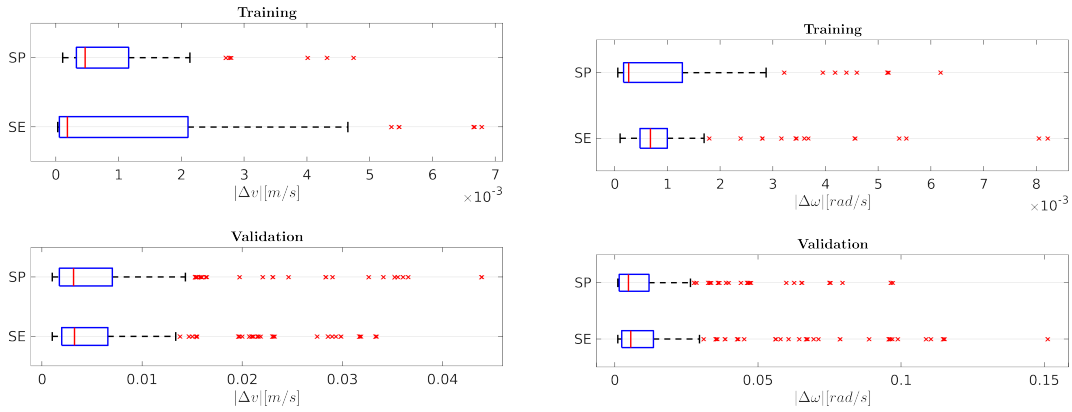


Figure 5.9: Black-Box training and validation fitting errors for $l = 0.8\ m$ with nominal inputs.

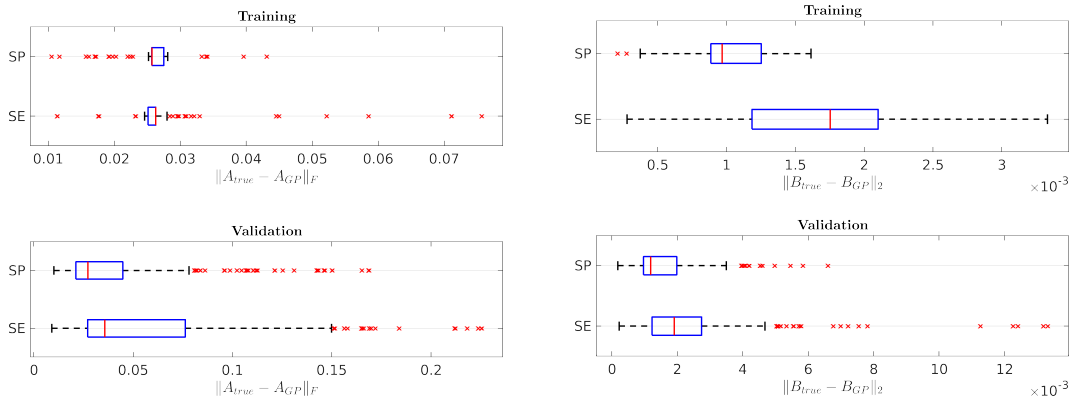


Figure 5.10: Black-Box matrices norms for $l = 0.8\ m$ with nominal inputs.

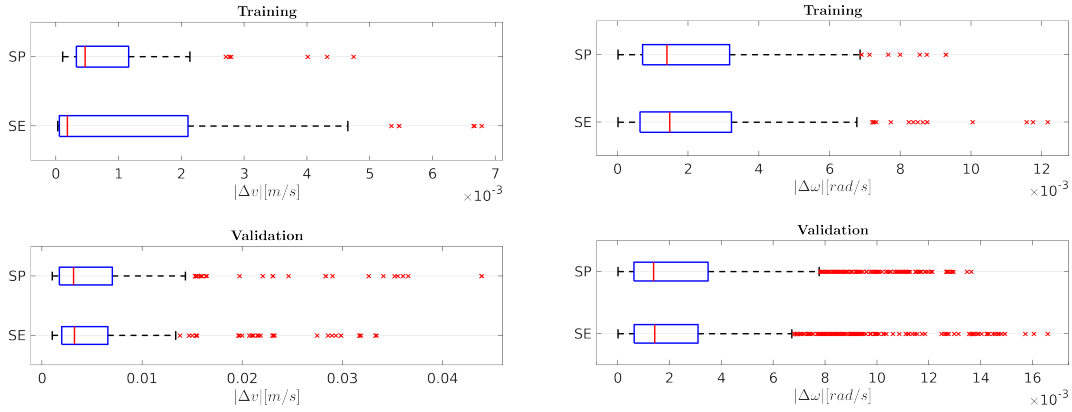


Figure 5.11: Black-Box training and validation fitting errors for $l = 0.8\ m$ with excited inputs.

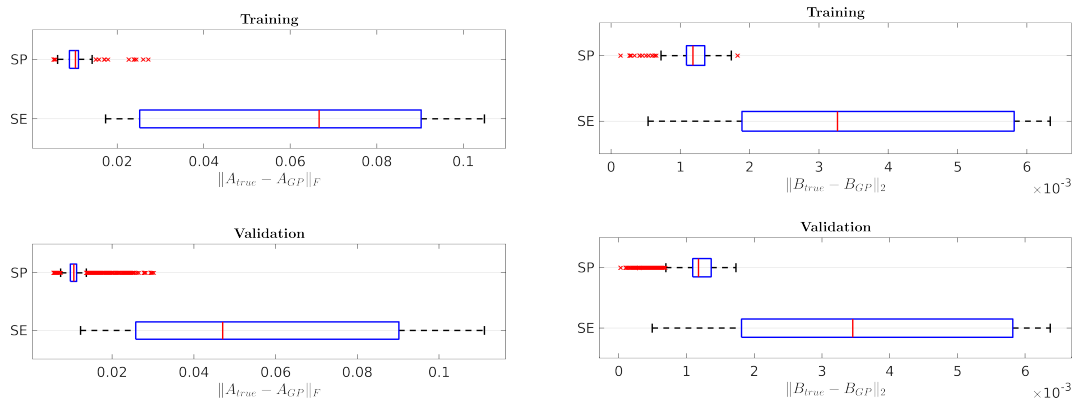


Figure 5.12: Black-Box matrices norms for $l = 0.8\ m$ with excited inputs.

5.2.2 Nominal Mass model

Here we present the results of the learning of the mismatch model for nominal model with parameters $l = 0.4\ m$, $M = 0.9\ kg$, $m = 0.2\ kg$, where we have the fitting errors and linearization matrices norms of the GP models in the training and validation sets, trained with nominal inputs and excited inputs datasets, respectively.

Grey-Box models: Nominal and Excited inputs

In this case the SE performs better at fitting the mismatch model, and the resulting linearization matrices are slighly in favor of the SE kernel.

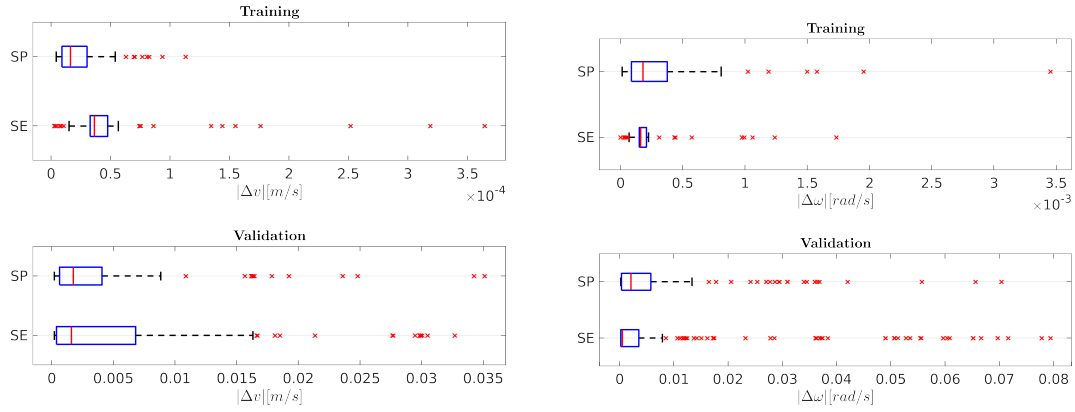


Figure 5.13: Grey-Box training and validation fitting errors $l = 0.4\ m$, $M = 0.9\ kg$,$m = 0.2\ kg$.

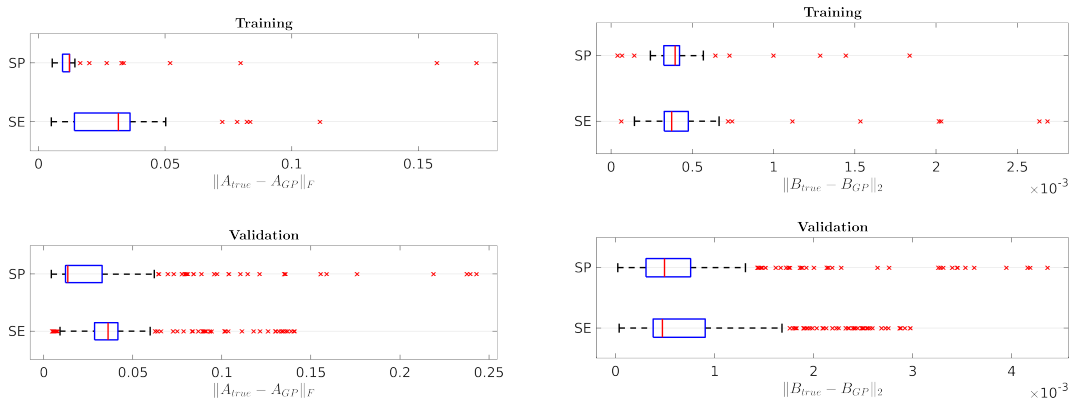


Figure 5.14: Grey-Box matrices norms for $l = 0.4\ m$,$M = 0.9\ kg$, $m = 0.2\ kg$.

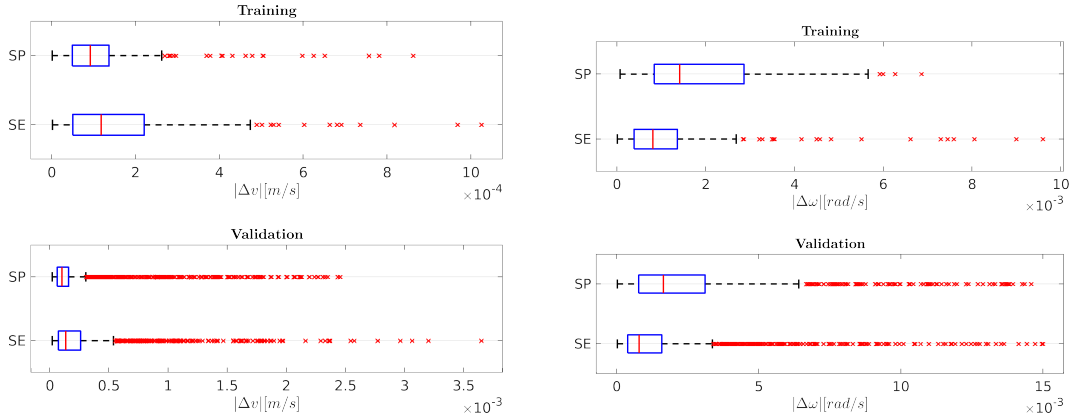


Figure 5.15: Grey-Box training and validation fitting errors for $l = 0.4\ m$, $M = 0.9\ kg$,$m = 0.2\ kg$, with excited inputs.

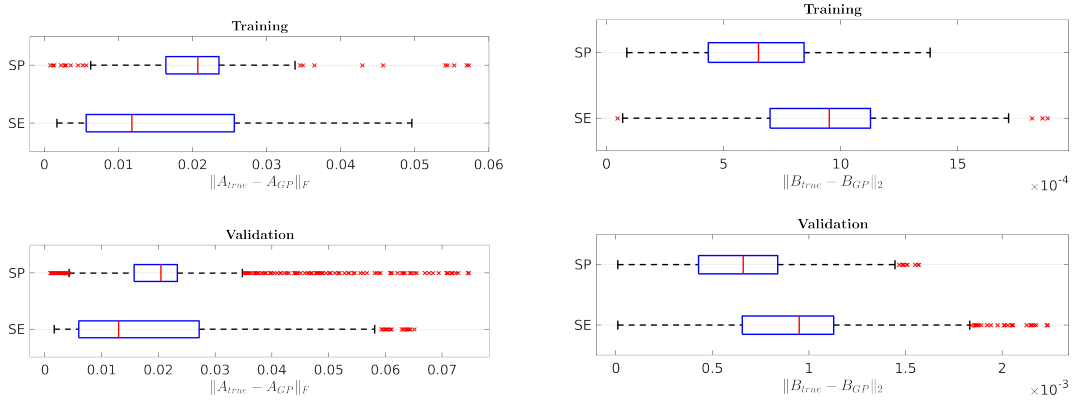


Figure 5.16: Grey-Box matrices norms for $l = 0.4\ m$, $M = 0.9\ kg$,$m = 0.2\ kg$, with excited inputs.

Blac-Box models: Nominal and Excited inputs

Both kernels have similar performance at fitting the velocity increments. However, the SP kernel seems to be better with the linearization matrices. Indeed, we have lower norms in the validation set, with both nominal and excited inputs.

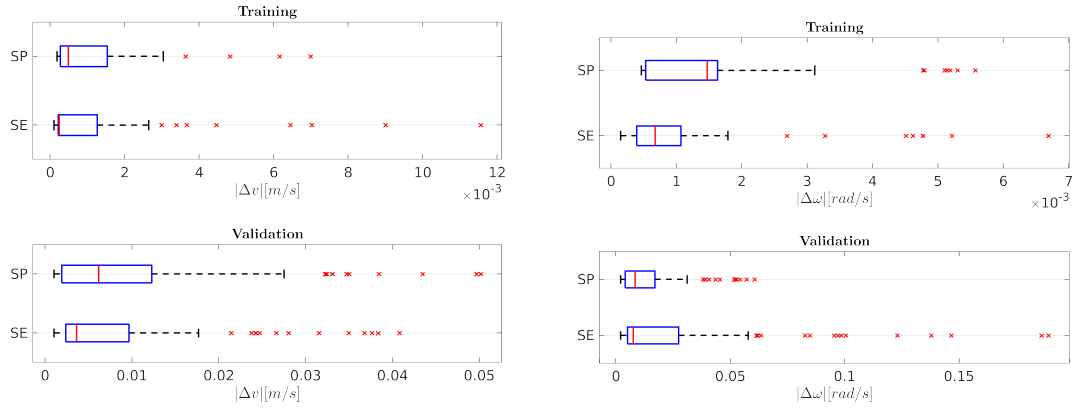


Figure 5.17: Black-Box training and validation fitting errors for $l = 0.4\ m$,$M = 0.9\ kg$, $m = 0.2\ kg$.

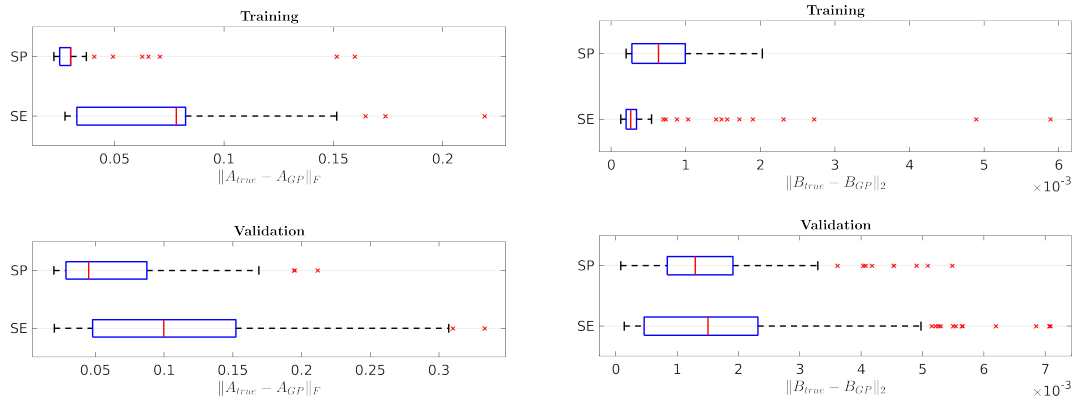


Figure 5.18: Black-Box matrices norms for $l = 0.4\ m$, $M = 0.9\ kg$,$m = 0.2\ kg$.

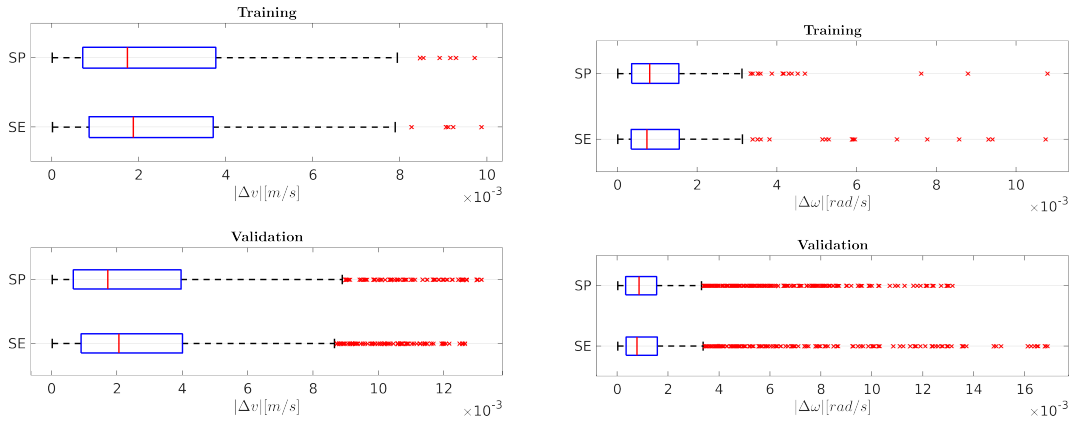


Figure 5.19: Black-Box training and validation fitting errors for $l = 0.4\ m$, $M = 0.9\ kg$,$m = 0.2\ kg$, with excited inputs.

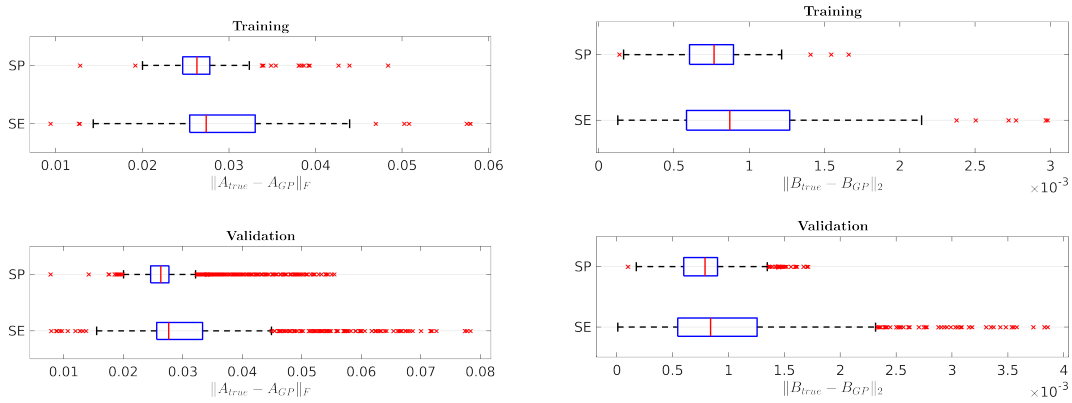


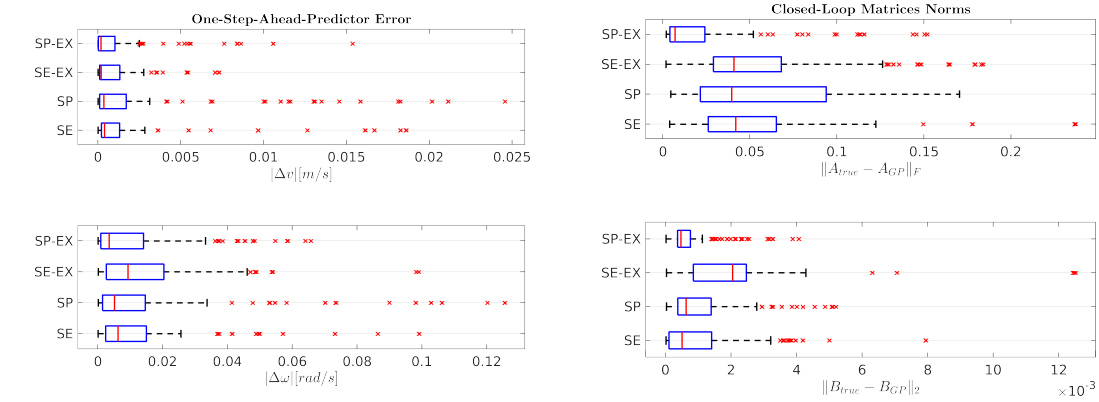
Figure 5.20: Black-Box matrices norms for $l = 0.4\ m$, $M = 0.9\ kg$,$m = 0.2\ kg$, with excited inputs.

5.3 Swing-Up Simulation

In this section we evaluate the performance of the trained GPs in the swing-up task of 30 seconds. For each GP, we evaluate its one-step-prediction performance, the norm of the linearization matrices and the distance between the closed-loop trajectory to the trajectory of the true model. We should expect good performances relatively for all GPs, given that they have been trained to do so. We will evaluate the GPs on both nominal length and nominal mass model. We evaluated all the models together, either trained on nominal trajectories or excited trajectories, to highlight if there's any improvement in performance by the richer dataset. Here, we define with SE-EX and SP-EX the respective models trained with excited inputs.

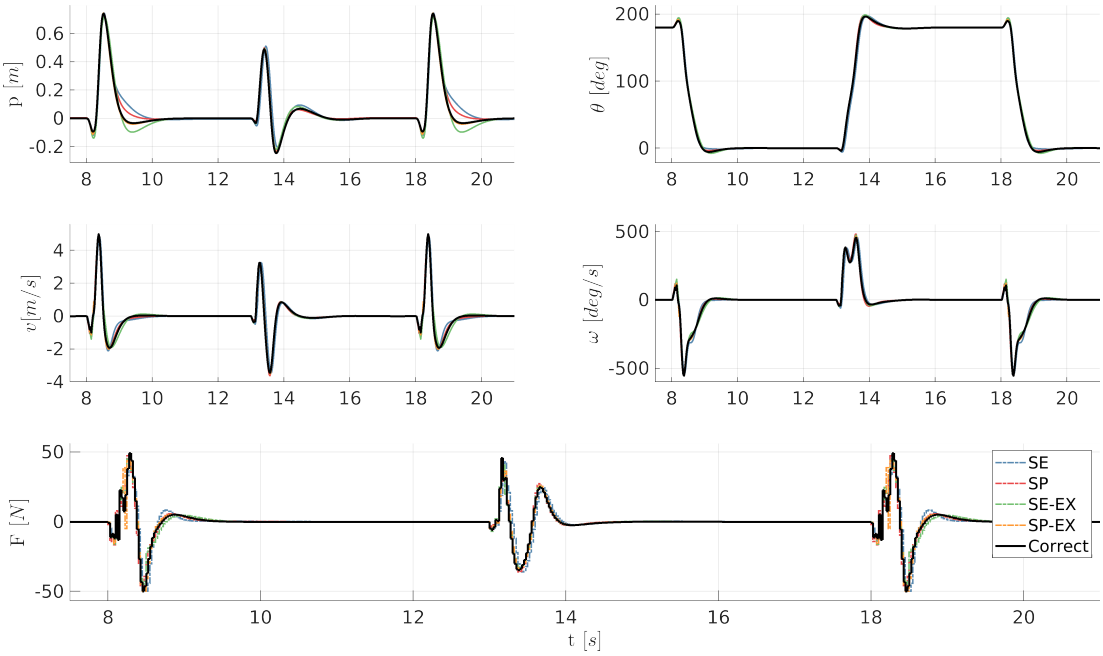
Nominal Length Grey-Box models

The one-step-ahead prediction errors are similar, but the SP-EX kernel is clearly better during the linearization phase. Moreover, the closed-loop trajectory for the SP-EX kernel is practically the same as the true model, confirming that a richer datasets improves the inference of the mismatch model.



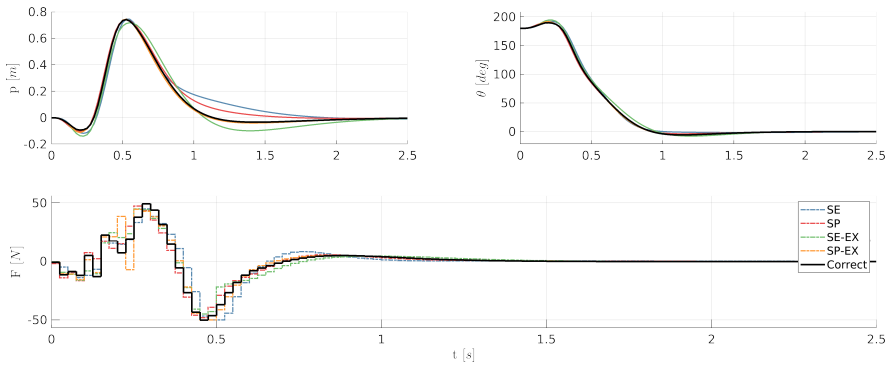
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

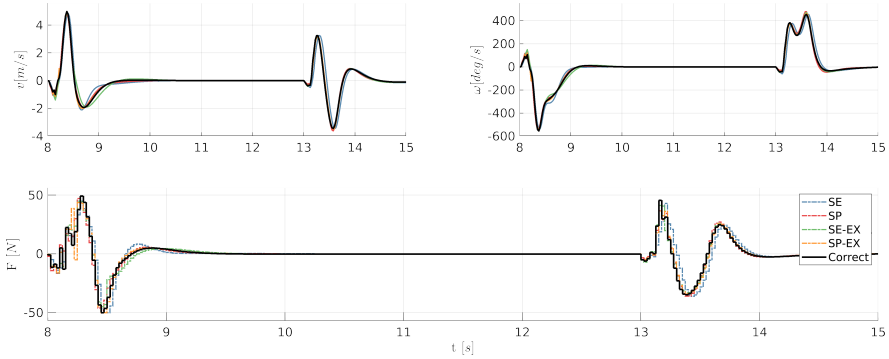


(c) Complete swing-up task.

Figure 5.21: Grey-Box closed-loop performances.

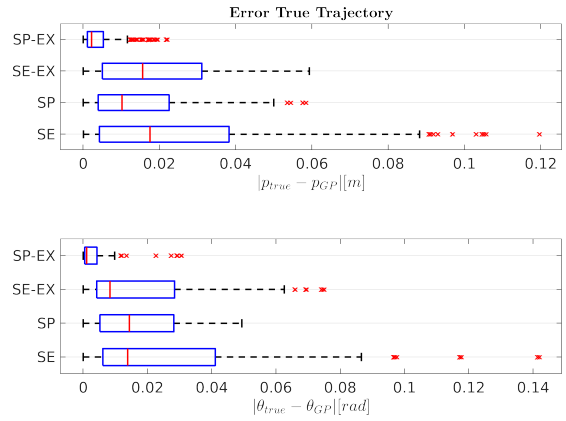


(a) Cart-pendulum positions.



(b) Cart-pendulum velocities

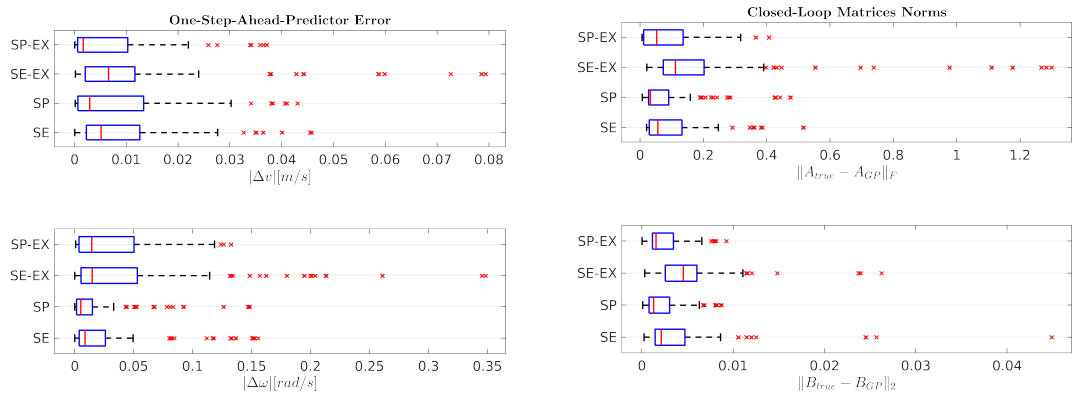
Figure 5.22: Grey-Box closed loop trajectories.



(a) Error with respect to the true trajectory.

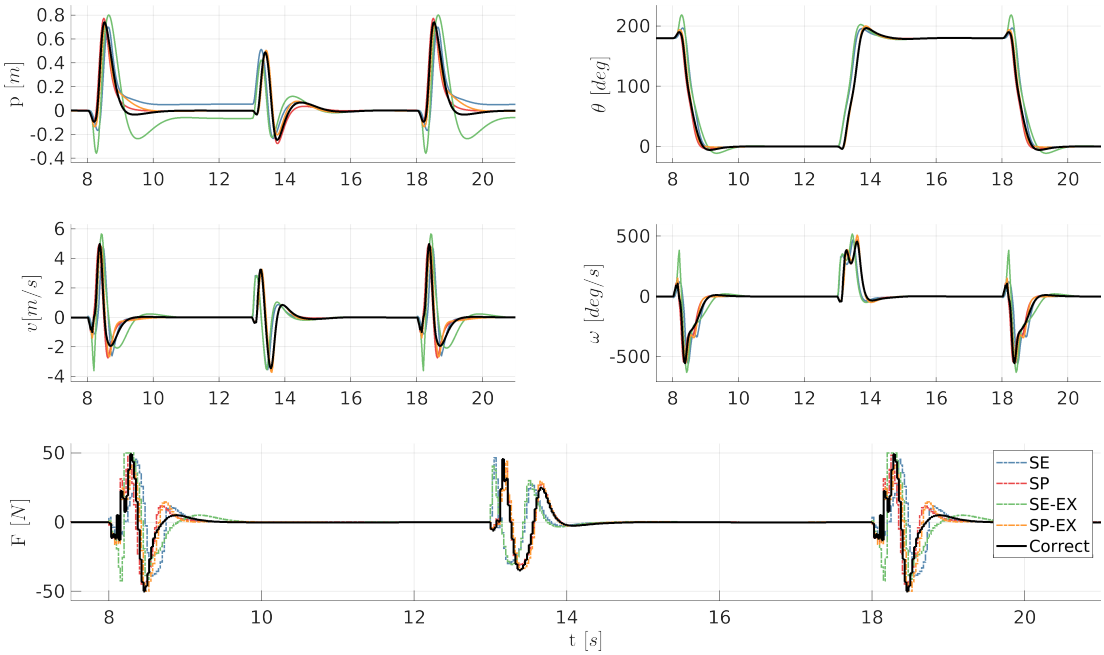
Nominal Length Black-Box models

In this case the prediction of the SE is sligthly better, however the SP kernel is better in the linearization matrices norm metric. This can be seen in the overall distance from the closed-loop trajectory.



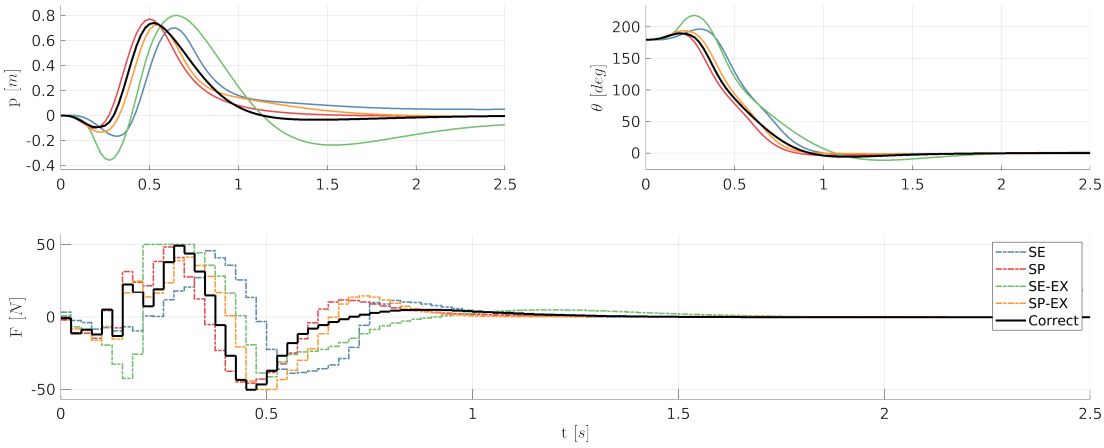
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

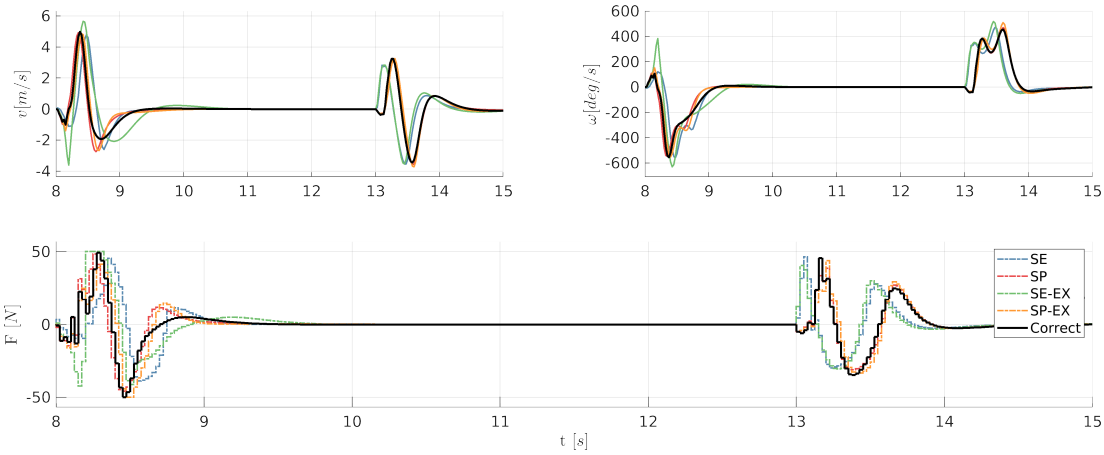


(c) Complete swing-up task

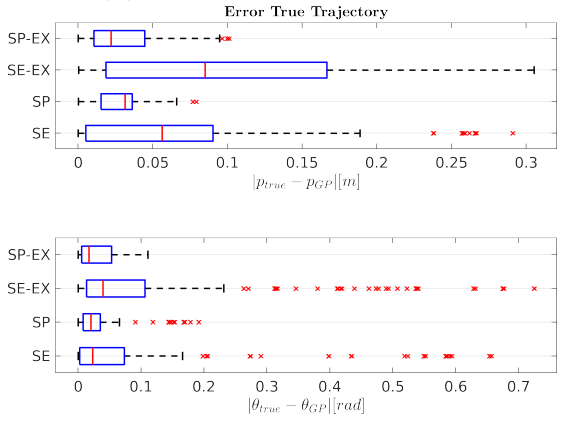
Figure 5.24: Black-Box closed-loop performances.



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.

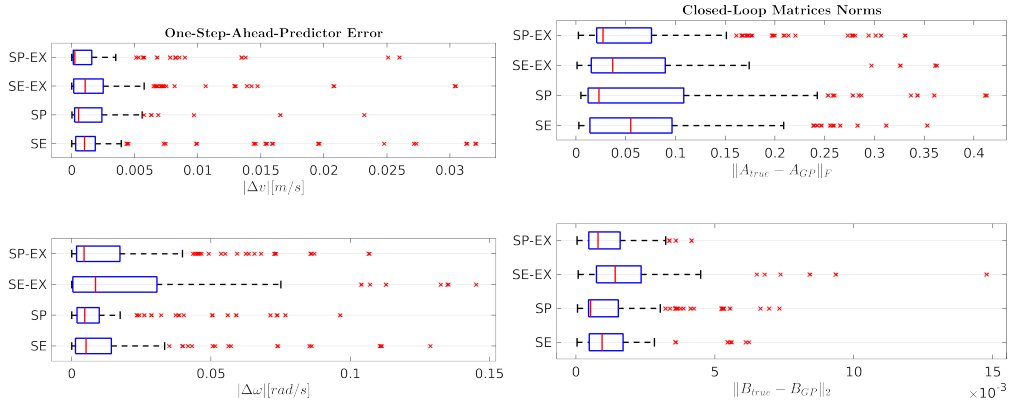


(c) Error with respect to the true trajectory

Figure 5.25: Black-Box closed loop trajectories.

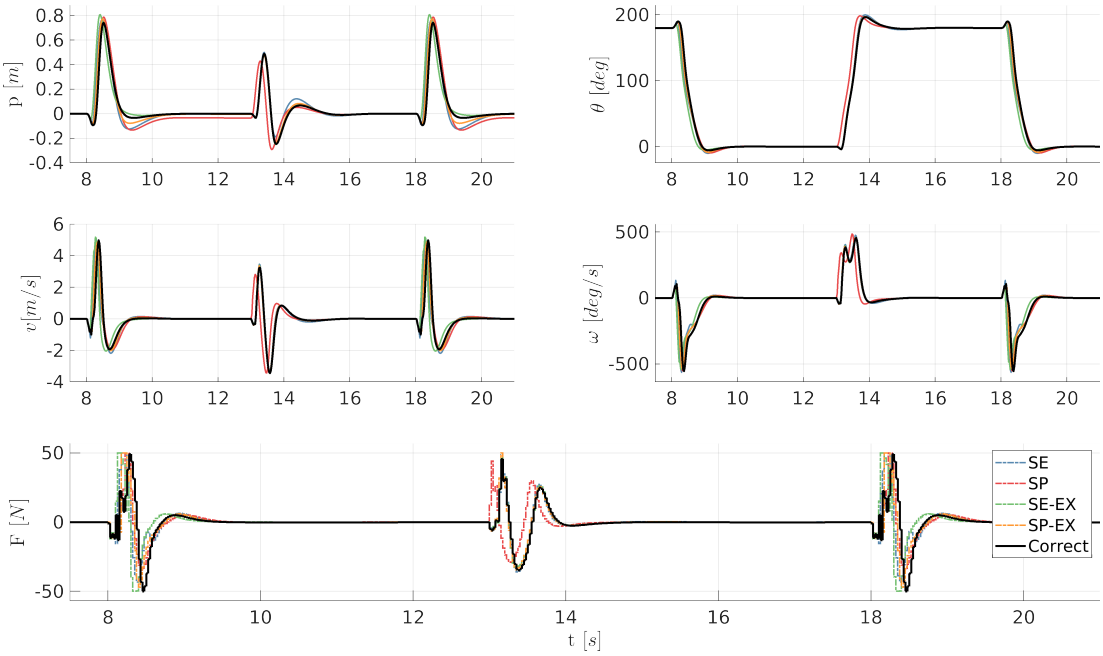
Nominal Mass Grey-Box models

Here we present the swing up for both grey-box and black-box models with the Nominal Mass model, indicating with SE-EX and SP-EX the models trained with excited inputs. Similar considerations as for the nominal length model hold. However, the SE kernel has a slithly better performance.



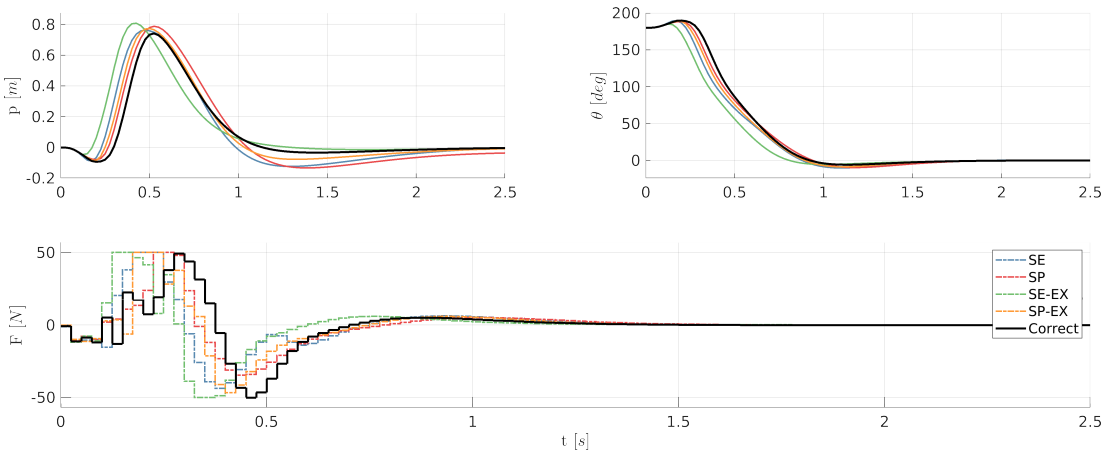
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

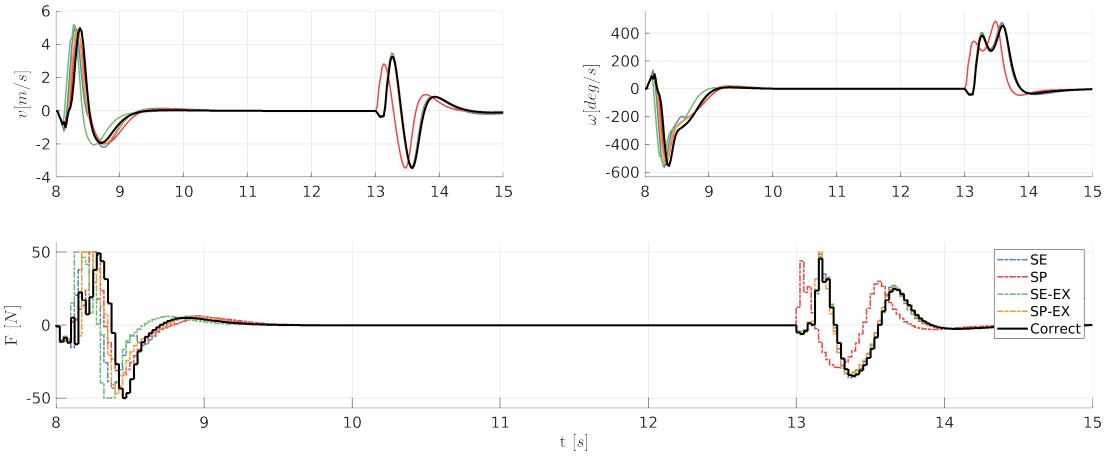


(c) Complete swing-up task

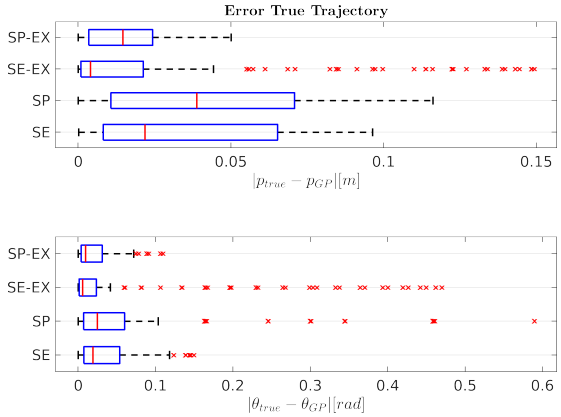
Figure 5.26: Grey-Box closed-loop evaluation.



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.

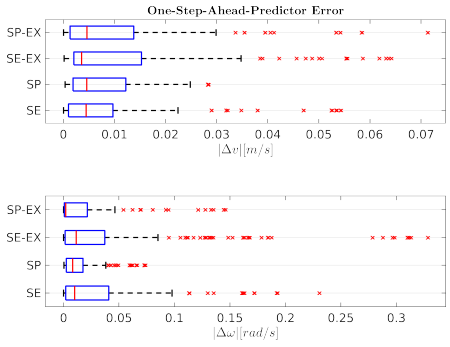


(c) Error with respect to the true trajectory

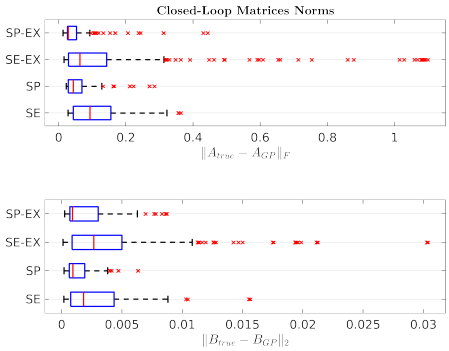
Figure 5.27: Grey-Box closed-loop trajectories.

Nominal Mass Black-Box models

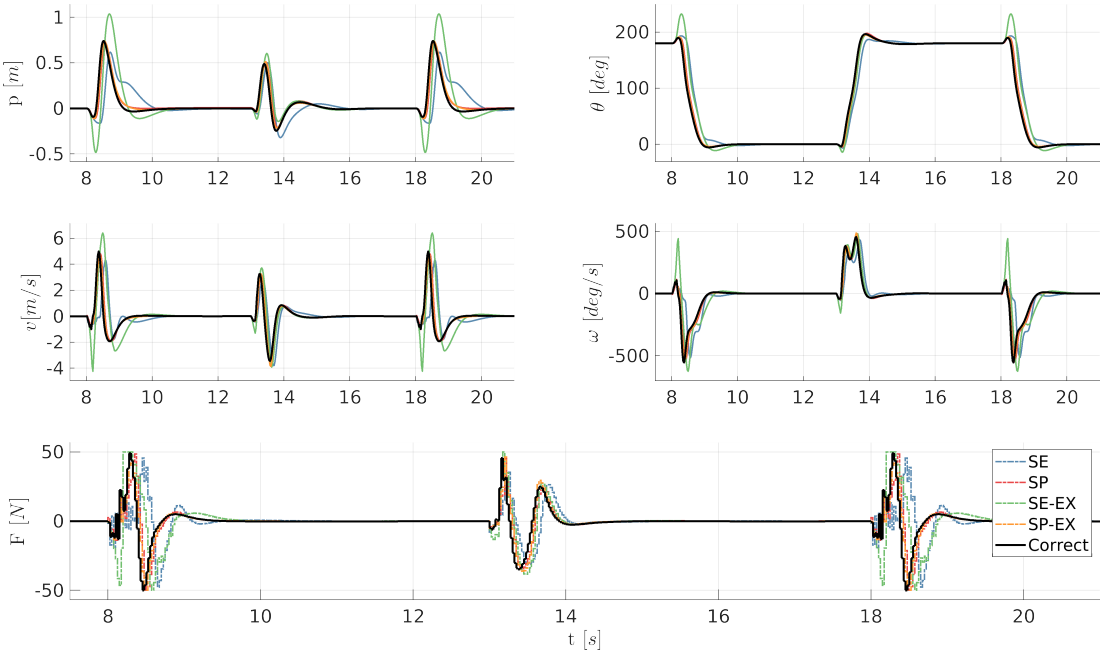
In this case, the SP kernel is clearly superior in all aspects. All the metrics are in its favor, it is easy to notice from the closed-loop trajectory that it is generalizing better.



(a) One-Step-Ahead Prediction Error.

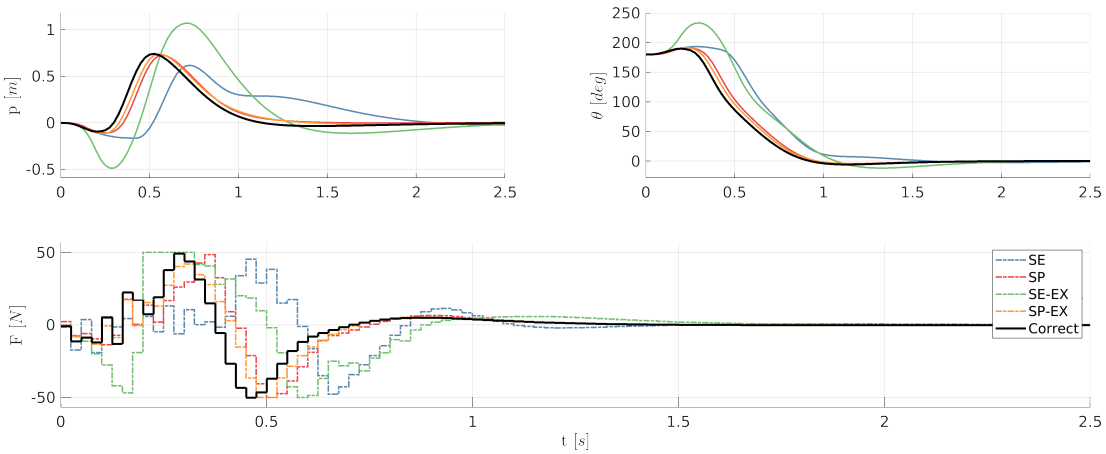


(b) Linearization matrices norm.

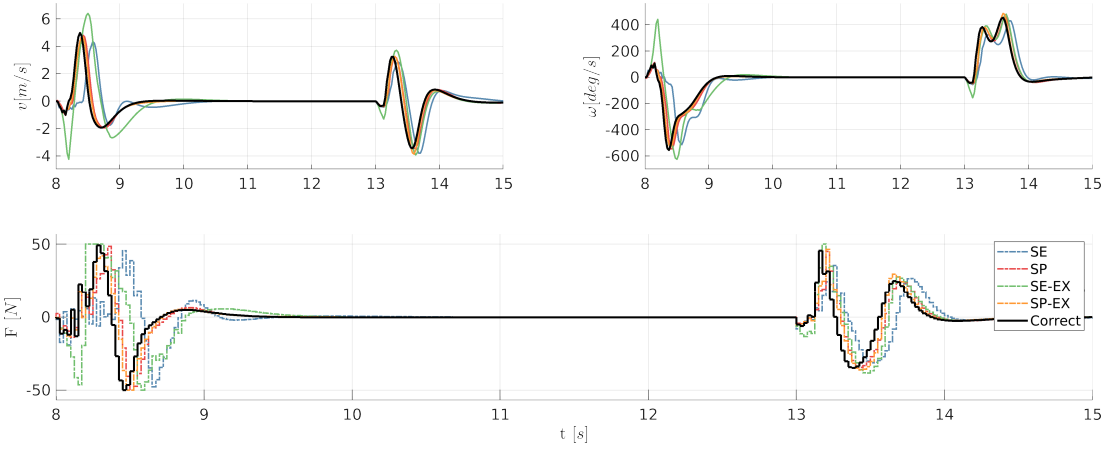


(c) Complete swing-up task.

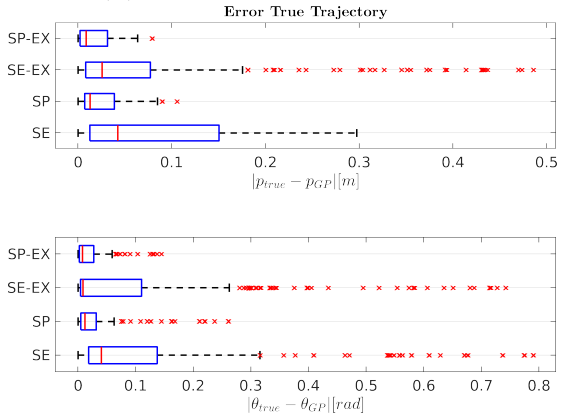
Figure 5.28: Black-Box closed-loop evaluation.



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.



(c) Error with respect to the true trajectory.

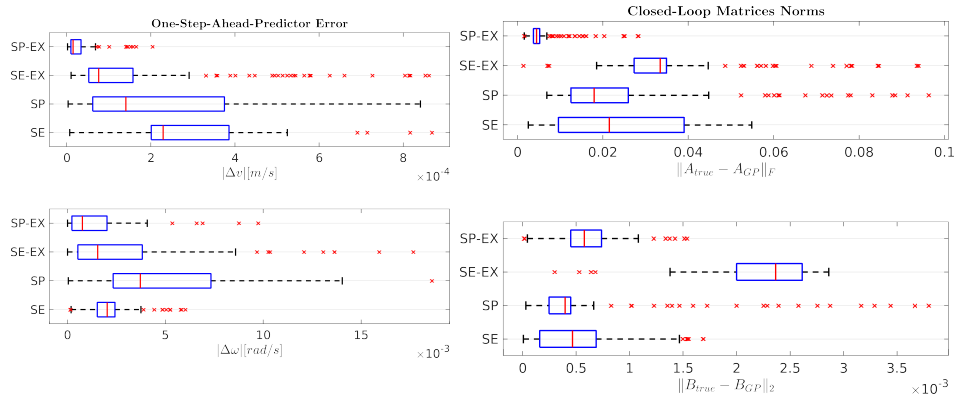
Figure 5.29: Black-Box closed-loop trajectories.

5.4 Moving Cart Simulation

In this section we present the results of the moving cart simulation, aimed at the evaluation of the generalization performances of the proposed GP. The task is to move between cart positions while maintaining the pendulum up, to see if changing the tasks affects the overall behavior of the gaussian models. The results from these simulation will allow us to clearly see which kernel is the superior choice. As mentioned before, the gaussian processes are trained with swing-up simulation data and their predictive performance are much better suited for that task.

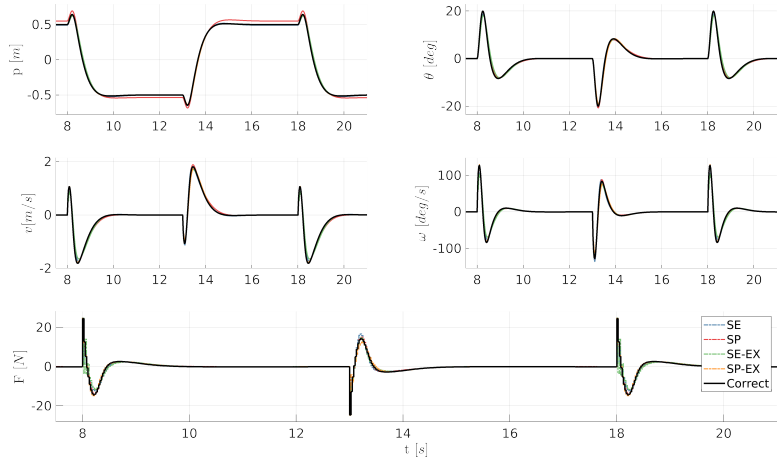
Nominal Length Grey-box models

The same considerations for the swing up simulation hold. The SP-EX kernel is outperforming all the other kernels, but interestingly, the SP has a closer closed-loop trajectory to true model. We can say that it has learnt the model.



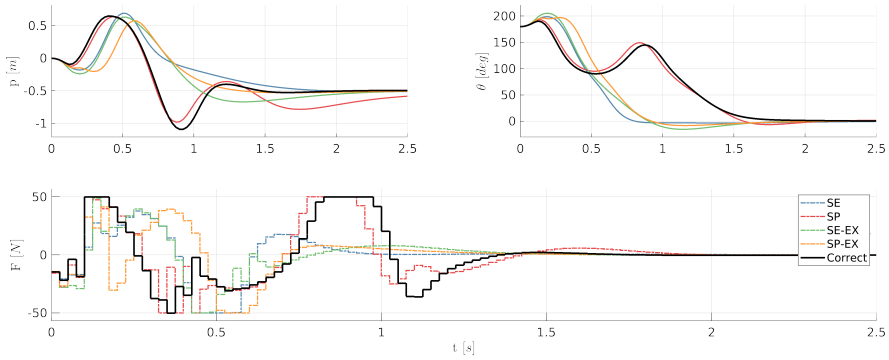
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

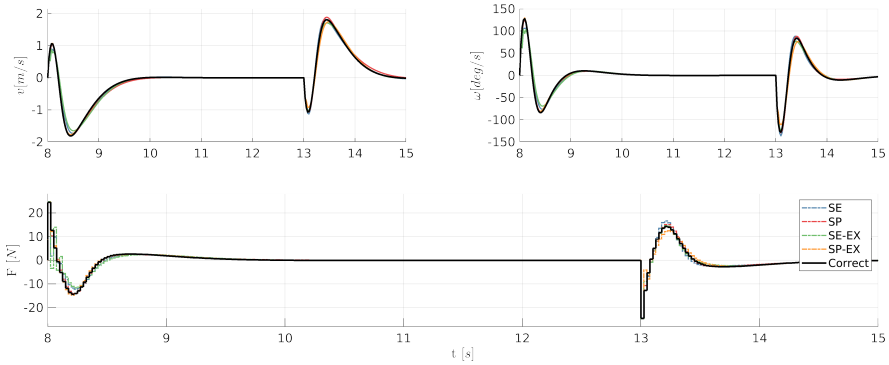


(c) Complete moving cart task.

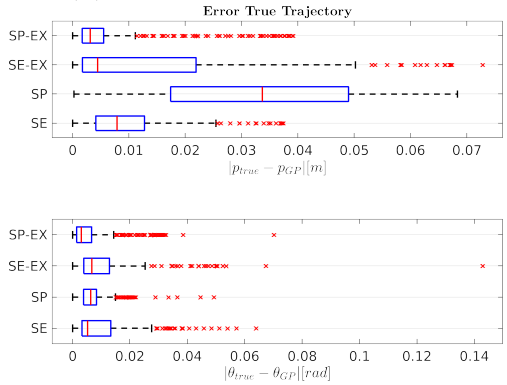
Figure 5.30: Grey-Box closed-loop performances



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.

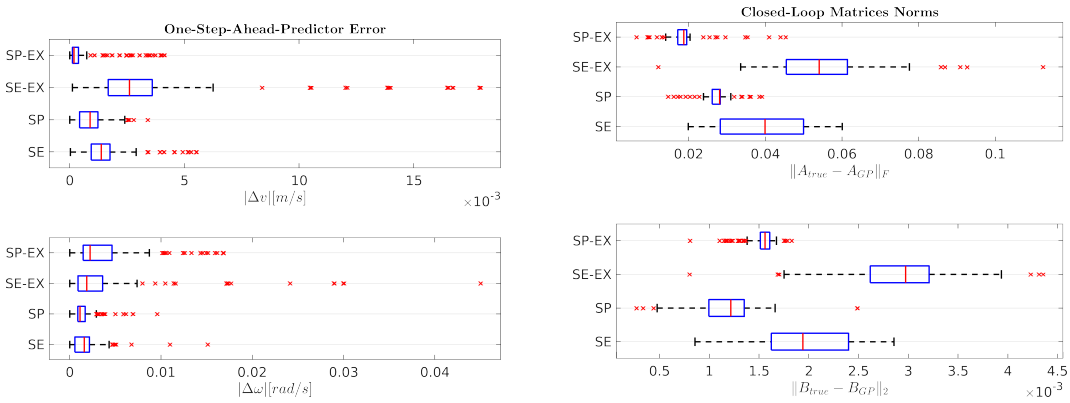


(c) Error with respect to the true trajectory.

Figure 5.31: Grey-Box closed loop trajectories.

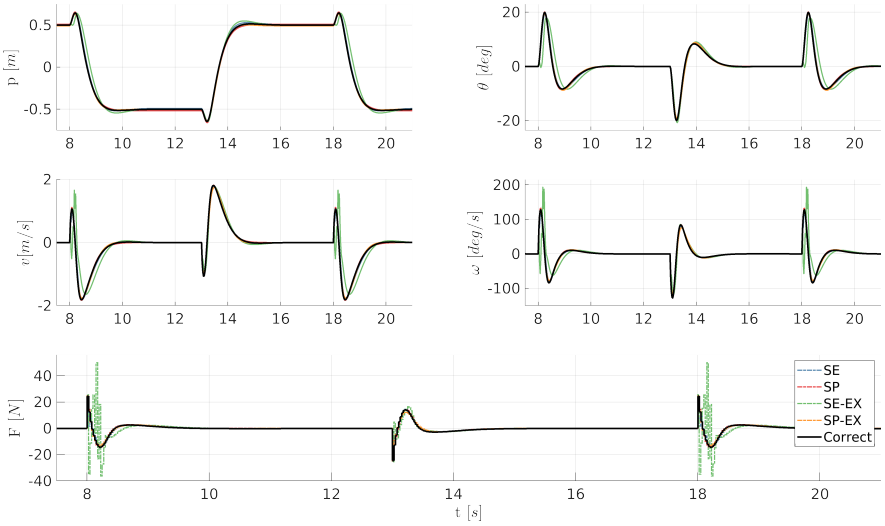
Nominal Length Black-box models

In this case, we have clear indications that the SP kernel is better. At the initial phase of the simulation, the SP-EX kernels follows closely the true trajectory and later completes the task. We can say that it has indeed learnt the mismatch model.



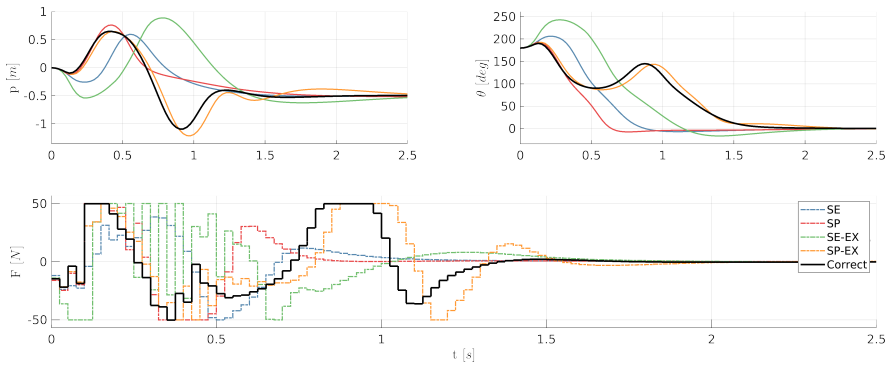
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

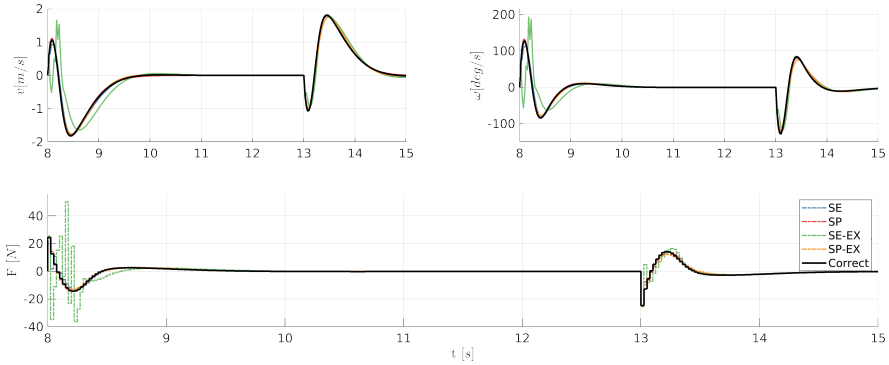


(c) Complete moving cart task.

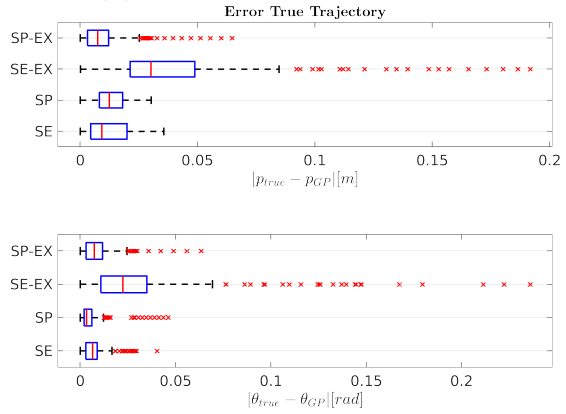
Figure 5.32: Black-Box closed-loop performances.



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.



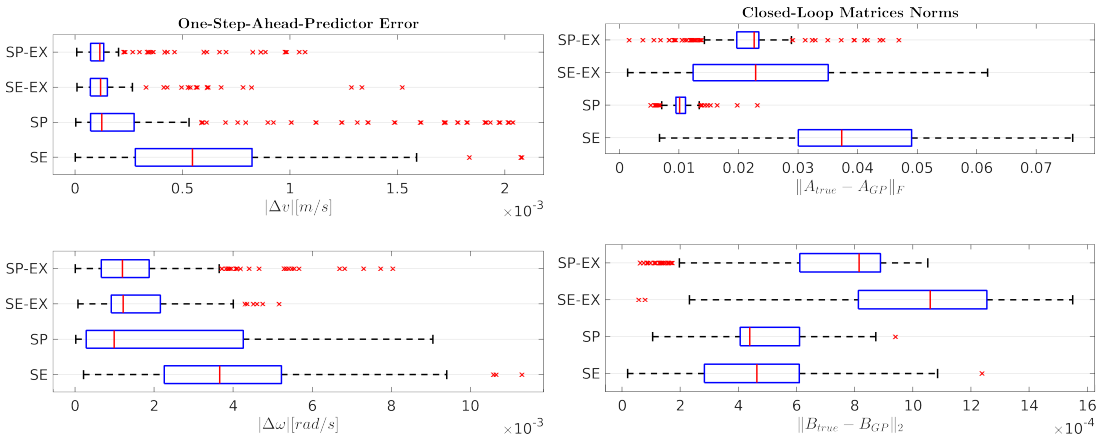
(c) Error with respect to the true trajectory.

Figure 5.33: Black-Box closed-loop trajectories.

Nominal Mass Grey-box models

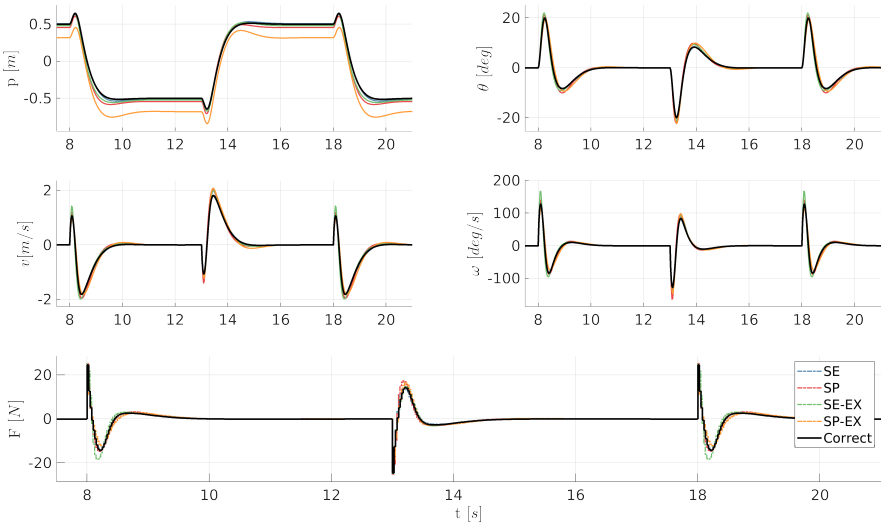
In this simulation, one can clearly see that the SE kernel fails at complete the task. We tried multiple times to optimize the hyperparameters, with the hope of improving its performance kernel without success. We could attribute this to the lack of datapoints, in the case of the SE Black-Box models the lengthscales were in the order of 10^4. From this simulation, we can say the the SP kernel is better at modelling the mismatch model of the nominal model. The metrics are very similar for both kernels. Although the SP fails to reach the reference position,

which results in a constant error, it is much better at the initial swing up of the pendulum. This is a clear indication that it has better generalizations properties: it has indeed learnt the mismatch model.



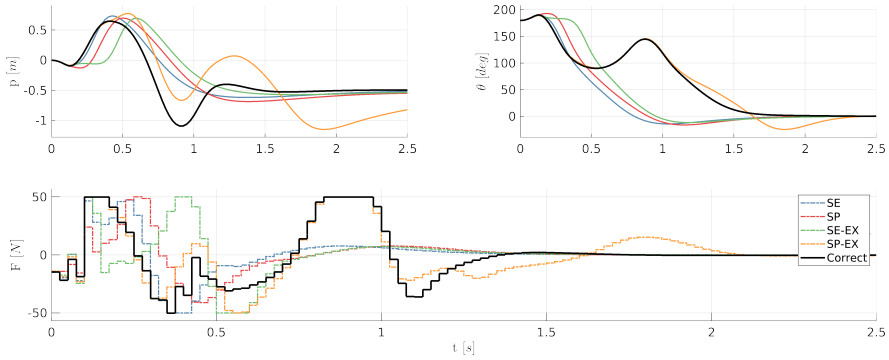
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

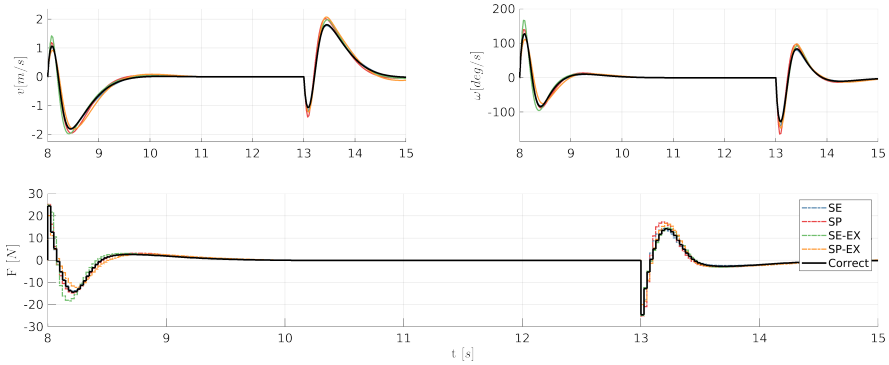


(c) Complete moving cart task.

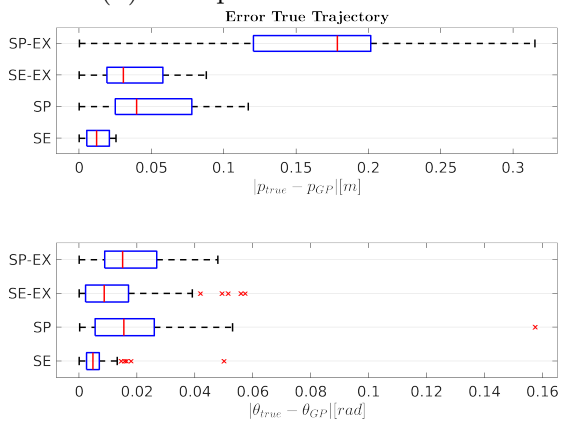
Figure 5.34: Grey-Box closed-loop performances.



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.

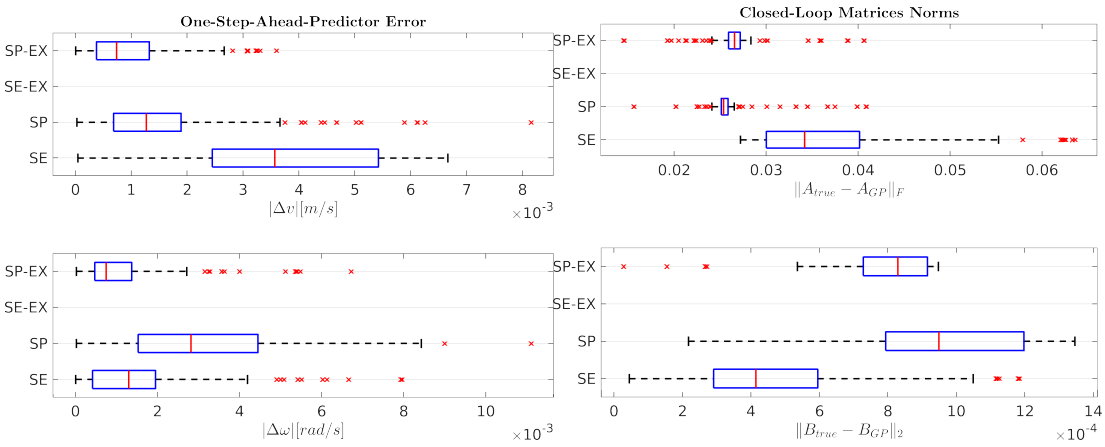


(c) Error with respect to the true trajectory

Figure 5.35: Grey-Box closed loop trajectories.

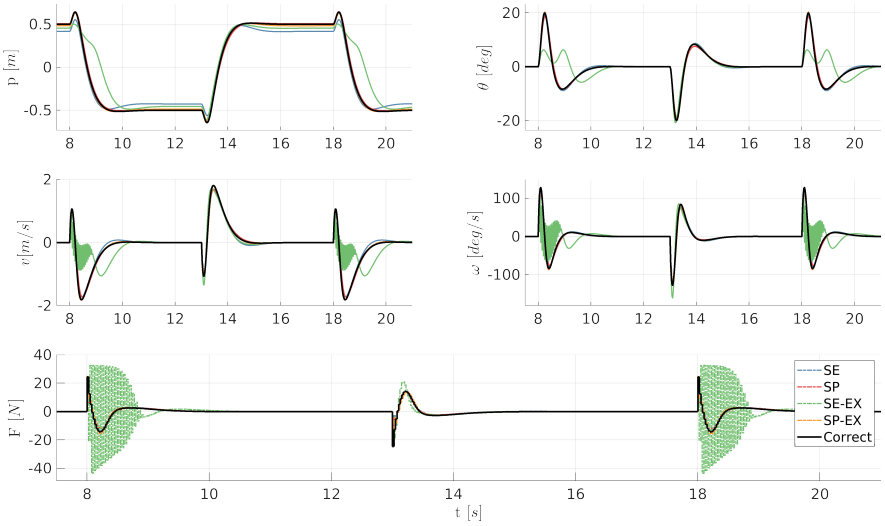
Nominal Mass Black-box models

In this case neither kernels were able to give satisfactory results. All the kernels are wrong about the true system dynamics, we can attribute completing the task to the NMPC controller, not the accuracy on the mismatch model prediction of the kernels.



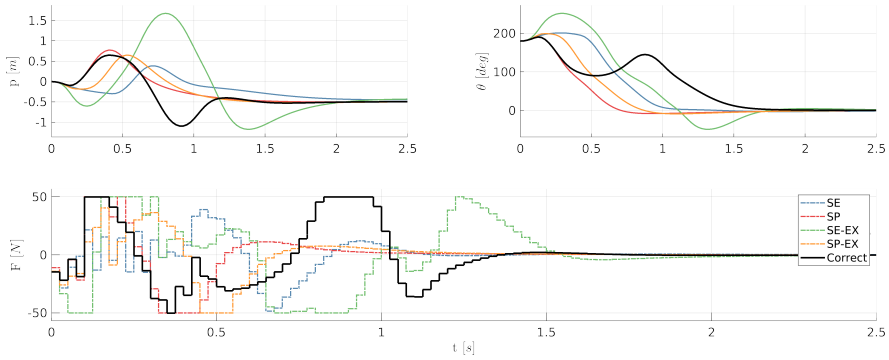
(a) One-Step-Ahead Prediction Error.

(b) Linearization matrices norm.

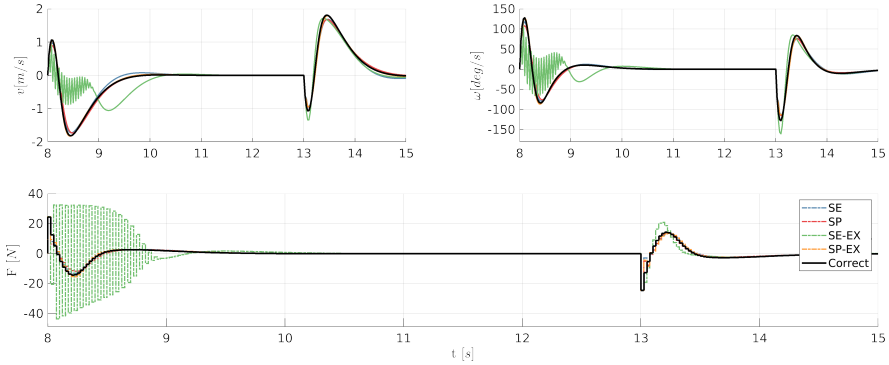


(c) Complete moving cart task.

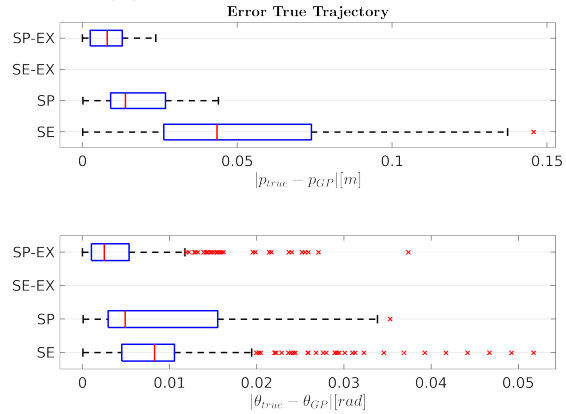
Figure 5.36: Black-Box closed-loop performances.



(a) Cart-pendulum positions.



(b) Cart-pendulum velocities.



(c) Error with respect to the true trajectory.

Figure 5.37: Black-Box closed loop trajectories.

5.5 Discussion

From the previous sections, we can assert that there's an overall improvement of the closed-loop trajectory using gaussian processes to learn the mismatch model. Althouhg, the selection of the best kernel is not quite precise if we were to use the fitting error in the validation set. This lack of criterium to select the right kernel is one limitation of the training procedure, we tried to give some direction by comparing the linearization performances of the kernels in the training sets and validation set. Indeed, we found that lower matrices fitting in the validation

set is strictly correlated to good closed-loop performance, specially for black-box models. The analisis in the swing-up simulation was useful to establish that the GP indeed learns the task from which the datapoints were collected. The results of appying an external input are quite simple to elaborate: more richer information on the training dataset, the better we can generalize. The most interesting results came from the moving cart simulations, where during the initial swing up of the pendulum to reach the reference position, the SP models were clearly comparable to the true trajectory. That is, the SP kernels have learnt the mismatch models and that is something that the SE kernel hasn't achieved neither in grey nor black-box models. By being able to assign negative correlations to the distance between datapoints, the SP kernels are able to model the mismatch dynamics more accurately. This means that our class of prediction models is much richer, which is in accordance to the connection between Gaussian processes and RKHS formalism. Moreover, the mini-batch learning was key to finding the right hyperparameters. By applying gradient-descent to the marginal-likelihood with random subsets, we were doing a random approximation of the gradient loss – i.e. stochastic gradient-descent. This gaves us a way to prevent overfitting during the training of the SP, which was a persistent problem initially. To achieve good closed-loop performances, before applying mini-batch learning, the SP kernels were tuned very carefully by hand and it was quite a troublesome procedure. With mini-batch learning, the process has considerably become easier and faster.

Conclussions

Within this work we evaluated the performance the Generalized Spectral kernel, a class of kernels quite innovative to learn dynamical systems, confronting the proposed kernel with the standard Squared Exponential kernel. We introduced the basis principles for gaussian regression, and based on the connections between GPs and RKHS, we thought the SP kernels were a good alternative to the SE kernels. Our goal became to determine that its properties of being able to approximate arbitrarily well a stationary kernel were linked to an increase in performance of the resulting prediction models – which has indeed been the case. Later, we evaluated their fitting capabilities to determine which one performed better with the available data by training them with a standard swing-up task dataset, achieving similar performances. Given these similar fitting capabilities, to establish a better metric of their generalizations properties, we computed their relative linearization matrices within the training set and compared the distance with respect to the matrices obtained from the true models, which allows us to compare the kernels in a more sistemic way. Moreover, we trained the models with richer datasets that came from exciting the system with an external input during the swing up task with the hope to improve the infering of the mismatch model. We defined two kind of nominal models, one slighly wrong (nominal length model) while the other was much worse modelled (nominal mass model). For the grey-box models, both kernels had similar performances during the learning phase. For the black-box models, the generalized spectral kernel was clearly outperforming the SE kernel. We then applied the trained models to solve two control tasks: swing up of the pendulum and moving the cart with the pendulum stably upwards. During the swing up task, although the SE kernel had a satisfying performance, the closed-loop behaviors were clearly much better with the SP kernels, both for grey and black-box models, achieving closed-loop trajectories that were almost identical to the true model. However, the performances of both kernels weren't far from one another and we couldn't assert which one was better with confidence. The final answer came from the moving cart simulation, although the SE kernel proved to be still a valid choice, the overall closed-loop behavior was in favour of the SP kernel with both nominal models. We attribute this feat to their great flexibility, which allows to model more complex behavior, and accurately learn the mismatch model to compensate the parametric uncertainties present in the nominal model. The most interesting result was their predictive performance in the case of black-box models, where the learnt model was able not only to complete the tasks in both swing up and moving cart simulations - while

the SE kernel struggled - but the initial closed-loop trajectory was clearly similar to the true trajectory in all the models implementing SP kernels. This gave us the strong indication that the gaussian models with generalized spectral kernels had indeed learnt the mismatch model and now, with confidence, we can finally state that the SP kernel demonstrated to be the superior choice for learning the mismatch model.

Further development

The simulations were carried out with the same number of test points for both kernels, which is an essential element for doing predictions with gaussian processes. In particular, one should investigate if a lower number of points gives can mantain the same generalization performances for the SP kernel. Moreover, we didn't apply any technique for reducing the test points, such as inducing points, while it is a standard procedure for prediction with gaussian processes. This would make this class of kernels an even more powerful tool, given that the NMPC controller could solve the OCP much faster with a higher precision. Another key point is the training of the hyper-parameters, given the high-flexibilility of the SP kernel, it can suffer from overfitting. To solve this problem, we introduced mini-batch training to improve to the training of the hyperparameters through the marginal-likelihood. One way that may lead to an improvement of their inference is by training them in a full Bayesian manner, treating the hyperparameters as random variables modelling prior knowledge, to avoid the problems arising from solely minimizing the ML through gradient-descent. Moreover, given that the SP kernel depends heavily on its scales factor, one should try to develop a scheme to infere them effectively, avoiding their annihilation during the training procedure. Still, much more work on determining the learning performances before applying the gaussian models to the NMPC controller is needed. The metrics defined in this work cannot be evaluated in practice, and for now, we solely rely in analyzing simulations for the choosing criteria.

Bibliography

[1] Petter Abrahamsen. *A review of Gaussian random fields and correlation functions*. Vol. 510. Norsk Regnesentral/Norwegian Computing Center Oslo, 1997.

[2] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Vol. 55. US Government printing office, 1964.

[3] Joel AE Andersson et al. "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[4] Yutao Chen et al. "Matmpc-a matlab based toolbox for real-time nonlinear model predictive control". In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 3365–3370.

[5] Yarin Gal and Richard Turner. "Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 655–664.

[6] Lukas Hewing et al. "Learning-based model predictive control: Toward safe learning in control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), pp. 269–296.

[7] Juraj Kabzan et al. "Learning-based model predictive control for autonomous racing". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3363–3370.

[8] Motonobu Kanagawa et al. "Gaussian processes and kernel methods: A review on connections and equivalences". In: *arXiv preprint arXiv:1807.02582* (2018).

[9] Miguel Lázaro-Gredilla et al. "Sparse spectrum Gaussian process regression". In: *The Journal of Machine Learning Research* 11 (2010), pp. 1865–1881.

[10] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[11] Miao Liu et al. "Gaussian processes for learning and control: A tutorial with examples". In: *IEEE Control Systems Magazine* 38.5 (2018), pp. 53–86.

[12] Lennart Ljung. "Perspectives on system identification". In: *Annual Reviews in Control* 34.1 (2010), pp. 1–12.

[13] Lennart Ljung. "System identification". In: *Signal analysis and prediction*. Springer, 1998, pp. 163–173.

[14] David John Cameron Mackay. *Bayesian methods for adaptive models*. California Institute of Technology, 1992.

[15] Alonso Marco et al. "Automatic LQR tuning based on Gaussian process global optimization". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 270–277.

[16] Manfred Morari and Jay H Lee. "Model predictive control: past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[17] Truong X Nghiem. "Linearized gaussian processes for fast data-driven model predictive control". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 1629–1634.

[18] Sebastian W Ober, Carl E Rasmussen, and Mark van der Wilk. "The promises and pitfalls of deep kernel learning". In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 1206–1216.

[19] Joaquin Quinonero-Candela and Carl Edward Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *The Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

[20] Rien Quirynen. "Numerical simulation methods for embedded optimization". In: (2017).

[21] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems* 20 (2007).

[22] Yves-Laurent Kom Samo and Stephen Roberts. "Generalized spectral kernels". In: *arXiv preprint arXiv:1506.02236* (2015).

[23] Johan Schoukens and Lennart Ljung. "Nonlinear system identification: A user-oriented road map". In: *IEEE Control Systems Magazine* 39.6 (2019), pp. 28–99.

[24] Jonas Sjöberg et al. "Nonlinear black-box modeling in system identification: a unified overview". In: *Automatica* 31.12 (1995), pp. 1691–1724.

[25] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[26] Andrew Wilson and Ryan Adams. "Gaussian Process Kernels for Pattern Discovery and Extrapolation". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1067–1075. URL: `https://proceedings.mlr.press/v28/wilson13.html`.

[27] Andrew Gordon Wilson et al. "Deep Kernel Learning". In: vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, 2016, pp. 370–378. URL: https://proceedings.mlr.press/v51/wilson16.html.