Learning-supported and Force Feedback Model Predictive Control in Robotics

Dissertation
zur Erlangung des akademischen Grades

Doktoringenieurin
(Dr.-Ing.)

von **M.Sc. Janine Matschek**
geb. am 28.01.1989 in Schönebeck, Deutschland

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr.-Ing. Rolf Findeisen
Prof. Dr.-Ing. Sandra Hirche
Prof. Daniel Limón Marruedo

Promotionskolloquium am 31.05.2021

"*It is good to love many things, for therein lies the true strength, and whosoever loves much performs much, and can accomplish much, and what is done in love is well done.*"

Vincent van Gogh, Dutch painter (1853 – 1890)

Contents

Abstract

Autonomous dynamical systems enter our daily lives and homes. They appear in the form of self-driving cars, vacuum cleaning robots, and smart manufacturing cobots. Hence, these systems inevitably interact with humans or their surroundings. Safety and reliable performance are two critical aspects, which autonomous systems must meet. These goals can only be achieved by a tight interconnection between several technologies. This includes, among others, the dynamic control of systems and the ability of these systems to learn from interactions. This thesis aims to interlink these two concepts more closely. In particular, we propose to equip model-based controllers with a measure and awareness of the system surroundings. This goal is tackled from three perspectives:

First, model predictive control schemes for direct force control are proposed. This tailored model predictive control formulation regulates the interaction forces between a cobot and its environment. Since these forces quantify the interaction of the robotic system with workpieces, other robots, or humans, direct control of contacts is achieved. We hereby utilise the benefits of model predictive control, which allows handling the nonlinearities of robotic systems, includes prior knowledge in terms of models, and explicitly achieves constraint satisfaction. We show how constraints on the interaction forces allow for improved performance and increased safety. They allow guaranteeing tight contact without exceeding safety-critical force limitations. Consequently, the developed model predictive force controller enable to extend the usage of cobots to applications that might have been considered too delicate or dangerous until now.

Second, this thesis examines data- and learning-supported predictive force controllers. We equip the developed model predictive force controllers with machine learning models of the interaction. The contact forces are learned via Gaussian processes, which perform well despite noise in sensor data and allow for incorporating prior knowledge about the interaction. Inspired by force control, we develop a generalised concept. We present how Gaussian processes can be included as system output models in predictive controllers, while we provide closed-loop guarantees. The proposed concepts of output learning for predictive controllers constitute a promising step towards the integration of control engineering and computer science. In particular, we achieve increased autonomy of robotic systems that interact with diverse environments via the proposed algorithms. We enable the transfer of predesigned robot control setups to a large variety of applications and environments with the developed learning-supported force controller.

Third, this thesis introduces Gaussian processes reference generators for predictive

controllers to address the learning from interactions. Since model predictive controllers rely on predictions of future evolutions, the knowledge of the desired motions over the prediction horizon is required. In many cases, these references are not known a priori but encoded by data and obtained via communication and interaction with other systems. Gaussian processes can be used to model, filter, and predict these signals such that the model predictive controller can proactively and foresightedly steer the system to follow the reference. This thesis proposes constrained learning algorithms for the data-based reference generation by Gaussian processes. The proposed constraints in the training phase of Gaussian processes encode trackability conditions. We guarantee the recursive feasibility of learning-supported model predictive control despite uncertainties. The developed reference learning scheme for predictive control tackles the integration of machine learning and control by including systems theory into the data-based training of Gaussian processes. Hence, we achieve the incorporation of extensive prior knowledge in machine learning to obtain reliable, realistic, and safe references and controllers.

These three derived concepts for interactive and learning-supported model predictive control are illustrated and evaluated in simulations and experiments. In particular, real-time feasible implementations on lightweight robots are successfully conducted, which underline the respective benefits of the developed approaches and prove the applicability of our theoretical findings to real-world problems.

This work shows that model predictive controllers are very well suited to operate systems in safety-critical and delicate tasks if the system's interaction with the surroundings is taken into account. We show that the awareness of interaction allows systems to learn from the environment and to increase their autonomy while ensuring safety.

Deutsche Kurzfassung

Autonome dynamische Systeme wie beispielsweise selbstfahrende Autos, Staubsauger-roboter und intelligente Fertigungsroboter halten Einzug in unser tägliches Leben. Diese Systeme interagieren unweigerlich mit dem Menschen oder seiner Umgebung, wodurch deren Zuverlässigkeit sowie Sicherheit unerlässlich wird. Diese beiden Ziele können nur durch eine enge Verzahnung der eingesetzten Technologien erreicht werden. Schlüsselkomponenten sind dabei die Regelung von Systemen und die Fähigkeit dieser Systeme, aus Interaktionen zu lernen. Ziel der vorliegenden Arbeit ist es zu untersuchen, wie sich diese beiden Konzepte mit Garantien an die Sicherheit verschmelzen lassen. Insbesondere wird in dieser Arbeit aufgezeigt, wie sich modellbasierte Regler mit einer Kenntnis beziehungsweise einem Bewusstsein für ihre Systemumgebung ausstatten lassen. Die genannten Herausforderungen werden in drei Teilen adressiert:

Zunächst werden modellprädiktive Reglerformulierungen für die direkte Kraftregelung vorgeschlagen. Eine neuartige modellbasierte Formulierung erlaubt es, Interaktionskräfte zwischen Robotern und ihrer Umgebung explizit zu berücksichtigen. Da diese Kräfte die Interaktion des Robotersystems mit Werkstücken, anderen Robotern oder Menschen beschreiben, ermöglicht dies eine direkte und gezielte Regelung ihrer Kontaktkräfte. Hierbei werden die Schlüsselvorteile der modellprädiktiven Regelung genutzt, die es erlauben, Nichtlinearitäten von Systemen zu berücksichtigen, Vorwissen in Form von Modellen einfließen zu lassen und Beschränkungen einzuhalten. Es wird gezeigt, wie eine Beschränkung der Interaktionskräfte eine verbesserte Regelperformance und eine erhöhte Sicherheit ermöglicht. So kann mit dem Ansatz ein gewünschter Kontakt garantiert werden, ohne sicherheitskritische Kraftbegrenzungen zu überschreiten. Dieser modellprädiktive Kraftregler ermöglicht den Einsatz von Robotern in Anwendungen, die bisher als zu kritisch oder gefährlich galten.

Zweitens werden in dieser Arbeit daten- und lerngestützte prädiktive Kraftregler untersucht und entwickelt. Hierzu wird der vorgeschlagene modellprädiktive Kraftregler um maschinelle Lernmodelle, welche die Interaktion beschreiben, erweitert. Die Kontaktkräfte werden dabei über Gaußsche Prozesse gelernt, welche Messrauschen explizit berücksichtigen und es erlauben, Vorwissen direkt einzubeziehen. Diese praktische Fragestellung wird abstrahiert und es wird aufgezeigt, wie sich Gaußsche Prozesse zum Erlernen eines Ausgangsmodells verwenden und in prädiktive Regler integrieren lassen, ohne Garantien für den geschlossenen Regelkreis zu verlieren. Die vorgeschlagenen Konzepte zum Lernen des Ausgangsmodells für prädiktive Regler stellen einen vielversprechenden ersten Schritt zur Verschmelzung von Regelungs- und Informationstechnik dar. Insbesondere erlauben die vorgeschlagenen Verfahren eine erhöhte

Autonomie von Robotersystemen, die in wechselnden Umgebungen interagieren.

Drittens wird in dieser Arbeit aufgezeigt, wie sich Referenzen für prädiktive Regler mittels Gaußscher Prozesse generieren lassen, die es ermöglichen, aus Interaktionen zu lernen. Da modellprädiktive Regler auf der Vorhersage zukünftiger Entwicklungen beruhen, ist es notwendig und vorteilhaft, dass die Referenzbewegung über den Prädiktionshorizont weitgehend bekannt ist. In vielen Fällen sind diese Referenzen jedoch a priori unbekannt beziehungsweise nur in Form von Messdaten verfügbar. Gaußsche Prozesse erlauben es, diese Signale zu modellieren, zu filtern und vorherzusagen, sodass der prädiktive Regler das System proaktiv und vorausschauend steuern kann, um dem Referenzsignal zu folgen. In dieser Arbeit werden neue Verfahren zum datenbasierten Lernen von Referenzen mittels Gaußscher Prozesse entwickelt. Die Verfolgbarkeit der gelernten Referenz durch das System wird hierbei durch geeignete Beschränkungen während des Lernens realisiert. Diese Beschränkungen garantieren die wiederholte Lösbarkeit der lerngestützten modellprädiktiven Regelung trotz Unsicherheiten. Das entwickelte Verfahren kombiniert maschinelles Lernen mit grundlegenden Konzepten der Regelungstechnik. Es bettet systemtheoretische Eigenschaften in das datengestützte Training Gaußscher Prozesse ein. Dies ermöglicht die Integration von Vorwissen in maschinelle Lernalgorithmen, um zuverlässige und realistische Referenzen für eine sichere und flexible Regelung zu erhalten.

Die entwickelten Konzepte zur interaktiven und lerngestützten modellprädiktiven Regelung werden in Simulationen und in realen Experimenten veranschaulicht. Echtzeitfähige Umsetzungen für Leichtbauroboter unterstreichen den Nutzen der entwickelten Ansätze und untermauern die Anwendbarkeit der theoretischen Ergebnisse.

Diese Arbeit zeigt auf, dass modellprädiktive Regler sehr gut für die Regelung sicherheitskritischer und schwieriger Aufgaben geeignet sind, wenn die Interaktion des geregelten Systems mit der Umgebung berücksichtigt wird. Die explizite Betrachtung von Interaktionen des Systems ermöglicht es, von der Umgebung zu lernen und hierdurch die Autonomie des Systems zu erhöhen, ohne die Sicherheit zu gefährden.

1 Introduction

To improve their performance, decrease costs, and enhance usability, there exists a strong desire to make systems "intelligent and autonomous". This desire spans from industrial production chains related to Industry 4.0, to self-driving cars, and home automation. A key element to achieve autonomy is, and has been, the concept of feedback and control. Feedback control allows systems to react to unplanned events, disturbances, and changing references. Machine learning, which currently receives considerable attention, tackles the same tasks – to increase the autonomy of systems and to equip them with intelligence. Driven by the availability of large amounts of data, machine learning gains more and more popularity to model systems and learn relationships between data. As machine learning and control have common goals, one should focus on merging them instead of regarding them independently from each other. Doing so allows fusing their benefits and enables new developments. This work combines methods from control theory and machine learning. Mainly we will focus on model predictive control strategies on the control side and Gaussian processes on the machine learning side. In particular, we propose new methods for Gaussian processes learning by including notions from systems theory, hence, including and exploiting control theory in machine learning. Vice versa, we support model predictive controllers with Gaussian process models, thus explicitly including learning in control, cf. Figure 1.1.

Model predictive control is an advanced control strategy that allows handling nonlinearities, multiple inputs and outputs, time-delayed systems, constraints, and various task formulations such as setpoint regulation or path following [4, 28, 83, 189]. Due to this versatility, it has been used in numerous applications spanning from process systems industry [186], and biomedical applications [30], to automotive engineering [15, 100]. Often, the satisfaction of constraints (for example on the temperatures in reactors, the insulin delivery rate in the artificial pancreas, or the state of charge in batteries of electric cars) is substantial for safe operation. Besides invariance-based approaches that ensure constraint satisfaction via control invariance of a safe set [66, 113, 123], model predictive controllers explicitly allow to take these requirements into account by including them in a constrained optimal control problem. The cost function optimises the future system evolution based on a defined measure and system model. This measure can, for example, penalise the deviation of a system from a desired setpoint, from a time-dependent trajectory, or from a geometric reference path. For each of these references, specific formulations of model predictive controllers are available and come along with a corresponding stability theory

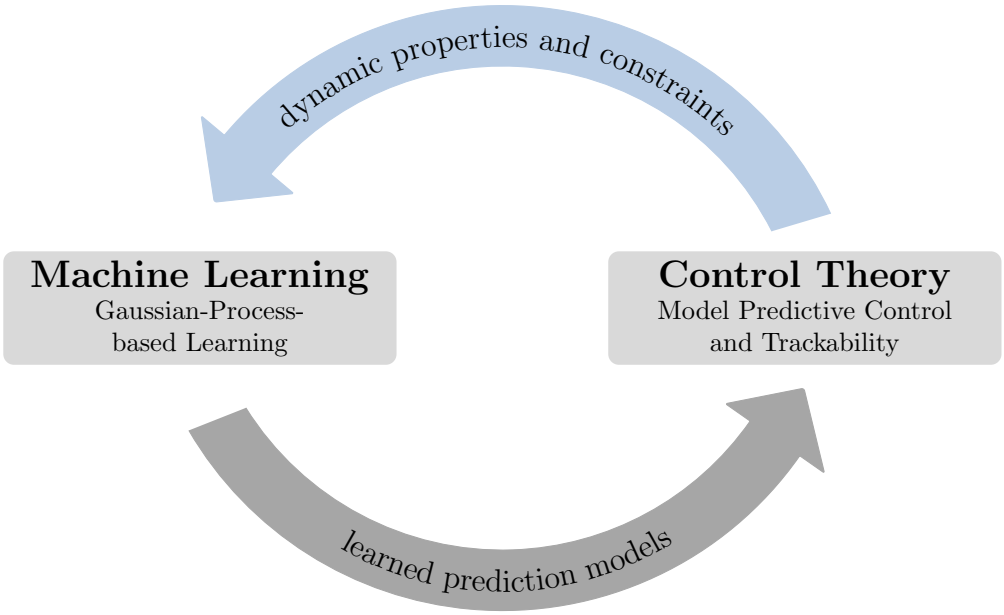


Figure 1.1: Coupling and fusing machine learning and control theory as considered in this thesis.

[56, 136, 139, 140, 153, 162]. The predictions of the system are influenced by the control inputs that the controller chooses. Once the controller has determined the optimal input over the prediction horizon, it applies the first part of it to the real process. Feedback from the plant is included by repeating this open-loop optimisation at every sampling instance with new initial conditions obtained via measurements from the real system.

The quality of the involved predictions has a large influence on the performance of the closed-loop. The key influencing factors are the system model and the controller's knowledge about external signals such as references or disturbances. Often, it is assumed that these models and references are known a priori. They might be provided by higher-level planners, modelled via first principles, or obtained via system identification. However, often models are uncertain. Thus, machine learning methods are increasingly incorporated into the loop to account for limited insights into the processes. For example, system models for prediction can be identified via neural networks or Gaussian processes [29, 42, 76, 93, 118–120, 135, 171, 180, 223, 237]. Model predictive controllers provide the flexibility to include such models.

Gaussian processes build a particular class of these machine learning techniques that can be used for regression or system identification [117, 235]. They are based on conditional probabilities as well as Gaussian distributions and extend multivariate normal distributions to the infinite-dimensional case. Due to their stochastic nature, Gaussian processes often work particularly well with data that is corrupted with noise. They can directly take the uncertainty in the measurements into account and are considered not to be prone to over-fitting. Moreover, they allow predicting posterior

distributions of the variables of interest, such that information about the model quality can be inferred. In model predictive control, Gaussian processes have been used to, for instance, provide disturbance models, learn dynamical system representations, or extend existing first-principle models in a hybrid setting [93, 177, 180, 223, 231, 237].

1.1 Contribution

In this work, Gaussian processes will be merged with model predictive control in two ways: The first approach uses Gaussian processes to obtain the output equations of dynamical systems. The resulting hybrid model, consisting of a first-principle dynamic state equation and the data-based output equation, is embedded into a model predictive control framework. The focus on the learning of output equations instead of the full dynamical model is motivated by an application example, where uncertainty mainly arises in the output. We consider a robotic application example, where a model predictive controller is employed to control the robot's position as well as the interaction forces of the robot with its environment. This task is first addressed by a pure "white-box" modelling approach. It shows the benefits of a new model predictive force controller formulation, ensuring safe operation. This safety is accomplished via a provably stable controller formulation, which takes constraints on positions and forces into account. This way, safe interaction of the robot and its environment is achieved, which is underlined by several simulation and hardware experiments on a lightweight robot.

Operating the same robot and controller in different environments, however, requires extensive recalibration and design decisions even though the parameters and dynamics of the robot are unchanged. Hence, we propose an extension to this purely first-principles-based controller design by including machine learning in the model predictive controller to represent changing environmental conditions. The data-based formulation enters the output of the dynamical system only. This structure reveals useful properties compared to dynamical systems modelling via Gaussian processes, which can be exploited in the controller design. Among others, it allows extending the established stability conditions of the first-principles-based formulation to learning-supported model predictive control.

The second way of merging Gaussian processes and model predictive control in this thesis considers the learning of external signals such as references for the prediction inside the controller. Even though in many cases references are a priori known, this is not true for all applications. For example, a corporation between multiple robots can require the negotiation of desired motions via data transmissions. Suppose one robot should synchronise its movement to another robot or human. In that case, sensory data about the robot's surroundings encode its reference. Thus, the reference is a priori unknown and might be corrupted by noise. Learning of the reference via Gaussian processes allows us to take the uncertainty into account. It enables the filtering, pre-

diction, and, if necessary, modification or adaptation of the reference. A constrained learning and update scheme for Gaussian processes is proposed. It guarantees that the learned references are suitable for the model predictive controller and ensure recursive feasibility and closed-loop stability. This way, model predictive controller formulations are facilitated that can exploit their full potentials. Besides the closed-loop guarantees, performance improvements are enabled via the prediction of the reference by Gaussian processes, which allows a foresighted control. This concept of reference learning is illustrated with several simulation examples as well as a hardware implementation on a robot used in a minimally invasive medical treatment. This application example requires precise positioning of surgical instruments with respect to a patient position. The robot must accurately synchronise its movements to respiratory motions of patients while handling tools such as needles inserted in the patient's body. As shown, the learning-supported controller can meet the required sub-millimetre precision by the combination of reference learning, prediction, and constrained optimisation.

In a nutshell, this work entails three main contributions, which are

- Interaction force learning and control while guaranteeing force limitation via model predictive control for safe robot interaction.
- Constrained Gaussian process learning for prediction of references suitable for model predictive control guaranteeing trackability and recursive feasibility.
- Experimental validation of the two proposed learning-supported model predictive control schemes on robot force control and robotic respiratory motion compensation.

Overall, the proposed learning-supported model predictive control schemes allow for reliable and safe control of dynamical systems with improved control performance via learning and adaptation of reference and output models based on noisy data.

1.2 Outline

The structure of this thesis is illustrated in Figure 1.2. A general introduction to model predictive control is provided in Chapter 2. Particular emphasis is laid on the formulation of model predictive controllers tailored for tasks spanning from setpoint regulation, over trajectory tracking to path following, as also published in [153]. Besides providing a mathematical description of each control task, stability conditions are given for regulation, tracking, and path following, respectively. Similarities, connections, and differences between these formulations are illustrated by means of robotic simulation examples. The bottom line of this comparison is to encourage the selection of the controller formulation fitting for the specific task at hand.

In Chapter 3, a model predictive controller is proposed for the hybrid control of positions and forces in robotics. This setup entails the benefits of taking the system

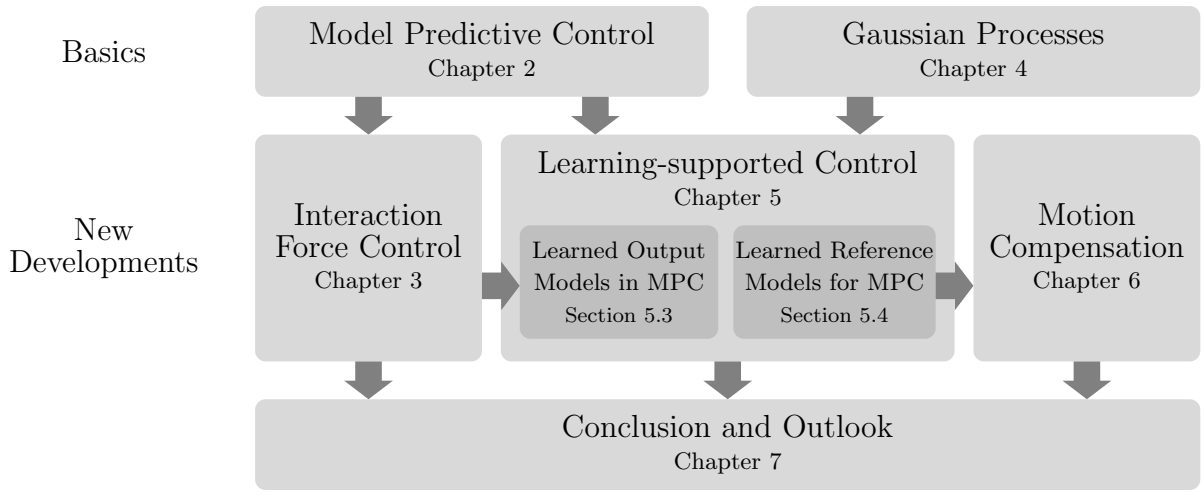


Figure 1.2: Structure of the remainder of this thesis.

dynamics as well as the environmental interactions of the robot into account. By doing so, it accomplishes the desired interaction force and motion. Safety of the proposed approach is ensured via the explicit consideration of motion and force constraints, which allow establishing the desired contact with guaranteed force limits from above and below. The results have been published in [154].

Chapter 4 provides an introduction to Gaussian processes, which are stochastic machine learning techniques. Besides the basic principles and mathematical formulations for the learning and prediction with Gaussian processes, we highlight some properties that are especially interesting from the control perspective. Among others, their differentiability is exploited in the following chapters. Based on the foundation laid by Chapters 2 and 4, learning-supported model predictive control strategies can be derived thereafter.

Chapter 5 proposes strategies of learning-supported model predictive control with Gaussian processes. To do so, a general introduction into learning-supported model predictive control is given. We propose two specific learning-supported controller setups. First, Gaussian processes are used inside a predictive control formulation to address uncertain output functions. Starting from the motivating example of contact force control in robotics from Chapter 3, an adaptive, supervised, offline-learning approach is presented, which guarantees closed-loop stability and increases the performance of the controller. The proposed methodology is evaluated in real time via a hardware implementation on a lightweight robot while providing comparisons to first principle models. The second learning-supported control formulation includes dynamical system properties in the training of Gaussian processes. Supervised offline and online learning of references from data is considered, which ensures trackability of the learned references. To do so, constrained training of Gaussian processes is proposed, which guarantees the recursive feasibility of a tracking predictive controller using the learned reference. The outlined findings led to a series of publications [156–158, 160].

In Chapter 6, the proposed reference learning approach is used to control a lightweight robot in a medical application. Here, respiratory motion signals from a patient are learned, smoothed, predicted, and tracked by the learning-supported controller using online data updates. Superior prediction performance of the constrained Gaussian process compared to unconstrained learning and standard filtering techniques is shown. Moreover, the learning-supported model predictive controller achieves sub-millimetre accuracy thanks to the provably trackable reference prediction.

Finally, Chapter 7 summarises the proposed algorithms and achieved results. It reviews the established combination of control and machine learning via Gaussian-process-supported model predictive control. The inclusion of learning into control via Gaussian process output models, as well as the inclusion of control in learning via constrained Gaussian process training conditioned on trackability properties is discussed. An outline of interesting future research directions concludes this thesis.

2 Model Predictive Control

This chapter outlines and compares model predictive control approaches for setpoint regulation, trajectory tracking, and path following. Besides, the basic conditions for stability and recursive feasibility are reviewed. Simulation studies from robotics highlight the differences between the approaches and provide insights into the controller design for the different tasks.

2.1 Introduction and Setup

Model predictive control (MPC) is an optimal control strategy that enjoys great popularity in both industry and academia [64, 162]. This popularity originates from advantages such as the ability and flexibility to explicitly handle soft and hard constraints, nonlinear system dynamics, and multi-input-multi-output (MIMO) systems. Furthermore, a wide range of options is available to ensure stability, which builds the basis for several extensions to robust and stochastic control in the presence of uncertainty [64, 136, 162–164, 167, 187, 189]. From a practical side, MPC often leads to explainable behaviour of the system under control since it is model-based. The involved variables remain interpretable, which simplifies the tuning of the controller. In general, the basic principle of MPC is easily comprehensible. At the same time it often outperforms classical controllers as PID for complex problems such as the control of constrained nonlinear MIMO systems.

The basic idea in MPC is the repeated solution of an optimal control problem to obtain an optimal input for the system under consideration. The goal is to minimise a cost function that represents the desired behaviour of the system over a given finite prediction horizon. For example, the cost function may penalise the deviation of the system states from a given setpoint over the prediction horizon. Besides the cost function, constraints can be taken into account in the optimisation either for the full prediction horizon or at specific time instances, such as the beginning (initial condition) and at its end (terminal condition). Once an optimal input signal over the prediction horizon is obtained, only the first part of the optimal input is applied to the system. The prediction and optimisation are performed again with a receding prediction horizon in the next iteration. To be able to predict the system behaviour over the horizon, a system model is needed. This prediction model enables to evaluate the performance of the system under specific input signals not only for the current time instance but also for future times. The prediction allows the MPC to be "foresightful", which allows, for example, to reduce overshooting and oscillations. It avoids constraint

violation and prevents the controller from being short-sighted. The model quality has a significant effect on the closed-loop performance of the controller. Since the calculation of the optimal input is based on the predicted system behaviour, it is sensitive to prediction errors originating from an imprecise system representation. Therefore, one of the drawbacks of MPC is the necessity to have a suitable model that captures the dynamic system as precise as possible. Obtaining such a precise system model is, in general, not trivial. It needs insight into the underlying physical relationships and processes as well as sufficient measurement data. Though, the system model should not be too complex in terms of system order and degree of nonlinearity, as this increases the computational demand. This issue is related to another drawback of MPC, as the repeated online solution of a possibly nonconvex optimal control problem can require high computational power to meet real-time requirements. Besides its effect on the performance, the system model used in MPC also plays an essential role in closed-loop stability. In nominal MPC schemes, stability proofs assume that the model is a perfect representation of the system. Inherently, nominal MPC has certain robustness against modelling errors as the repeated optimisation includes knowledge about the existing system in terms of initial conditions each time new measurements are available. Based on these general ideas, we can specify model predictive controllers more formally [153]:

We consider nonlinear, continuous-time, and time-invariant systems of the form

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0, \tag{2.1a}$$

$$y(t) = h(x(t)), \tag{2.1b}$$

where $t \in \mathbb{R}$ is the time, $x(t) \in \mathbb{R}^{n_\mathrm{x}}$ denotes the state, and $u(t) \in \mathbb{R}^{n_\mathrm{u}}$ is the input of the system. The dimensions of the state and input are n_x and n_u, respectively. The map $f : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$ describes the dynamics of the system. The mapping $h : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{y}}$ represents the relationship from states to outputs $y(t) \in \mathbb{R}^{n_\mathrm{y}}$ with dimension n_y*. In the context of this work, the system outputs y do not necessarily correspond to the taken measurements but refer to the variables of interest.

We consider *restricted* systems, where additional constraints have to be taken into account. One might consider constraints due to safety reasons by excluding some regions of the state, input, or output space which are considered dangerous for the system under control or its environment. Moreover, constraints can restrict the considered area where a model is valid or represent limitations in the available outputs. They can also be used to influence the controller or closed-loop performance via tuning or accelerate the solution if the search space is narrowed down. These restrictions can take the form of state, input, and output constraints which are represented in this work via the sets $\mathcal{X} \subseteq \mathbb{R}^{n_\mathrm{x}}$,$\mathcal{U} \subseteq \mathbb{R}^{n_\mathrm{u}}$ and $\mathcal{Y} \subseteq \mathbb{R}^{n_\mathrm{y}}$, respectively.

In this work, we rely on the following assumptions on the restricted system (2.1):

*In general, the output function can also depend on the input u, which is not considered here for simplicity.

Assumption 1. *The state and output constraint set $\mathcal{X}$ and $\mathcal{Y}$ are closed, the input constraint set $\mathcal{U}$ is compact.*

Assumption 2. *The system dynamics $f : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$ and the output map $h : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{y}}$ are assumed to be sufficiently often continuously differentiable and locally Lipschitz for all $(x, u)^\top \in \mathcal{X} \times \mathcal{U}$.*

Assumption 3. *For any continuous input signal $u(\cdot)$ and for all initial points $x_0 \in \mathcal{X}$, system (2.1) admits a unique absolutely continuous solution.*

Model predictive control can directly address the control of a constrained system. It includes the restrictions and the dynamic system (2.1) in the optimal control problem, in which a cost function is minimised. The reference definition influences the form of the optimal control problem, which in consequence also influences the way how to provide stability guarantees. One can use different formulations depending on the reference definitions, i.e. if the reference is constant (setpoint), explicitly time-dependent (trajectory) or a geometric path. In the following, model predictive controllers for the different references and goals are outlined. The presentation is mainly based on [153].

2.2 Setpoint Regulation

In setpoint regulation, the goal is to bring the controlled variables of a system to a constant reference value $r_\mathrm{s} \in \mathcal{Y}$. This task often occurs in practice when a system should be operated at a steady state. Application examples of MPC for setpoint regulation are temperature control in a greenhouse [48], the control of liquid level and temperature in a continuous stirred tank reactor [203], or the control of the oxygen level in a coke production [240]. The control error in setpoint regulation problems is given by

$$e_\mathrm{s}(t) := r_\mathrm{s} - y(t). \tag{2.2}$$

Here, the controlled variable $y(t)$ is subtracted from the time-independent reference value or *setpoint* r_s. Naturally, the reference should be designed such that it can be reached and maintained by the system output, which is outlined below.

2.2.1 Definition of Setpoints

Often, the setpoint is a steady state. Thus, bringing the control error to zero by driving the output $y(t)$ to the setpoint r_s corresponds to steering the state $x(t)$ to a value x_s that satisfies $r_\mathrm{s} = h(x_\mathrm{s})$. The associated input to keep the system at x_s is denoted by u_s such that $0 = f(x_\mathrm{s}, u_\mathrm{s})$ with $x_\mathrm{s} \in \mathcal{X}$ and $u_\mathrm{s} \in \mathcal{U}$. Note that multiple u_s and x_s might exist such that $r_\mathrm{s} = h(x_\mathrm{s})$ and $0 = f(x_\mathrm{s}, u_\mathrm{s})$ are fulfilled. One way to obtain the target state x_s and the corresponding reference input u_s for a given $r_\mathrm{s} \in \mathcal{Y}$ is the formulation of the following optimisation problem

$$
\begin{aligned}
(x_\mathrm{s}, u_\mathrm{s}) &= \arg\min_{x,u} \; ||u|| \\
\text{subject to} \\
0 &= f(x,u), \\
r_\mathrm{s} &= h(x), \\
x &\in \mathcal{X}, \quad u \in \mathcal{U}.
\end{aligned}
\tag{2.3}
$$

Note that minimizing $||u||$ is a design decision to obtain a solution which requires minimal actuation and that other optimization criteria could be chosen as well. If $x_\mathrm{s} \in \mathcal{X}$, $r_\mathrm{s} \in \mathcal{Y}$, and $u_\mathrm{s} \in \mathcal{U}$, we call the setpoint a feasible setpoint.

Assumption 4. *We assume that there exists a solution to the regulation problem, i.e. $x_s \in \mathcal{X}$, $r_s \in \mathcal{Y}$, and $u_s \in \mathcal{U}$ are given.*

2.2.2 Model Predictive Control for Setpoint Regulation

As the goal of the controller design is to bring the output error to zero, the cost functional of the model predictive controller can be formulated to minimise $e_\mathrm{s}(t)$. Additionally the input error $w_\mathrm{s}(t) = u_\mathrm{s} - u(t)$ with $w_\mathrm{s}(t) \in \mathbb{R}^{n_\mathrm{u}}$ and the state error $\varepsilon_\mathrm{s}(t) = x_\mathrm{s} - x(t)$ with $\varepsilon_\mathrm{s}(t) \in \mathbb{R}^{n_\mathrm{x}}$ can be penalised. In this context, a typical cost functional for the setpoint stabilisation problems is

$$
J_\mathrm{s}\left(\bar{e}_\mathrm{s}, \bar{w}_\mathrm{s}, \bar{\varepsilon}_\mathrm{s}\right) := \int_0^T L_\mathrm{s}\left(\bar{e}_\mathrm{s}(\tau), \bar{w}_\mathrm{s}(\tau)\right) \mathrm{d}\tau \; + E_\mathrm{s}\left(\bar{\varepsilon}_\mathrm{s}(T)\right). \tag{2.4}
$$

Here, $L_\mathrm{s} : \mathbb{R}^{n_\mathrm{y}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}_0^+$ denotes the stage cost or cost function, $E_\mathrm{s} : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}_0^+$ is called terminal penalty, and T is the prediction horizon. In this work, sampled-data MPC formulations are used [63], as they build a bridge between real-world systems operating in continuous time and discrete implementation of controllers.

The optimal control problem that is solved at every sampling instant t_k can be formulated as

$$
\min_{\bar{u} \in \mathcal{C}} \; J_\mathrm{s}\left(\bar{e}_\mathrm{s}, \bar{w}_\mathrm{s}, \bar{\varepsilon}_\mathrm{s}\right) \tag{2.5a}
$$

subject to $\forall \tau \in [0, T]$

$$
\begin{aligned}
\dot{\bar{x}}(\tau) &= f(\bar{x}(\tau), \bar{u}_k(\tau)), \quad \bar{x}(0) = x(t_k), && \text{(2.5b)} \\
\bar{e}_\mathrm{s}(\tau) &= r_\mathrm{s} - h(\bar{x}(\tau)), && \text{(2.5c)} \\
\bar{w}_\mathrm{s}(\tau) &= u_\mathrm{s} - \bar{u}(\tau), && \text{(2.5d)} \\
\bar{\varepsilon}_\mathrm{s}(\tau) &= x_\mathrm{s} - \bar{x}(\tau), && \text{(2.5e)} \\
\bar{x}(\tau) &\in \mathcal{X}, \; \bar{u}(\tau) \in \mathcal{U}, \; h(\bar{x}(\tau)) \in \mathcal{Y}, && \text{(2.5f)} \\
\bar{x}(T) &\in \mathcal{F}_\mathrm{s}. && \text{(2.5g)}
\end{aligned}
$$

Here, (2.5b) defines the system dynamics with the initial condition $x(t_k)$. We use an overline $\bar{\cdot}$ to indicate predicted signals. The prediction horizon starts at 0 instead of t_k for notational simplicity and since the system and cost function are time-invariant. We also ensure that the minimum is attained using min instead of inf in (2.5a). The constraints (2.5c), (2.5d) and (2.5e) can be regarded as an extended system output defining the output, input and state error, respectively. Note that both the desired state x_s and input u_s are based on the reference r_s and therefore posses an inherent dependency on it. In (2.5f), the state, input, and output constraints are covered and (2.5g) restricts the state at the end of the prediction horizon to be inside the terminal region $\mathcal{F}_\mathrm{s}$, which is often used to ensure stability [56, 163]. This terminal region $\mathcal{F}_\mathrm{s}$ should be a subset of the state constraint set $\mathcal{X}$ and the pointwise preimage of the output constraint set $\mathcal{Y}$, i.e. $\mathcal{F}_\mathrm{s} \subseteq \mathcal{X} \cap h^{-1}(\mathcal{Y})$. As pointed out in [56], the pointwise preimage is not necessarily simply connected and in general hard to obtain. However, by definition $x_\mathrm{s} \in \mathcal{X} \cap h^{-1}(\mathcal{Y})$ so that the intersection is at least non-empty.

2.2.3 Stability of MPC for Setpoint Regulation

It is well known that the repeated application of open-loop optimal control policies (using finite time horizons) does not necessarily lead to a stable closed loop [163]. However, a comprehensive stability theory of model predictive control is available. Various stability criteria exist which differ for each reference definition. This short overview of nominal stability of MPC does not consider model-plant-mismatch, noise or disturbances. In the following, a brief overview of stability criteria for setpoint stabilisation is given based on [153]. Provable stable MPC formulations for setpoint regulation often use appropriate terminal state constraints and end penalties, see, e.g. [163] for an overview. However, also formulations without terminal constraints exist [84, 104, 137]. In all these works, state feedback is used where the stage cost penalises the system states. Whenever L_s penalises system outputs instead of states, additional conditions must be included to cope for the semi-definiteness of L_s with respect to the states. Stability guarantees can, for example, be obtained if additional detectability properties are satisfied. For example, [189] relies on input/output-to-state stability of the open-loop system, while [147] requires weak detectability of the considered system and the usage of a weak detector.

In the present setup, we consider sampled-data MPC, for which we will state suitable conditions to achieve stability in the sense of the convergence of the outputs. The stage cost L_s, terminal cost E_s, and terminal region $\mathcal{F}_\mathrm{s}$ will be chosen according to the following assumptions [63]:

Assumption 5. *The stage cost $L_\mathrm{s} : \mathbb{R}^{n_\mathrm{y}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}_0^+$ is continuous, $L_\mathrm{s}(0,0) = 0$ and it is lower bounded by a class $\mathcal{K}_\infty$ function α_1 such that $L_\mathrm{s}(e_\mathrm{s}, w_s) \geq \alpha_1(\|e_\mathrm{s}\|)$ for all $(e_\mathrm{s}, w_\mathrm{s})$.*

Assumption 6. *The terminal cost* $E_\mathrm{s} : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}_0^+$ *is positive semi-definite,* $E_\mathrm{s}(0) = 0$*, and it is continuously differentiable.*

Assumption 7. *The terminal constraint set* $\mathcal{F}_\mathrm{s} \subseteq \mathcal{X}$ *is closed.*

Assumption 8. *For all* $\tilde{x} \in \mathcal{F}_\mathrm{s}$ *and the considered sampling time* $T_\mathrm{s} > 0$*, there exists an input* $u_\mathcal{F}(\cdot) \in \mathcal{U}$ *such that, for all* $\tau \in [0, T_\mathrm{s})$*,*

$$\frac{\partial E_\mathrm{s}}{\partial x} \cdot f(x(\tau), u_\mathcal{F}(\tau)) + L_\mathrm{s}(e_\mathrm{s}(\tau), w_\mathrm{s}(\tau)) \leq 0$$

and the closed-loop solution stays in the terminal region $x(\tau) = x(\tau, \tilde{x}|u_\mathcal{F}) \in \mathcal{F}_\mathrm{s}$*; i.e. the terminal region* $\mathcal{F}_\mathrm{s}$ *is control invariant.*

Provided that these assumptions hold, one can state the following theorem:

Theorem 1. *If the optimal control problem* (2.5) *is feasible for the initial time instant* t_0 *and its stage cost, terminal cost, and the terminal constraints satisfy Assumptions 5-8, then the optimal control problem* (2.5) *is recursively feasible and the error* $e_\mathrm{s}(t)$ *converges to zero under sampled-data NMPC.*

A proof for convergence with sampled-data MPC, which applies also here, can be found in [63]. Please note that, as already mentioned, the stage cost L_s does merely penalise errors of the outputs of the system, which makes it semi-definite with respect to the states. Therefore, convergence of the output error rather than the states is achieved. However, the terminal cost E_s and the terminal constraint set $\mathcal{F}_\mathrm{s}$ depend on the system states, such that these do not grow unbounded and end in the terminal region in each iteration. Convergence of the state instead of the output (error) is obtained, for example, by additionally assuming that the system is input/output-to-state stable and by replacing the lower bound in Assumption 5 with $L_\mathrm{s}(e_\mathrm{s}, w_\mathrm{s}) \geq \alpha_1(\|e_\mathrm{s}\|) + \alpha_1(\|w_\mathrm{s}\|)$.

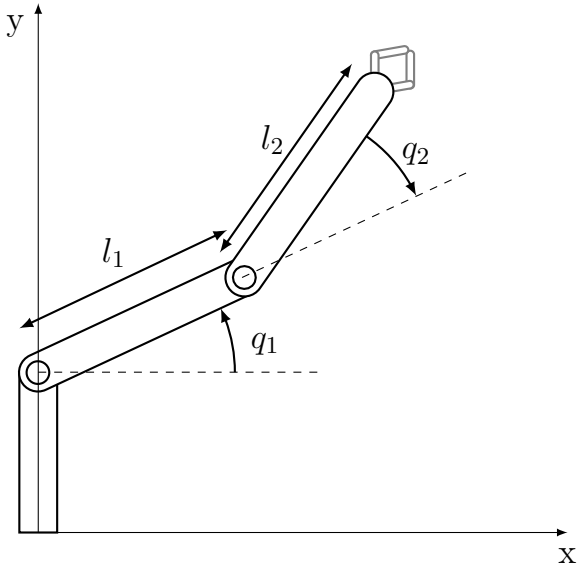
2.2.4 Illustrative Example

We illustrate the setpoint regulation problem using the control of a robotic manipulator as depicted in Figure 2.1a. We limit the problem to two joints of the seven-degrees-of-freedom robot, which results in a planar arm configuration and a two-dimensional workspace, cf. 2.1b. The dynamics of the robot can be described via

$$\dot{q}(t) = \omega(t) \tag{2.6}$$
$$\dot{\omega}(t) = B^{-1}(q(t))u(t), \tag{2.7}$$
$$y(t) = h_\mathrm{fk}(q(t)) \tag{2.8}$$

with the states $x(t) = (q(t), \omega(t))^\top, x(t) \in \mathbb{R}^4$ where $q(t) = (q_1(t), q_2(t))^\top$ describes the angles of the two joints and $\omega(t) = (\omega_1(t), \omega_2(t))^\top$ their angular velocities. The

(a) KUKA Lightweight robot.



(b) Planar two degrees of freedom (DOF) configuration.

Figure 2.1: Robotic manipulator considered as a control example.

input $u(t) \in \mathbb{R}^2$ represents the torques in each joint which are mapped to the angular accelerations via the inverse of the inertia matrix $B : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$. Note that this model ignores friction, Coriolis and centrifugal effect, and assumes that gravity effects are fully compensated internally or that the robot operates planar. Via forward kinematics $h_{\text{fk}} : \mathbb{R}^2 \to \mathbb{R}^2$ of the robot, the Cartesian position y of the end-effector can be obtained. To control the system, a nominal model predictive controller with quadratic cost functions $L_{\text{s}} = e_{\text{s}}^\top Q_{\text{s}} e_{\text{s}} + w_{\text{s}}^\top R_{\text{s}} w_{\text{s}}$ and $E_{\text{s}} = \varepsilon_{\text{s}}^\top Q_{\text{E,s}} \varepsilon_{\text{s}}$ is used. The prediction horizon is $T = 0.5\,\text{s}$ and the terminal constraint set $\mathcal{F}_{\text{s}} = x_{\text{s}}$ is used. The constraint sets for input, state, and output space are $\mathcal{U} = [-5\,\text{Nm}, 5\,\text{Nm}] \times [-5\,\text{Nm}, 5\,\text{Nm}]$, $\mathcal{X} = [0, \pi] \times [0, \pi] \times [-2\,\frac{\text{rad}}{\text{s}}, 2\,\frac{\text{rad}}{\text{s}}] \times [-2\,\frac{\text{rad}}{\text{s}}, 2\,\frac{\text{rad}}{\text{s}}]$ and $\mathcal{Y} = \left\{(y_1, y_2) | y_1^2 + y_2^2 \leq (l_1 + l_2)^2\right\}$, where l_1 and l_2 are the lengths of the links, see also Figure 2.1b. The involved parameter values for the dynamic model and the controller are given in Appendix A.1. In the Cartesian workspace a setpoint change for the end-effector from $(0.4\,\text{m}, 0.4\,\text{m})$ to $(0.4\,\text{m}, 0.5\,\text{m})$ at $t = 2.5\,\text{s}$ is desired, cf. Figure 2.2 (black line). The resulting Cartesian end-effector position of the robot in the vertical direction is depicted in Figure 2.2 as a blue dashed line. Due to the predictive capability of the controller, the system begins to move towards the new setpoint before 2.5 s. However, significant deviations from the actual reference, see also Figure 2.3, and a visible overshooting occur. Different tuning of the controller can reduce such an overshoot. For example, reducing the weights on the control error makes the controller less aggressive. However, this might deteriorate the position accuracy in the non-transient phase. Alternatively, tracking formulations with smooth time-dependent reference trajectories can be exploited as outlined in the following subsection.

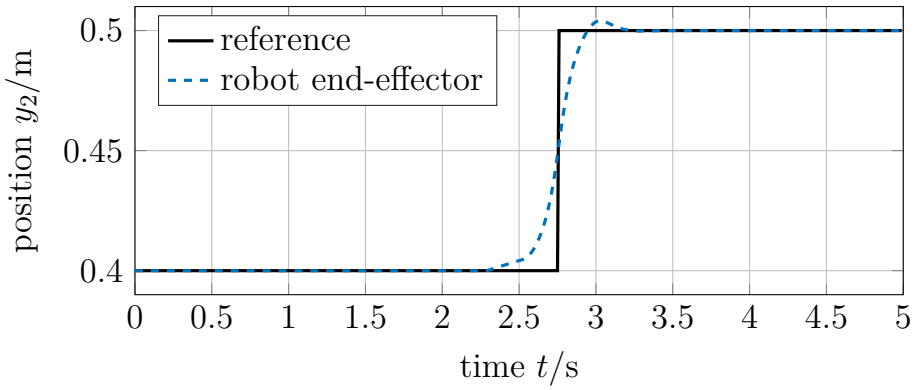


Figure 2.2: Cartesian vertical position y = y_2 of the reference and the robot end-effector for a 10 cm setpoint change from (0.4 m, 0.4 m) to (0.4 m, 0.5 m).

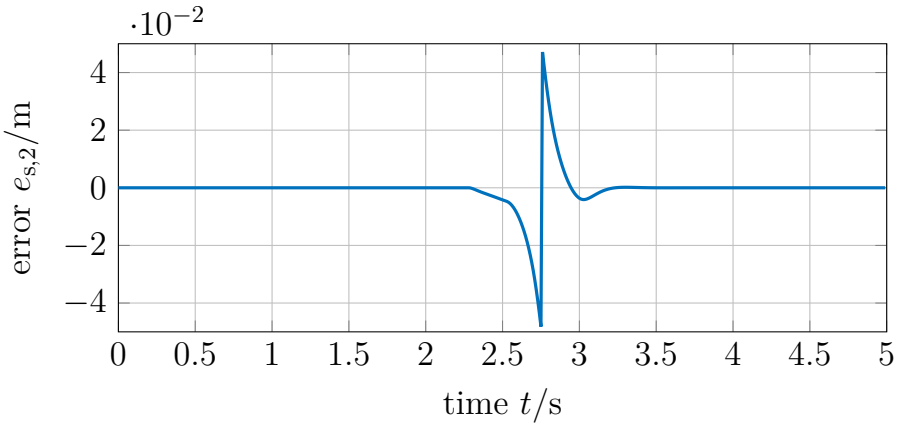


Figure 2.3: Control error in Cartesian y-direction for MPC with a 10 cm setpoint change.

2.3 Trajectory Tracking

In trajectory tracking tasks, the reference is time-dependent. Application examples where the tracking of a given trajectory is important are for instance the control of unmanned aerial vehicles [2, 108], mobile robots [124], drinking water networks [182], or the artificial pancreas [98]. The control error in trajectory tracking problems is given by

$$e_{\mathrm{tt}}(t) := r_{\mathrm{tt}}(t) - y(t), \tag{2.9}$$

where $r_{\mathrm{tt}} : \mathbb{R}_0^+ \to \mathbb{R}^{n_y}$ defines the time-dependent reference trajectory.

2.3.1 Time-dependent References

There exist several forms of reference trajectories depending on the application. They can be given explicitly as an analytic function known to the controller, or only in terms of the reference value at the current time. If the reference is known completely a priori, this information can be exploited by a model predictive controller [57]. Alternatively, the reference can be obtained via the solution of an exogenous system [103]. For

both cases, stabilising MPC formulations exist if the reference is designed accordingly [53, 56, 57, 147]. For instance, time-varying terminal conditions for given trajectories are proposed in [56, 57]. In [53] and [147] exosystems for reference generation are used. Stability is achieved via time-varying terminal constraints and by including an additional model of the exosystem, respectively.

However, often one faces the problem that the references are not designed in a suitable and useful way. For instance, rapidly changing references might not be reachable or trackable. Moreover, the following of an arbitrary reference could lead to constraint violation. Reference governors can be used in such cases since they act as pre-filters to guarantee specific properties of the reference. They modify the reference to avoid state or input constraint violation of the system or to improve the performance, see, e.g. [72] for an overview. The idea of reference adaptation inside a model predictive controller was proposed in [139–141]. This concept is applicable, even if the reference evolution is a priori unknown. An additional degree of freedom is added in the predictive controller via the adaptation of an artificial reference. The deviation of the artificial reference from the original one is minimised. At the same time, the system is controlled to follow the artificial reference. By modification of the control objective via the artificial reference, recursive feasibility of the tracking MPC scheme is achieved despite untrackable changes in the original reference.

In the following, we consider references which are planned offline such that they are entirely or at least partially known to the controller. We show how to design the MPC accordingly to guarantee stability. This framework is extended in Chapter 5 to include machine learning as reference generators in case of limited knowledge of the reference. We use the following definition, to ensure reliable behaviour and good performance of the tracking controller:

Definition 1. *(Trackability under Constraints).*
A reference $r_{\mathrm{tt}} : \mathbb{R}_0^+ \to \mathbb{R}^{n_{\mathrm{y}}}$ *is said to be trackable for the system* (2.1) *if it fulfils the output constraints* $r_{\mathrm{tt}}(t) \in \mathcal{Y}$ *and can be followed given the system dynamics once starting on it, i.e.* $\exists u_{\mathrm{tt}}(t) \in \mathcal{U}$ *such that* $r_{\mathrm{tt}}(t) = h(x_{\mathrm{tt}}(t))$ *with* $\dot{x}_{\mathrm{tt}}(t) = f(x_{\mathrm{tt}}(t), u_{\mathrm{tt}}(t))$, $r_{\mathrm{tt}}(0) = h(x_{\mathrm{tt}}(0))$, *and* $x_{\mathrm{tt}}(t) \in \mathcal{X}$ *for all* $\forall t \in \mathbb{R}_0^+$.

Though restrictive, trackability of the reference according to Definition 1 allows to determine terminal cost and constraints to prove recursive feasibility of tracking MPC. This constrained trackability guarantees that the system can stay on the reference when starting on it under output, state, and input constraints.

2.3.2 Model Predictive Control for Trajectory Tracking

Instead of a constant value, in tracking MPC, a time-varying reference should be followed, see for instance [57, 62, 65, 110, 139–141, 146, 161]. The formulation in error coordinates allows reformulating this problem into the stabilisation of the origin. However, the tracking error e_{tt} possesses time-varying dynamics, such that the predictive

control scheme must account for a time-varying system. The cost function for tracking MPC can be defined as

$$J_{\mathrm{tt}}\left(\bar{e}_{\mathrm{tt}}, \bar{w}_{\mathrm{tt}}, \bar{\varepsilon}_{\mathrm{tt}}\right) := \int_0^T L_{\mathrm{tt}}\left(\bar{e}_{\mathrm{tt}}(\tau), \bar{w}_{\mathrm{tt}}(\tau)\right) \mathrm{d}\tau \; + E_{\mathrm{tt}}\left(\bar{\varepsilon}_{\mathrm{tt}}(T)\right). \tag{2.10}$$

In analogy to the setpoint stabilisation case, $\bar{w}_{\mathrm{tt}}(t) := u_{\mathrm{tt}}(t) - \bar{u}(t)$ with $\bar{w}_{\mathrm{tt}}(t) \in \mathbb{R}^{n_{\mathrm{u}}}$, and $\bar{\varepsilon}_{\mathrm{tt}}(t) := x_{\mathrm{tt}}(t) - \bar{x}(t)$ with $\bar{\varepsilon}_{\mathrm{tt}}(t) \in \mathbb{R}^{n_{\mathrm{x}}}$. Furthermore, $L_{\mathrm{tt}} : \mathbb{R}^{n_{\mathrm{y}}} \times \mathbb{R}^{n_{\mathrm{u}}} \to \mathbb{R}_0^+$ denotes the stage cost, and $E_{\mathrm{tt}} : \mathbb{R}^{n_{\mathrm{x}}} \to \mathbb{R}_0^+$ is the terminal penalty. In this formulation, the cost function is not time-varying. Rather, the explicit time dependency enters the optimal control problem via the constraints. The optimal control problem for trajectory tracking can be formulated as

$$\min_{\bar{u} \in \mathcal{C}} \; J_{\mathrm{tt}}\left(\bar{e}_{\mathrm{tt}}, \bar{w}_{\mathrm{tt}}, \bar{\varepsilon}_{\mathrm{tt}}\right) \tag{2.11a}$$

subject to $\forall \tau \in [0, T]$

$$\dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(0) = x(t_k), \tag{2.11b}$$
$$\bar{e}_{\mathrm{tt}}(\tau) = r_{\mathrm{tt}}(t_k + \tau) - h(\bar{x}(\tau)), \tag{2.11c}$$
$$\bar{w}_{\mathrm{tt}}(\tau) = u_{\mathrm{tt}}(t_k + \tau) - \bar{u}(\tau), \tag{2.11d}$$
$$\bar{\varepsilon}_{\mathrm{tt}}(\tau) = x_{\mathrm{tt}}(t_k + \tau) - \bar{x}(\tau), \tag{2.11e}$$
$$\bar{x}(\tau) \in \mathcal{X}, \; \bar{u}(\tau) \in \mathcal{U}, \; h(\bar{x}(\tau)) \in \mathcal{Y}, \tag{2.11f}$$
$$\bar{x}(T) \in \mathcal{F}_{\mathrm{tt}}(t_k + T), \tag{2.11g}$$

where $\mathcal{F}_{\mathrm{tt}}(t) \subseteq \mathcal{X} \cap h^{-1}(\mathcal{Y}), \forall t \in \mathbb{R}_0^+$. The difference to the MPC formulation for setpoint stabilisation (2.5) is the time dependency of the reference r_{tt} (and consequently u_{tt} and x_{tt}) as well as the time dependency of the terminal constraint set $\mathcal{F}_{\mathrm{tt}}$ in (2.11g). One possible choice of this time-dependent terminal constraint is an equality constraint forcing the state to be on the state reference $x_{\mathrm{tt}}(t_k + T)$. How to obtain less restrictive terminal inequality constraints via time-varying level sets of Lyapunov functions is shown, for example, in [57].

2.3.3 Stability of MPC for Trajectory Tracking

When considering the tracking of time-dependent references, stability conditions from setpoint stabilisation cannot directly be applied. Instead, the time dependency of the reference leads to time-varying error dynamics such that similarities to MPC designs for time-varying systems appear. Setpoint changes can lead to the infeasibility of the predictive controller if reachability is not considered, cf. [139]. Time-varying terminal regions provide a possibility to achieve trackability. Here, knowledge about the reference is directly exploited to calculate those terminal sets, see, e.g. [57, 65,

110, 161]. Furthermore, trackability according to Definition 1 is used, such that the existence of feasible inputs to follow the reference is guaranteed. If the reference is designed accordingly, the following assumptions [56, 57] ensure convergence for output tracking using sampled-data NMPC:

Assumption 9. *The stage cost* $L_{\mathrm{tt}} : \mathbb{R}^{n_{\mathrm{y}}} \times \mathbb{R}^{n_{\mathrm{u}}} \to \mathbb{R}_0^+$ *is continuous,* $L_{\mathrm{tt}}(0,0) = 0$ *and is lower bounded by a class* $\mathcal{K}_\infty$ *function* α_1 *such that* $L_{\mathrm{tt}}(e_{\mathrm{tt}}, w_{\mathrm{tt}}) \geq \alpha_1(\|e_{\mathrm{tt}}\|)$ *for all* $(e_{\mathrm{tt}}, w_{\mathrm{tt}})$.

Assumption 10. *The terminal cost* $E_{\mathrm{tt}} : \mathbb{R}^{n_{\mathrm{x}}} \to \mathbb{R}_0^+$ *is positive semi-definite and continuously differentiable in* x.

Assumption 11. *The terminal constraint set* $\mathcal{F}_{\mathrm{tt}} \subseteq \mathcal{X}$ *is closed and time-varying.*

Assumption 12. *For all* $\tilde{x} \in \mathcal{F}_{\mathrm{tt}}$ *and the considered sampling time* $T_{\mathrm{s}} > 0$*, there exists an admissible input* $u_{\mathcal{F}}(\cdot) \in \mathcal{U}$ *such that, for all* $\tau \in [0, T_{\mathrm{s}})$*,*

$$\frac{\partial E_{\mathrm{tt}}}{\partial x} \cdot f(x(\tau), u_{\mathcal{F}}(\tau)) + L_{\mathrm{tt}}(e_{\mathrm{tt}}(\tau), w_{\mathrm{tt}}(\tau)) \leq 0$$

and the closed-loop solution fulfils $x(\tau) = x(\tau, \tilde{x}|u_{\mathcal{F}}) \in \mathcal{F}_{\mathrm{tt}}(\tau)$*; i.e. the terminal region is control invariant.*

In contrast to setpoint stabilisation, the terminal constraint set $\mathcal{F}_{\mathrm{tt}}$ depends on time (see Assumption 11) due to the inherently time-varying tracking error. For a detailed discussion on how to construct corresponding terminal regions, see [56, 57]. Please note, that these works depend on the knowledge of the reference evolution and corresponding inputs to track it. Similarly to the previous subsection, stability of the closed loop in the sense of convergence can be ensured:

Theorem 2 ([56, 57])**.** *If the optimal control problem* (2.11) *is feasible for the initial time instant* t_0 *and the stage cost, the terminal cost and the terminal constraints satisfy Assumptions 9-12, then* (2.11) *is recursively feasible and the tracking error* e_{tt} *converges to zero under sampled-data NMPC.*

2.3.4 Illustrative Example

We use the same robotic system and overall setup as the example in Section 2.2.4. The model predictive controller uses the same model and control parameters (prediction horizon, weightings, constraints) while the only difference lies in the reference definition. The planned reference is shown in black in Figure 2.4 (top). The transient time of the reference trajectory is comparable to the time needed in the previous example for the setpoint change. The desired position change does not take the system dynamics and constraints into account. In such a case, trackability cannot be guaranteed. The controlled robot cannot follow the reference, as this would require input torques

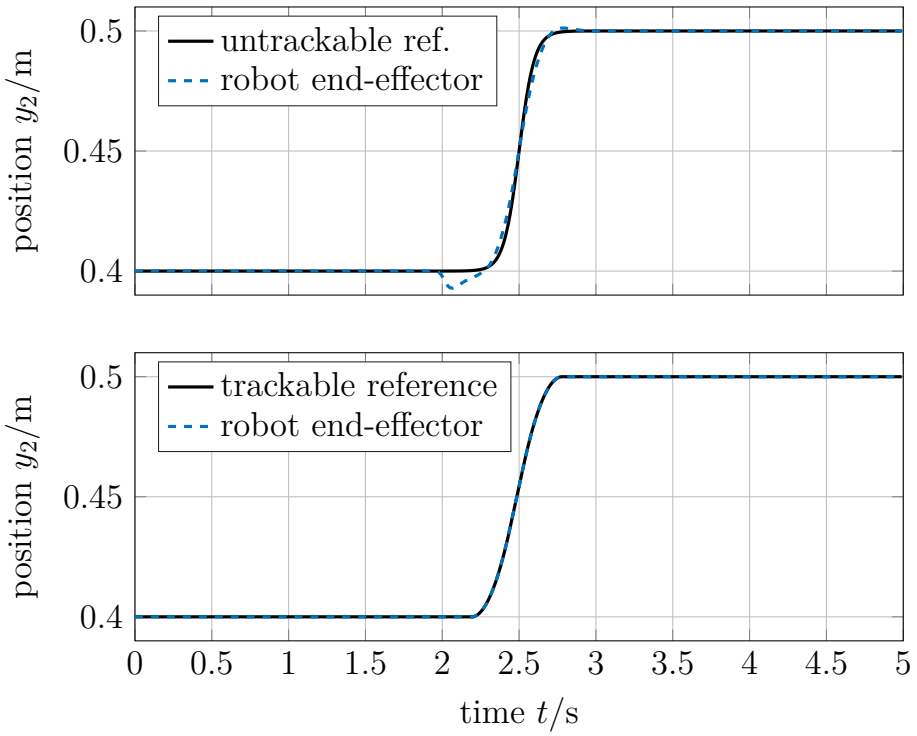


Figure 2.4: Cartesian y-position of reference and robot end-effector.

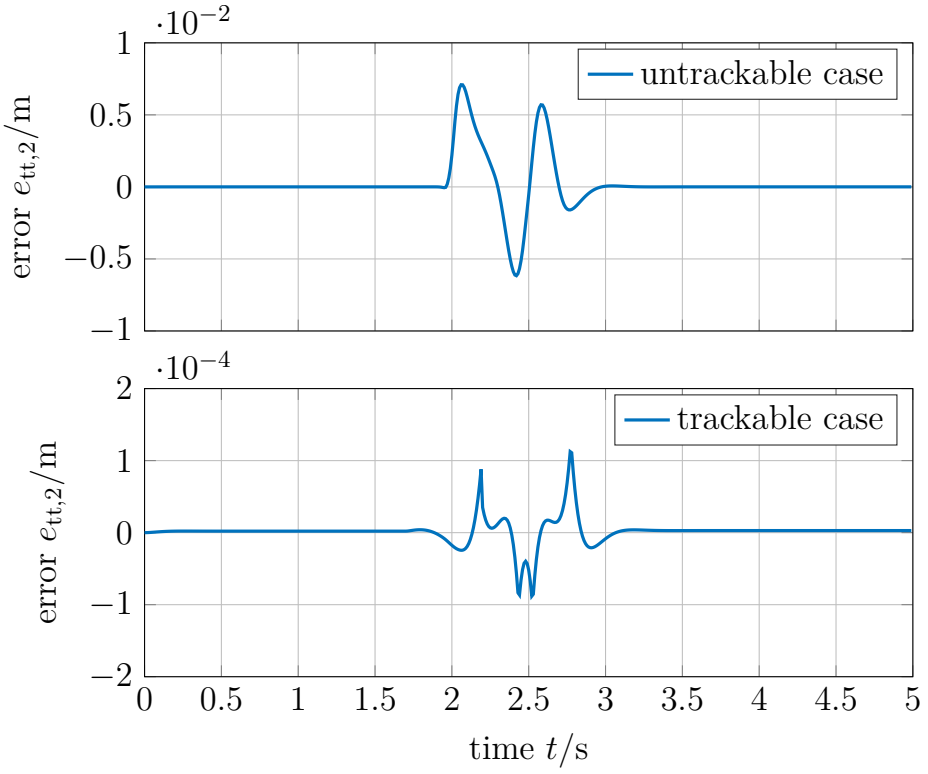


Figure 2.5: Control error in Cartesian y-direction for the same predictive controller with two different reference trajectories.

up to 13 Nm, which violates the upper bound of the input constraints of 5 Nm. Since the MPC control input is limited by this upper bound, significant control errors occur, which can also be seen in Figure 2.5, top. Trackability can be achieved if the reference trajectory design accounts for the system dynamics and constraints. The tracking of such a reference is depicted in Figure 2.4, bottom. Even though the transition time is comparable to the previous reference (Figure 2.4, top) the tracking performance with the same predictive controller is significantly increased, cf. Figure 2.5, bottom. Please note, that both reference trajectories lead to the same Cartesian positions of the end-effector (a straight vertical line) while they only differ in their timing laws. Consequently, the timing law of a reference strongly influences the achievable tracking precision. Sophisticated planning of the trajectories can help to meet the precision requirements. Alternatively, the timing law of references can be adjusted online by the controller, which leads to path-following MPC formulations as outlined in the following subsection.

2.4 Path Following

While references formulated via trajectories have a specific and fixed timing law, this might not be central in the desired control task. Rather often it is introduced by a path planner which transfers the control task into the reference trajectory. For example, in UAV inspection tasks, a specific geometric route should be visited by the vehicle. The exact timing when to be where on that route might be secondary as long as the UAV finishes its inspection mission in a specific time or as fast as possible. Another example is the cutting of workpieces by a robot which should work precisely, but the velocity along the cutting lines is less important. Often those and similar tasks are formulated as trajectory tracking problems, where a path planner not only fixes the geometry of the reference but also its velocity profile. The offline fixation of the reference velocity, however, might not be optimal since disturbances, such as wind or changing material properties, can not be included. In path following formulations, the velocity of the reference is not fixed a priori [56, 153]. Instead, the planner provides a geometric curve, and the velocity along it is adjusted online by the controller. Applications of MPC for path-following problems span, for example, from robotic manipulators [61, 154], an X-Y-table [126], and a tower crane [26] up to unmanned areal vehicles [101, 102].

2.4.1 Reference Path Definition

In path following, the reference path can be defined as a parametrised regular curve in the output space

$$\mathcal{P} := \{y_\mathrm{r} \in \mathbb{R}^{n_\mathrm{y}} | y_\mathrm{r} = r_\mathrm{pf}(\theta(t))\} \tag{2.12}$$

with the parametrisation $r_\mathrm{pf} : \Theta \to \mathbb{R}^{n_\mathrm{y}}$. The reference in path following is only indirectly depending on time via the path parameter $\theta(t) \in \Theta$, see also Figure 2.6.

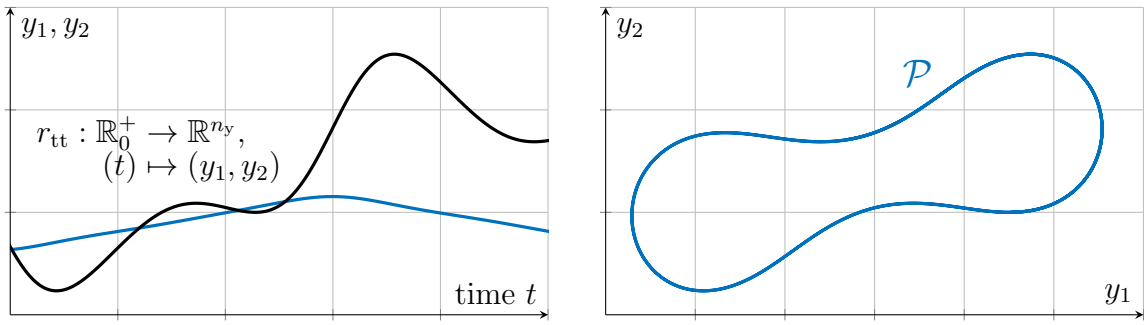


Figure 2.6: Illustration of trajectory tracking (left) and path following (right).

In turn, the evolution of the path parameter over time is not fixed a priori. We will consider constraints on the path parameter of the form $\Theta := [\theta_{\mathrm{start}}, \theta_{\mathrm{end}}]$ and $\dot{\theta} \geq 0$. In the controller, the path parameter is steered forward to its end value θ_{end} such that the whole parametrised reference is transversed.

Remark 1. *The parametrisation of the reference can be described by a virtual dynamical single-input single-output system*

$$\begin{aligned} \dot{z}(t) &= g(z(t), v(t)), \\ \theta(t) &= l(z(t)), \end{aligned} \tag{2.13}$$

where $z(t) \in \mathbb{R}^{n_z}$ is the virtual state, $v(t) \in \mathbb{R}$ is the virtual input, and the output is the path parameter $\theta(t) \in \mathbb{R}$. The virtual system dynamics are $g : \mathbb{R}^{n_z} \times \mathbb{R} \to \mathbb{R}^{n_z}$, while $l : \mathbb{R}^{n_z} \to \mathbb{R}$ denotes the virtual output equation. The naming virtual *indicates the fact that this parametrisation is chosen freely and is not representing a real physical system. Rather, it defines the time dependency of the path parameter, which is, as already mentioned, not fixed a priori but depends on the virtual input $v(t)$. Constraints on the virtual states, inputs and the path parameter can be introduced via $z(t) \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}, v(t) \in \mathcal{V} \subseteq \mathbb{R}, \theta(t) \in \Theta \subseteq \mathbb{R}$. For simplicity, the virtual dynamics (2.13) can be chosen as an integrator chain. In this case, forward motion can be enforced by choosing the virtual state constraints to restrict the derivative of the path parameter to be positive until reaching θ_{end}.*

The path following error is in general given by

$$e_{\mathrm{pf}}(t) := r_{\mathrm{pf}}(\theta(t)) - y(t). \tag{2.14}$$

If there exist inputs v and u which can guarantee that $e_{\mathrm{pf}}(t) = 0$ if $e_{\mathrm{pf}}(0) = 0$ for $t \in \mathbb{R}_0^+$, then the path is exactly followable by the system.

Definition 2. *(Path followability under Constraints).*
Followability of a path $\mathcal{P}$ for system (2.1) is given if the path fulfils the output constraints $\mathcal{P} \in \mathcal{Y}$ and is followable given the system dynamics, i.e. $\exists u_{\mathrm{pf}}(t) \in \mathcal{U}$ such

that for all $t \in \mathbb{R}_0^+$ *holds that* $r_{\mathrm{pf}}(\theta(t)) = h(x_{\mathrm{pf}}(t))$*, with* $\dot{x}_{\mathrm{pf}}(t) = f(x_{\mathrm{pf}}(t), u_{\mathrm{pf}}(t))$ *and* $x_{\mathrm{pf}}(t) \in \mathcal{X}$*, while the path parameter* $\theta(t)$ *evolves continuously with time, and* $\dot{\theta} > 0$ *holds almost everywhere in* $[\theta_{\mathrm{start}}, \theta_{\mathrm{end}}]$*.*

Convergence of the path following error e_{pf} to zero implies (under mild technical assumptions) that the state x converges to a manifold in the state space [56, 153, 173, 174]. The corresponding reference input u_{pf} can be obtained for special system classes analytically or by optimisation, see, e.g. [56, 59]. Unlike the trajectory tracking case, these reference state and input are not given via functions of time directly. Instead, they depend on the path parameter θ, which can be adjusted online by the controller as outlined below.

2.4.2 Model Predictive Control for Path Following

The cost function used for path following MPC is defined as

$$J_{\mathrm{pf}}\left(\bar{e}_{\mathrm{pf}}, \bar{\theta}, \bar{u}, \bar{v}, \bar{x}, \bar{z}\right) := \int_0^T L_{\mathrm{pf}}\left(\bar{e}_{\mathrm{pf}}(\tau), \bar{\theta}(\tau), \bar{u}(\tau), \bar{v}(\tau)\right) \mathrm{d}\tau + E_{\mathrm{pf}}\left(\bar{x}(T), \bar{z}(T)\right). \quad (2.15)$$

Here, $L_{\mathrm{pf}} : \mathbb{R}^{n_{\mathrm{y}}} \times \mathbb{R} \times \mathbb{R}^{n_{\mathrm{u}}} \times \mathbb{R} \to \mathbb{R}_0^+$ denotes the stage cost or cost function and $E_{\mathrm{pf}} : \mathbb{R}^{n_{\mathrm{x}}} \times \mathbb{R}^{n_{\mathrm{z}}} \to \mathbb{R}_0^+$ is the terminal penalty for path following MPC. As we consider a form with virtual path dynamics the cost additionally depends on the predictions of the virtual state z, input v and output θ. The end cost penalises the predictions of original system states x as well as the virtual states z.

The optimal control problem for path following can be formulated as

$$\min_{\bar{u}\in\mathcal{C}, \bar{v}\in\mathcal{C}} \; J_{\mathrm{pf}}\left(\bar{e}_{\mathrm{pf}}, \bar{\theta}, \bar{u}, \bar{v}, \bar{x}, \bar{z}\right) \quad (2.16a)$$

subject to $\forall \tau \in [0, T]$

$$\dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(0) = x(t_k), \quad (2.16b)$$
$$\dot{\bar{z}}(\tau) = g(\bar{z}(\tau), \bar{v}(\tau)), \quad \bar{z}(0) = z(t_k), \quad (2.16c)$$
$$\bar{e}_{\mathrm{pf}}(\tau) = r_{\mathrm{pf}}(l(\bar{z}(\tau)) - h(\bar{x}(\tau)), \quad (2.16d)$$
$$\bar{\theta}(\tau) = l(\bar{z}(\tau)), \quad (2.16e)$$
$$\bar{x}(\tau) \in \mathcal{X}, \; \bar{u}(\tau) \in \mathcal{U}, \; h(\bar{x}(\tau)) \in \mathcal{Y}, \quad (2.16f)$$
$$\bar{z}(\tau) \in \mathcal{Z}, \; \bar{v}(\tau) \in \mathcal{V}, \; \bar{\theta}(\tau) \in \Theta, \quad (2.16g)$$
$$(\bar{x}(T), \bar{z}(T))^\top \in \mathcal{F}_{\mathrm{pf}}, \quad (2.16h)$$

where $\mathcal{F}_{\mathrm{pf}} \subseteq (\mathcal{X} \times \mathcal{Z}) \cap (h^{-1}(\mathcal{Y}) \times l^{-1}(\Theta))$. Additionally to the original system dynamics (2.16b) the virtual system dynamics (2.16c) are added. The control error (2.16d) thus is a function of the outputs of an extended system with states $x_{\mathrm{ext}} := (x, z)^\top$. Via (2.16f) and (2.16g) state, input, and output constraints of this extended system

representation are considered. In the same way, the terminal constraint (2.16h) consider both x and z. Important to note is that the terminal constraint set $\mathcal{F}_{\mathrm{pf}}$ does not depend on time. This is caused by the path following error dynamics, which are not time-variant as in the tracking case. Instead, they depend on the extended system states, which can be influenced via the two inputs u and v. Via the virtual input v, the controller has an additional degree of freedom compared to the tracking formulation. In general, this allows for improved performance.

2.4.3 Stability of MPC for Path Following

The error dynamics in path following formulations are not time-varying as in the tracking case. Instead, path following MPC can be viewed as the stabilisation of the origin for an extended error system which includes both the original and the virtual system. The following assumptions are required for concluding convergence:

Assumption 13. *The stage cost* $L_{\mathrm{pf}} : \mathbb{R}^{n_{\mathrm{y}}} \times \mathbb{R} \times \mathbb{R}^{n_{\mathrm{u}}} \times \mathbb{R}^{n_{\mathrm{v}}} \to \mathbb{R}_0^+$ *is continuous and is lower bounded by a class* $\mathcal{K}_\infty$ *function* α_1 *such that* $L_{\mathrm{pf}}(e_{\mathrm{pf}}, \theta, u, v) \geq \alpha_1(\|(e_{\mathrm{pf}}, \theta - \theta_{\mathrm{end}})^\top\|)$ *for all* $(e_{\mathrm{pf}}, \theta, u, v)$.

Assumption 14. *The terminal cost* $E_{\mathrm{pf}} : \mathbb{R}^{n_{\mathrm{x}}} \times \mathbb{R}^{n_{\mathrm{z}}} \to \mathbb{R}_0^+$ *is positive semi-definite and continuously differentiable in* x *and* z.

Assumption 15. *The terminal constraint set* $\mathcal{F}_{\mathrm{pf}} \subseteq \mathcal{X} \times \mathcal{Z}$ *is closed.*

Assumption 16. *For all* $(\tilde{x}, \tilde{z})^\top \in \mathcal{F}_{\mathrm{pf}}$ *and the considered sampling time* $T_{\mathrm{s}} > 0$*, there exist inputs* $(u_{\mathcal{F}}, v_{\mathcal{F}})^\top(\cdot) \in \mathcal{U} \times \mathcal{V}$ *such that for all* $\tau \in [0, T_{\mathrm{s}})$

$$\left(\frac{\partial E_{\mathrm{pf}}}{\partial x}, \quad \frac{\partial E_{\mathrm{pf}}}{\partial z}\right) \cdot \begin{pmatrix} f(x(\tau), u_{\mathcal{F}}(\tau)) \\ g(z(\tau), v_{\mathcal{F}}(\tau)) \end{pmatrix} + L_{\mathrm{pf}}(e_{\mathrm{pf}}(\tau), \theta(\tau), u_{\mathcal{F}}(\tau), v_{\mathcal{F}}(\tau)) \leq 0$$

and the closed-loop solution $x(\tau) = x(\tau, \tilde{x}|u_{\mathcal{F}})$ *and* $z(\tau) = z(\tau, \tilde{z}|v_{\mathcal{F}})$ *stay in* $\mathcal{F}_{\mathrm{pf}}$*; i.e. the terminal region is control invariant.*

With these assumptions the following convergence and stability results for predictive path following is obtained [56, 58]:

Theorem 3. *If the optimal control problem* (2.16) *is feasible for the initial time instant* t_0 *and the stage cost, the terminal cost, and the terminal constraints are chosen to fulfil Assumptions 13-16, then* (2.16) *is recursively feasible and the path-following error* e_{pf} *converges to zero under sampled-data NMPC.*

Proving Theorem 3 relies on the possibility of reformulating the path-following problem into the setpoint stabilisation of an extended system. For an extensive discussion and a complete convergence proof as well as insights into the computation of suitable terminal control laws and terminal regions to fulfil the stated assumptions, the reader

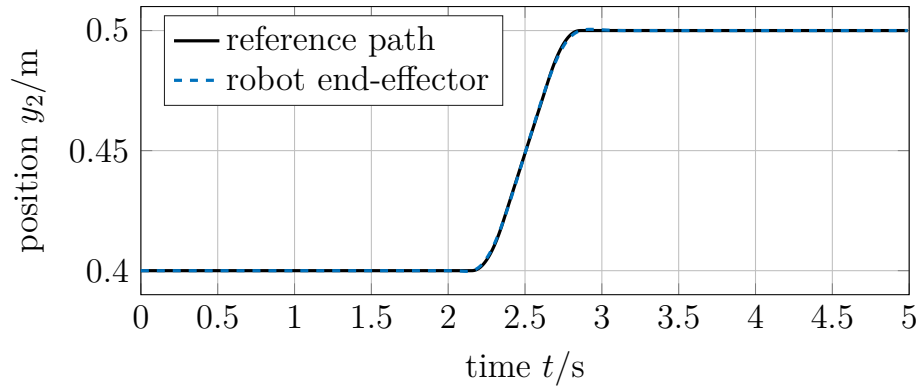


Figure 2.7: Cartesian y-position of the reference and the robot end-effector for an MPC path following formulation.

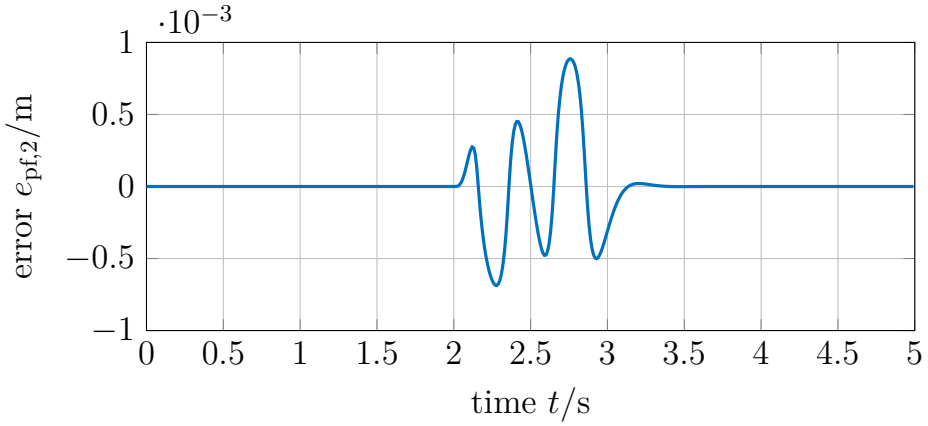


Figure 2.8: Control error in Cartesian y-direction for MPC with path following formulation.

is referred to [56, 58]. Stability proofs for path following for *discrete-time* systems and *state feedback* can be obtained in a straight forward manner from setpoint stabilisation problems, following the classical ideas, for example, presented in [163].

2.4.4 Illustrative Example

Whenever a control task does not require a specific fixed velocity profile (for instance in contrast to synchronisation tasks) path following builds an alternative to tracking formulations. In this example, the same setup as in the previous examples on setpoint regulation (Section 2.2.4) and trajectory tracking (Section 2.3.4) is used. Instead of fixing the reference timing law as in Section 2.3.4 the reference path is given in terms of a geometric curve $\mathcal{P} = \{y \in \mathbb{R}^{n_y} | y_1 = 0.4, y_2 \in [0.4, 0.5]\}$ with the parametrisation $r_{\mathrm{pf}} = (0.4, 0.4{+}0.1(\theta{+}1))^\top$. The path parameter is limited by $\theta_{\mathrm{start}} = -1$ and $\theta_{\mathrm{end}} = 0$. Its evolution over time is driven by the virtual system $\dot{z}_1(t) = z_2(t), \dot{z}_2(t) = v(t)$, where $z_1(t) = \theta(t)$. Additional constraints for the virtual system are included in the predictive controller, which are $-10 \leq v(t) \leq 10$ and $0 \leq z_2(t) \leq 2$. The additional entries in the weighting matrices for the virtual system are chosen to achieve a similar

transition time as in the previous examples and are given in Appendix A.1. The Cartesian reference and the end-effector position for the vertical motion are depicted in Figure 2.7. The corresponding control error is depicted in Figure 2.8. Even though a larger control error occurs compared to the trackable trajectory reference case, the control error is significantly smaller than in the untrackable case, cf. Figure 2.5. The benefit of the path following over the tracking case with a trackable reference is that the reference planner does not need to define a velocity profile a priori. As we have seen in the untrackable trajectory case, an inadequate velocity profile can lead to a significant decrease in controller performance, which is prevented by the path following setup.

2.5 Summary

In this chapter, we have introduced and compared different formulations of model predictive controllers tailored towards the solution of regulation, tracking, and path following tasks. Besides the summary of state-of-the-art approaches, we have highlighted differences, similarities, and connections between the controller formulations both from the theoretical and practical side. Various simulation studies approached the latter for a robotic manipulator. These studies have been performed with a realistic manipulator model, which was reduced to obtain easily comprehensible yet realistic insights into the predictive controller design and evaluation. The main intention of this chapter was to give an overview of existing approaches, relate them to each other, and lay the foundation for the advances in the remainder of this work. This chapter builds the basis for the design of interaction force controllers in the following chapter. The presented control formulations will be used and extended in Chapter 5 to incorporate learning in MPC. In particular, tracking formulations will be used in Chapter 5, since it allows for timed and synchronous motions. In conclusion, we would like to emphasise that each of the presented control formulations has specific advantages. Accordingly, the choice of the control concept should always be made concerning the task at hand to exploit the corresponding benefits.

3 Model Predictive Force Control

In this chapter, we propose model predictive controllers for contact control in robotics. The direct consideration of the force in the cost function and the constraints allows for safe interaction of the robot with its environment. MPC force control allows flexibility in the formulation and obtaining guarantees such as limited contact forces, prevention from contact loss, and overall closed-loop stability. To do so, we extend the path following formulation from Chapter 2. We underline the applicability of the proposed controllers in simulations and experiments.

3.1 Introduction to Force Control

In many applications, robots interact with their environment. For example, robots are used in handling and pick-and-place tasks where they should grab different objects and move them to a desired position [218]. In milling, grinding, or polishing, robots have to work on object surfaces to process them [41, 217, 220]. Interactions between several robots occur, for example, in cooperative handling tasks [85, 175, 190]. Moreover, robots are also increasingly interacting with humans in manufacturing, teleoperation, health care, and home assistance [40, 49, 95, 125, 166, 168, 176, 234, 241]. For successful task completion, this interaction must be quantified and needs to be considered in the used controllers. A natural measure for the interaction are interaction forces that occur between the robot and its surroundings. Robotic force control is a well known yet still very active research area, cf. [115, 211, 228] and the aforementioned publications. Mainly, force control strategies can be categorised into direct and indirect approaches [211, 228]. Direct force control is used when the applied forces should follow desired values, and a "direct" force-feedback loop is closed to ensure this. In indirect force control, the robot's position or velocity is controlled to achieve a dynamically compliant behaviour to external forces. The controllers ensure that the closed-loop system dynamics represent a desired mechanical compliance/stiffness or impedance/admittance. Indirect force controllers add an additional layer of compliance, to among others, prevent oscillations, which are a common issue in direct force control via PID controllers. In principle, model predictive controllers are not as prone to oscillations or overshoots as PID controllers due to their predictive capabilities if suitable models are available. As we will show, MPC based direct force control schemes are interesting tools whenever the contact forces should adopt desired values instead of behaving like artificial springs or dampers. Using force control is also helpful in partially unknown environments where pure position control can lead to undesired collisions with potentially

high impact forces or contact loss during manipulation. Note that if contacts only occur in a subspace of the workspace of a robot, hybrid force-position control can be used, where some directions are position-controlled, and some are force-controlled [211, 228].

Limiting the occurring contact forces can provide additional safety. Damage of the robot, a workpiece, or harm to a human could occur if the contact forces become too large, making it safety-critical to ensure force limitation. On the other side, contact forces, e.g. during grasping tasks, should not drop below a certain threshold to ensure a safe and reliable grasp or contact.

Different approaches that allow limiting contact forces exist. They can be classified based on the primary control goal, i.e. by whether the controllers consider position control with force limitation or a combination of position and force control with constraints. The latter type addresses indirect and direct force controllers.

In [241], a fully counter-balanced robot is used for ultrasonic probe placement. The mechanical construction allows compensating the gravity forces, such that only small actuation is needed to perform motions. This construction allows to limit contact forces by the design of the actuators but is therefore limited to specific robotic systems. A more general way to ensure constraints on contact forces is the use of model predictive controllers. Position control tasks can be performed with MPC while taking force constraints into account. For example, [111, 112] consider a robot arm that reaches out to a desired goal posture, where the cost function to be optimised is based on the positioning errors. Additionally, the contact forces, which occur due to a cluttered environment with possibly multiple contacts, should be limited. A soft constraint is included in the cost function to limit the forces. A mobile robot, which should reach a particular goal position, is controlled with MPC in [170]. Constraints in the optimal control problem formulation are used to limit the occurring contact forces due to obstacle collisions. Unmanned aerial vehicles are position controlled via MPC in [116] and [36], where contact forces are limited via constraints from above and below, respectively. Contact force constraints in MPC position control are considered in [21, 38] for ground reaction forces in legged robot walking. In contrast to [111, 112, 170], the contact forces serve as an input to the robot. These can be mapped to joint torques, which often serve as inputs in robot control. Constraints on these torque inputs have been considered in pure position control MPC schemes as well [60, 61]. A position control MPC scheme with contact forces as inputs has also been considered for robotic food cutting [130, 169], where the end-effector contact forces are limited via selected feasible sets. For robot-assisted placement of an ultrasound probe on a patient's skin, a position control MPC has been proposed in [172]. While the respiratory motion is compensated primarily via position control, an additional force constraint is included in the MPC formulation for the experiments to keep the desired contact force.

There exist a series of results where in the MPC controller combined position and

force control tasks are considered. Simultaneous position and force control via MPC is considered in [49] for robot-assisted dressing. The optimisation goal is to obtain a forward motion with the robot while minimising forces on the human arm. These forces are exerted by the cloth, which is held by the robot. However, no constraints on the maximum force are included. A combination of position and force control is often also considered in bilateral teleoperation, where a human operates a master device and a slave robot should follow the commanded motion or force. The contact forces encountered at the slave robot can provide useful feedback to the human operator. In [212], an unconstrained linear quadratic Gaussian regulator performs combined position and force tracking in medical teleoperation. In [125, 209], MPC force controllers are employed on the master robot while the former also limits the motor torques. However, the focus in these papers is often laid on the compensation of communication delay instead of explicit contact force consideration, modelling or limitation. Besides application-oriented research, conceptual work to combine MPC with indirect force control has been conducted. Hybrid admittance control of robotic manipulators with MPC is considered in [109] and [230]. In [109], constraints on the contact forces are only indirectly considered by constraining the admittance-based position-reference update. In [230], constraints on the contact forces are explicitly included in the admittance MPC formulation to prevent contact loss or excess of maximum forces. While in [109] torque control with nonlinear MPC is considered, in [230] the MPC is based on a velocity controlled robot and a linearised system model. In both cases, the admittance-based MPC adjusts a position reference the robot should follow. Hybrid impedance control of robotic manipulators with MPC is considered in [14]. The cost function of the controller is designed to represent an impedance control law, while additional constraints on the position, velocity, and acceleration are included. As shown, the limitation of the acceleration of the linearised model indirectly also limits the contact forces.

Instead of adding additional admittance dynamics or imposing impedance behaviour in the MPC formulation, force control can also be achieved by the combination of MPC and direct approaches. Unlike direct force control with PID controllers, direct force control with MPC potentially lowers the tendency for oscillations that can lead to instability due to the model-based prediction. These predictions allow evaluating the long term effect of controller actions. Consequently, this makes the controller foresighted and less prone to oscillatory behaviour or overshooting. Hence, MPC force control does not necessarily need the intermediate layer of artificial compliance. This allows for reduced computational complexity, as no dynamics for the admittance are added. Rather, compliance can also be obtained in direct predictive force control via tuning of the weights in the cost function and the length of the prediction horizon. Direct force control with MPC in a medical setup for beating heart motion compensation was proposed in [40]. The MPC formulation is based on a linearised system model representing the force and its derivative as system states. Constant reference

forces and no constraints on the forces have been included. In [154], direct and hybrid position-force control is considered. Besides the explicit limitation of contact forces even in case of disturbances, the MPC follows a non-constant desired force reference. In contrast to [109, 230], no update of position references is used to encode the desired forces. A generic framework for model predictive force control is presented in [80, 81], where the force is added as an additional state variable. We refrain from the additional differentiation and state expansion and propose a model predictive force controller that considers the force as an output of the dynamic system. Our approach reduces the computational burden, especially when extending the force models by machine learning parts as outlined in later chapters. In the following, direct hybrid force-position control based on [154] is proposed. It makes the benefits of MPC available to force control without adding artificial dynamic equations of the forces or for the interaction. Hence, reduced computational complexity compared to indirect force control with MPC is achieved, while the drawbacks of classical direct PID force control such as oscillations are minimised.

3.2 Direct Force Control with MPC

Following the lines of [154], we show how model predictive controllers can be used to control the contact force together with the manipulator position explicitly. A short introduction to robot and contact force modelling is given in the following before designing the controllers and evaluating them in simulations and experiments.

3.2.1 Dynamic Modelling of a Robotic Manipulator

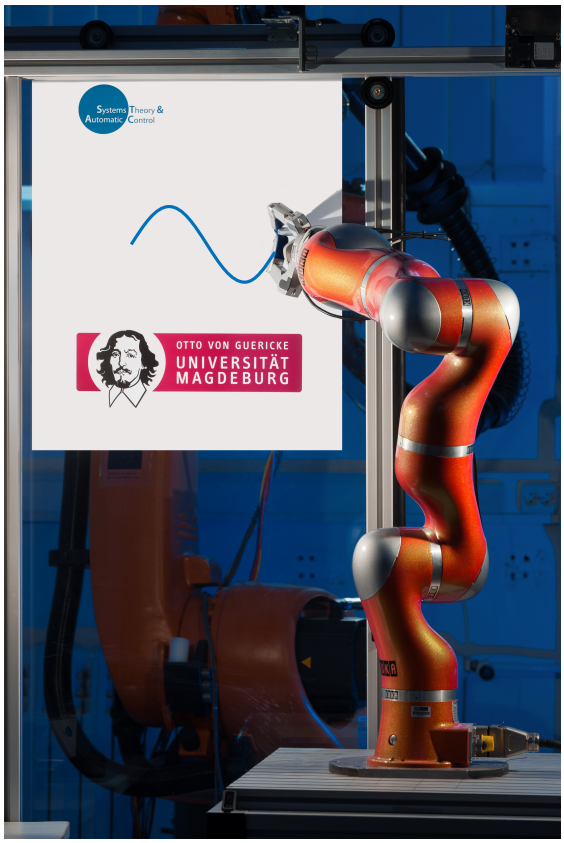
For visualisation and explanation, we consider the control of a robotic manipulator in contact with an environment, cf. Figure 3.1a. The dynamical model of a robotic manipulator with n_q links can in general be obtained via the Lagrangian formulation [210]. The Lagrangian $\mathcal{L} : \mathbb{R}^{n_\mathrm{q}} \times \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}$ is given by

$$\mathcal{L}(q, \dot{q}) := \mathcal{T}(q, \dot{q}) - \mathcal{W}(q),$$

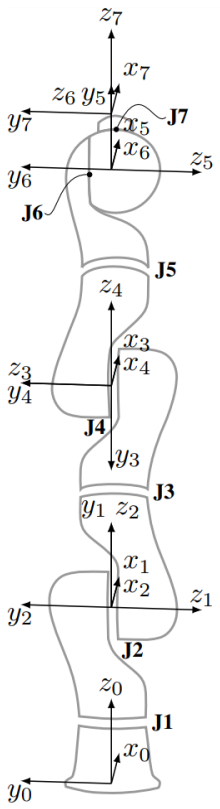
where $\mathcal{W} : \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}$ and $\mathcal{T} : \mathbb{R}^{n_\mathrm{q}} \times \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}$ are the potential and kinetic energy. Since the kinetic and potential energy of a rigid link manipulator depend on the joint coordinates $q \in \mathbb{R}^{n_\mathrm{q}}$ and their derivatives $\dot{q} \in \mathbb{R}^{n_\mathrm{q}}$ the equations of motions can be derived via the Lagrangian equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}}^\top + \frac{\partial \mathcal{L}}{\partial q}^\top = \xi_\mathrm{g}.$$

Here, $\xi_\mathrm{g} \in \mathbb{R}^{n_\mathrm{q}}$ encodes the generalised forces which are composed of joint actuation torques τ, gravitational torques τ_g, friction torques τ_f and external torques τ_ext applied by an interaction with the environment. Following standard modelling schemes and



(a) Seven link KUKA Lightweight robot in the control theory laboratory of our institute writing on a surface (picture similar to [61]).



(b) Coordinate assignment in accordance to the Denavit-Hartenberg convention. Picture adjusted based on [11].

Figure 3.1: Robotic manipulator considered for model predictive force control.

assumptions for the kinetic and potential energy of a robot manipulator [210] a second order dynamical model of the form

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_{\mathrm{f}}(\dot{q}) + \tau_{\mathrm{g}}(q) = \tau - \tau_{\mathrm{ext}} \tag{3.1}$$

can be obtained. The configuration dependent inertia matrix of the robot is denoted by $B : \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{n_{\mathrm{q}} \times n_{\mathrm{q}}}$ and the Coriolis and centrifugal effects are captured by $C : \mathbb{R}^{n_{\mathrm{q}}} \times \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{n_{\mathrm{q}} \times n_{\mathrm{q}}}$. While the gravitational force $\tau_{\mathrm{g}} : \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{n_{\mathrm{q}}}$ for a rigid link manipulator depends only on joint positions, the friction torque $\tau_{\mathrm{f}} : \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{n_{\mathrm{q}}}$ can be modelled via viscous and Coulomb friction such that is depends on joint velocities. We consider contact of the robot with the environment at the end-effector. The contact forces and moments occurring at the end-effector in a three-dimensional Cartesian space are captured by $F \in \mathbb{R}^6$. They can be mapped to the corresponding joint torques via $\tau_{\mathrm{ext}} = J(q)^\top F$ with the manipulator Jacobian $J : \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{6 \times n_{\mathrm{q}}}$.

To obtain a state space representation of (3.1) we consider the system states $x =$

$(q, \dot{q})^\top, x \in \mathbb{R}^{2n_\mathrm{q}}$ and inputs $u = \tau$ with $u \in \mathbb{R}^{n_\mathrm{q}}$. Hence, (3.1) can be rewritten into

$$\begin{pmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_{n_\mathrm{q}} \\ \dot{x}_{n_\mathrm{q}+1} \\ \vdots \\ \dot{x}_{2n_\mathrm{q}} \end{pmatrix} = \underbrace{\begin{pmatrix} x_{n_\mathrm{q}+1} \\ \vdots \\ x_{2n_\mathrm{q}} \\ B^{-1}(x_1, \ldots, x_{n_\mathrm{q}}) \left(u - J^\top(x_1, \ldots, x_{n_\mathrm{q}}) F - N(x) \right) \end{pmatrix}}_{f(x,u)} \tag{3.2}$$

$$\text{with } N(x) = C(x) \begin{pmatrix} x_{n_\mathrm{q}+1} \\ \vdots \\ x_{2n_\mathrm{q}} \end{pmatrix} + \tau_\mathrm{f}(x_{n_\mathrm{q}+1}, \ldots, x_{2n_\mathrm{q}}) + \tau_\mathrm{g}(x_1, \ldots, x_{n_\mathrm{q}}).$$

3.2.2 System Output and Task Definition

Equation (3.2) describes the state space dynamics of a fully actuated, rigid, open-chain, multiple-link robotic arm in joint coordinates. In Chapter 2, all MPC controllers are designed in the output space such that the variables of interest, i.e. the controlled variables, are selected via the definition of the system output. Given this setup, the same dynamical model (e.g. equation (3.2)) can be used while different control goals can be pursued via the selection of different system outputs and references. For hybrid position and force control, some parts of the output and the corresponding reference are poses, and some are forces. To avoid conflict between position and force references, position- and force-controlled subspaces can be defined. This separation requires knowledge about the task to be performed. For example, if we consider a robot writing on a whiteboard, cf. Section 3.3, the force acts perpendicular to the whiteboard. The movement along its surface, i.e. in the orthonormal directions, is position controlled. Classical hybrid force position control aims at decoupling the closed-loop control for position and force completely. However, in our setup, the model predictive control is simultaneously controlling all outputs to follow their respective references. Exact trackability or followability can be achieved if the references are designed appropriately. To do so, the references for the position and force must not contradict themselves. From a practical point of view, MPC can also tradeoff between conflicting control goals based on the specific weightings defined in the cost functional.

We consider unilateral, single-point contact of the end-effector with the environment, such that one component of the reference is selected to be the contact force normal to the environments surface with dimension $n_\mathrm{F} = 1$. A complementary subspace of the task space is chosen via the selection matrix $S \in \mathbb{R}^{(n_\mathrm{y}-n_\mathrm{F})\times n_\mathrm{y}}$, which is position controlled. This decoupling can be achieved, for instance, if the force-controlled direction builds one axis of the base frame and the robot end-effector pose is controlled only

along the remaining directions. Consequently, the system output is defined as

$$y = \begin{pmatrix} Sh_{\mathrm{fk}}(q) \\ h_F(q, \dot{q}) \end{pmatrix}, \tag{3.3}$$

where $h_{\mathrm{fk}} : \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{n_{\mathrm{y}}}$ represents the forward kinematics of the robot mapping from joint space to the end-effector pose and $h_F : \mathbb{R}^{n_{\mathrm{q}}} \times \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}^{n_{\mathrm{F}}}$ is a static prediction model describing the interaction force. More details on the forward kinematics, as well as the coordinate transformations and parameters involved in this mapping, can be found in Appendix A.2.

Setting up the base frame, the selection matrix S, and the desired motions and forces via $r_{\mathrm{s}}, r_{\mathrm{tt}}$, or r_{pf}, is in general part of path or motion planning. In this chapter, we focus on the control to achieve a given task defined via a preplanned reference. However, by considering setpoint regulation, trajectory tracking and path following, different requirements on the path planning are specified. While in setpoint regulation the reference is constant, trajectory tracking uses a time-dependent trajectory with fixed speed along the reference. Path following uses a parametrised curve composed of positions and corresponding forces at these positions without explicit time dependency. Path-following control can lower the effort for the path planner compared to tracking and brings the two concepts of motion planning and control closer together. Hence, path following merges aspects which are traditionally considered during planning with those from online control. In the following, three tasks are specified, which consider hybrid position and force control for robotic manipulators. They consider setpoint regulation, trajectory tracking and path following, respectively. The task definition formulations follow along the lines of [56, 153, 154].

First, a constant pose and force reference r_{s} is considered, which is assumed to be feasible. It contains $n_{\mathrm{y}} - 1$ pose references and one desired force value, which reflects the desired normal contact force. Examples are control tasks where the robot should hold a specific posture while applying a constant force as e.g. in assembling, when a workpiece should be glued to another by pressing it against it for some time while the glue is drying. The related control task is specified as follows:

Task 1 (Constant pose and force reference)**.** *Given the system* (3.2) *with output* (3.3) *and the reference point* $r_{\mathrm{s}} \in \mathcal{Y}$*, design a controller that achieves:*

(i) Convergence: The system output (3.3) *converges to the constant reference* r_{s} *such that* $\lim_{t\to\infty} y(t) - r_{\mathrm{s}} = 0$*.*

(ii) Constraint Satisfaction: The constraints on states $x \in \mathcal{X}$*, inputs* $u \in \mathcal{U}$ *and outputs* $y \in \mathcal{Y}$ *are satisfied for all times.*

The second task considers trajectory tracking of force and pose references. To this end, a reference trajectory $r_{\mathrm{tt}} : \mathbb{R}_0^+ \to \mathbb{R}^{n_{\mathrm{y}}}$ is defined, which consists of $n_{\mathrm{y}} - 1$ poses

and one force component. It is assumed that this reference is known a priori and fulfils constrained trackability according to Definition 1. Based on this setup, the following control task can be posed.

Task 2 (Reference tracking)**.** *Given the system* (3.2) *with output* (3.3) *and the reference trajectory* $r_{\mathrm{tt}}(t) \in \mathcal{Y}$*, design a controller that achieves:*

(i) Convergence: The system output (3.3) *converges to the reference trajectory* r_{tt} *such that* $\lim_{t\to\infty} y(t) - r_{\mathrm{tt}}(t) = 0$*.*

(ii) Constraint Satisfaction: The constraints on states $x \in \mathcal{X}$*, inputs* $u \in \mathcal{U}$ *and outputs* $y \in \mathcal{Y}$ *are satisfied for all times.*

Alternatively, many robotic control tasks can be formulated as path following problems. The path planner does not need to assign a velocity profile to the path. Instead, the evolution of the reference is determined by the controller via the virtual system (2.13). Examples are polishing, deburring tasks, as well as robot-supported rehabilitation where given positions and forces should be transversed without explicit time dependency. The path $\mathcal{P}$ from (2.12) describes pose and force references. This time, the reference for the force (and the pose) does not explicitly depend on time via a fixed timing law. Rather, the desired force is assigned to each pose reference and these are jointly defined in $\mathcal{P}$. The path $\mathcal{P}$ is assumed to be known a priori and is followable according to Definition 2. The following control task can then be specified:

Task 3 (Path following force control)**.** *Given the system* (3.2) *with output* (3.3)*, and the path* $\mathcal{P}$*, design a controller that achieves:*

(i) Convergence: The system output (3.3) *converges to the set* $\mathcal{P}$ *such that* $\lim_{t\to\infty} y(t) - r_{\mathrm{pf}}(\theta(t)) = 0$*.*

(ii) Forward Motion along the Path: The reference $r_{\mathrm{pf}}(\theta(t))$ *moves in* $\mathcal{P}$ *along the direction of increasing values of* θ*, i.e.* $\dot{\theta}(t) \geq 0$ *and* $\lim_{t\to\infty} \theta(t) = \theta_{\mathrm{end}}$*.*

(iii) Constraint Satisfaction: The constraints on states $x \in \mathcal{X}$*, inputs* $u \in \mathcal{U}$ *and outputs* $y \in \mathcal{Y}$ *are satisfied for all times.*

Tasks 1-3 contain the requirement of constraint satisfaction in the output space. In hybrid position and force control, this also encodes constraints on the minimum and maximum normal contact forces. This way, the fulfilment of Task 1, 2, and 3 encodes a notion of safety that prevents from contact loss and potential damage by high contact forces, which is important in industrial settings (damage on workpieces or the robot) as well as in human-robot cooperation. To obtain a measure for the considered contact force, a short recap on contact force modelling is given in the following.

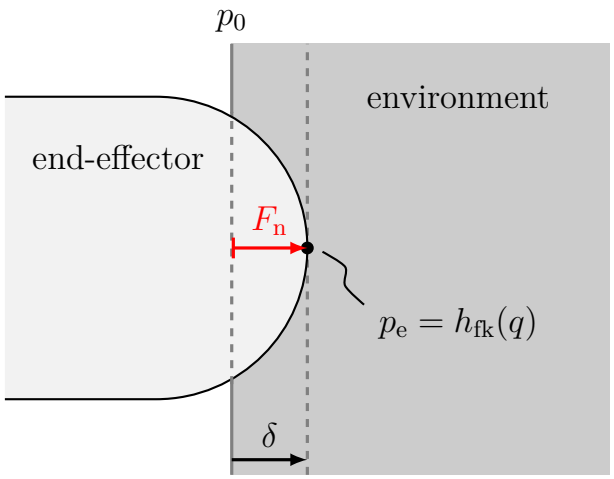


Figure 3.2: The penetration depth δ is defined as the distance between the end-effector pose $p_\mathrm{e} = h_\mathrm{fk}(q)$ and the initial contact position p_0 along the normal of the environment surface. The resulting normal force F_n is modelled as a function of this penetration depth.

3.2.3 Contact Force Modelling

The output (3.3) contains a model of the contact forces besides the forward kinematics of the robot. When the robot is in contact with a compliant environment, a static linear or nonlinear model can be used to describe this interaction forces. We consider the contact force normal to the environment surface hence $n_\mathrm{F} = 1$. Note that several other choices to quantify the interaction are possible as, e.g., taking the full six-dimensional force-torque vector F in the end-effector coordinate frame into account. Still, it has to be ensured that $n_\mathrm{u} = n_\mathrm{y}$ to preserve full actuation of the manipulator. In [67], an overview, literature review, and comparison of different commonly used force models for the normal force are presented. Four of these modelling approaches for the normal contact force are given in the following. Each of these models depends on the penetration depth δ, cf. Figure 3.2. For a static environment it can be modelled as a function of the joint coordinates via $h_\delta : \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}, (q) \mapsto \delta := h_\delta(q)$. Here, $h_\delta(q) = h_\mathrm{p}(h_\mathrm{fk}(q))$ is the composition of the functions $h_\mathrm{p} : \mathbb{R}^{n_\mathrm{y}} \to \mathbb{R}$ and $h_\mathrm{fk} : \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}^{n_\mathrm{y}}$, where the end-effector pose $p_\mathrm{e} := h_\mathrm{fk}(q)$ is obtained via the forward kinematics, cf. Figure 3.2 and Appendix A.2.

Among the purely elastic models, which can be used to quantify the interaction, are the linear spring model of Hook

$$h_{F_\mathrm{Hk}}(q) := K_\mathrm{e}\delta = K_\mathrm{e}h_\delta(q) \tag{3.4}$$

and the nonlinear spring model of Hertz

$$h_{F_\mathrm{Hz}}(q) := K_\mathrm{e}\delta^\alpha = K_\mathrm{e}(h_\delta(q))^\alpha, \tag{3.5}$$

where $K_\mathrm{e} \in \mathbb{R}$ is a spring constant and the coefficient $\alpha \in \mathbb{R}_0^+$ introduces nonlinearity

and is equal to 1.5 in the original work of Hertz. Another popular choice is to model the interaction forces via a parallel linear spring and damper system (Kelvin-Voigt model)

$$h_{F_{\mathrm{KV}}}(q,\dot{q}) := K_{\mathrm{e}}\delta + D_{\mathrm{e}}\dot{\delta} = K_{\mathrm{e}}h_{\delta}(q) + D_{\mathrm{e}}h_{\dot{\delta}}(\dot{q}) \tag{3.6}$$

where $\dot{\delta} := h_{\dot{\delta}}(\dot{q})$, with $h_{\dot{\delta}} : \mathbb{R}^{n_{\mathrm{q}}} \to \mathbb{R}$, is the penetration velocity and D_{e} is the damping coefficient of the environment. The Hunt-Crossley model builds a nonlinear extension of the Kelvin-Voigt model and is given by

$$h_{F_{\mathrm{HC}}}(q,\dot{q}) := K_{\mathrm{e}}\delta^{\alpha} + D_{\mathrm{e}}\delta^{\alpha}\dot{\delta} = K_{\mathrm{e}}h_{\delta}^{\alpha}(q) + D_{\mathrm{e}}h_{\delta}^{\alpha}(q)h_{\dot{\delta}}(\dot{q}) \tag{3.7}$$

where the coefficient α is usually chosen between 1 and 2. Especially the dependency of the damper on the penetration depth makes the Hunt-Crossley model a more realistic representation of physical interactions than the Kelvin-Voigt model [67]. Additionally, the nonlinear nature allows obtaining better representations of interactions with soft materials, such as for the interaction of robotic surgical systems with human tissue [39, 86]. In MPC related publications, often purely elastic models are used, cf. [111, 112, 125, 154, 170]. Especially for more rigid materials, these build a suitable first-order approximation of the occurring normal contact forces. At the same time, the effort for the identification of model parameters reduces significantly, cf. [86, 134] for an overview of identification methods for the different model structures.

The elastic and dissipative contact force models are valid only if the robot is in contact with the environment. For example, a linear spring model produces negative sticking or pulling forces for negative penetrations, i.e. if the robot is not in contact. Some spring models might not even be defined for negative penetrations and would result in imaginary forces. Consequently, to obtain globally valid models often piecewise definitions of the maps such as

$$h_F(q,\dot{q}) = \begin{cases} h_{F_i}(\cdot) & \delta \geq 0, \\ 0 & \text{else} \end{cases}$$

are used. Clearly, h_F is in general nonlinear even for linear functions h_{F_i} with $i \in \{\mathrm{Hk,Hz,KV,HC}\}$. In the general case, h_F will not be continuously differentiable, which makes it unsuited for many classical control schemes, including the MPC schemes proposed in Chapter 2. Either an approximation to avoid the discontinuity, for instance proposed in [70] and depicted in Figure 3.3, can be used or an expansion to hybrid MPC is necessary. Alternatively, the considered motions of the robot should ensure contact for all times to ensure differentiability in the feasible area. MPC can be used to achieve the second idea via constraining the force to be greater than zero. All in all, each of the force models (3.4)-(3.7) can be used as part of the system output (3.3) such that

$$h_F(q,\dot{q}) = h_{F_i}(q,\dot{q}) \tag{3.8}$$

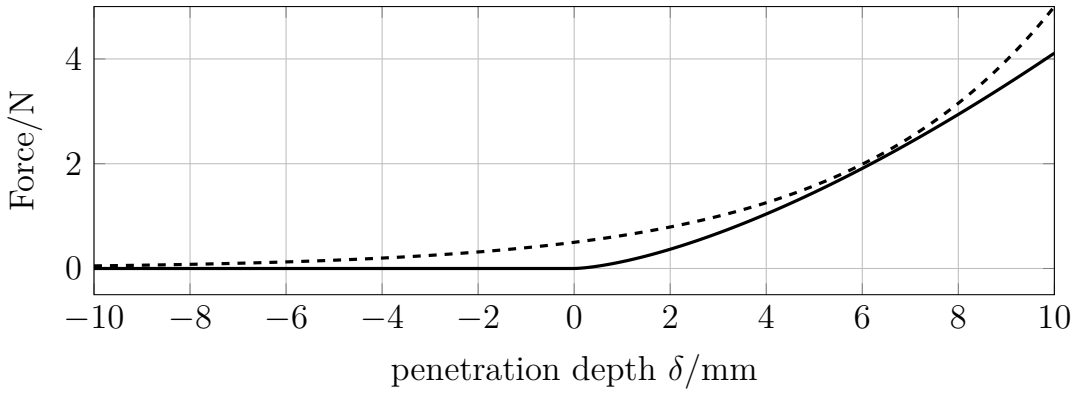


Figure 3.3: Two nonlinear elastic contact force models. The solid line is a nonlinear spring model with discontinuity at $\delta = 0$ and the dashed line is an exponential spring model proposed in [70], which is continuously differentiable everywhere.

with $i \in \{\text{Hk,Hz,KV,HC}\}$ under the assumption that the model predictive controller ensures contact for all times.

3.2.4 Model Predictive Force Control: Task Solution and Stability

The nominal MPC schemes presented in Chapter 2 allow to fulfil the requirements stated in the Tasks 1, 2, and 3. Constraint satisfaction, as demanded in Task 1(ii), 2(ii), and 3(iii) can explicitly be addressed by the constraints (2.5f), (2.11f), and (2.16f) in the optimal control problems (2.5), (2.11), and (2.16). Moreover, forward motion (Task 3(ii)) of the reference in path following can be incorporated in MPC via constraints on the virtual system states, cf. (2.16g), and the design of the cost function that includes θ, cf. (2.15).

Convergence of the output to the respective reference from Tasks 1(i), 2(i), and 3(i), can be achieved if the MPC is provably converging or stable. Conditions to prove stability have been outlined in Chapter 2. Besides the design decisions for the setup of the stage cost, terminal cost, and terminal constraints given in the Subsections 2.2.3, 2.3.3 and 2.4.3, Assumptions 1-3 must be fulfilled. As we have specified the dynamic model (3.2) and the system output (3.3) with (3.8) in this chapter, Assumption 2 must be verified. Differentiability of f from (3.2) is ensured whenever each of the involved mappings is differentiable. The entries of τ_g, B, J, and C consist of additive and multiplicative composition of trigonometric functions, see also [210]. Consequently, the elements of τ_g, B, J, and C are continuously differentiable with respect to q and $\dot{q}$ and so are τ_g, B, J, and C. The derivative of the inverse of the inertia matrix can be expressed via $\frac{\mathrm{d}}{\mathrm{d}q}B^{-1}(q) = -B^{-1}(q)\frac{\mathrm{d}}{\mathrm{d}q}B(q)B^{-1}(q)$. It exists when $B(q)$ is of full rank such that $\det(B(q)) \neq 0$. This invertibility does not hold in general for all $q \in \mathbb{R}^{n_\mathrm{q}}$. Instead, singularities of B occur, for instance, when the robotic arm is fully stretched (boundary singularity) or if rotation axes are aligned (internal singularities). The Jacobian builds a useful tool to determine these singularities such that the path

planner and the controller can avoid those points, cf. [210]. Hence, we restrict the joint angles via $q \in \text{proj}_{1,\dots,n_\text{q}}(\mathcal{X})$, where $\text{proj}_{1,\dots,n_\text{q}}(\mathcal{X})$ denotes the components of the constraints restricting the joint angles, such that the following assumption is fulfilled

Assumption 17. *The state constraints $\mathcal{X}$ are chosen such that* $\det(B(q)) \neq 0$ *for all* $q \in \text{proj}_{1,\dots,n_\text{q}}(\mathcal{X})$.

Given Assumption 17, $B^{-1}(q)$ is differentiable. Finally, differentiability of τ_f must be given to conclude differentiability of f. Standard assumptions for modelling viscous and Coulomb friction, such as $F_\text{v}\dot{q}$ and $F_\text{C}\,\text{sgn}(\dot{q})$ with friction constants F_v and F_C, do not fulfil this requirement. Therefore, we propose to use the arctangent function to approximate the signum function such that $\tau_\text{f} = F_\text{v}\dot{q} + F_\text{C}\arctan(M\dot{q})$, where $M \in \mathbb{R}_0^+$ is a large positive number. Using this modelling ansatz, differentiability of all components in the system model are given such that f is continuously differentiable in x and u. Moreover boundedness of the first derivative of f is given at least locally, such that f is locally Lipschitz.

The system output (3.3) contains the forward kinematics h_fk and the force model h_F from (3.8). Similar to τ_g, B, J, and C, the forward kinematics h_fk consist of additive and multiplicative composition of trigonometric functions, cf. Appendix A.2. Hence, h_fk is continuously differentiable. The proposed force models (3.4)-(3.7) in (3.8) are continuously differentiable for positive penetrations and consequently for positive forces. Therefore we include the following condition on the constraints:

Assumption 18. *The state constraints $\mathcal{X}$ are chosen such that* $h_\delta(q) > 0$ *for all* $q \in \text{proj}_{1,\dots,n_\text{q}}(\mathcal{X})$.

Alternatively, we can formulate output constraints such that contact is ensured:

Assumption 19. *The output constraints $\mathcal{Y}$ are chosen such that* $F_\text{n} = h_{F_i}(q) \geq 0$ *for all* $q \in \text{proj}_{1,\dots,n_\text{q}}(\mathcal{X})$*, i.e.* $\text{proj}_{n_\text{y}}(\mathcal{Y}) = [0, F_\text{max}]$*, with* $F_\text{max} \in \mathbb{R}^+$.

All in all, we can conclude that Assumption 2 is fulfilled for the hybrid position and force control task under the given modelling assumptions and constraints. Hence, model predictive control is a provable stable and safe way to achieve a hybrid position and force control if the weights and parameters are suitably chosen. The MPC formulations of Chapter 2 can directly be applied to the hybrid position and force control tasks 1-3. In the following, an application example illustrates these findings.

3.3 Application Example

In this section, the proposed hybrid force and position control is applied to a KUKA Lightweight robot writing on a compliant surface, cf. Figure 3.1a. The provided results have appeared in [154]. A more detailed description, as well as extended results, are presented here. The goal is to control the robot such that its end-effector (the pen

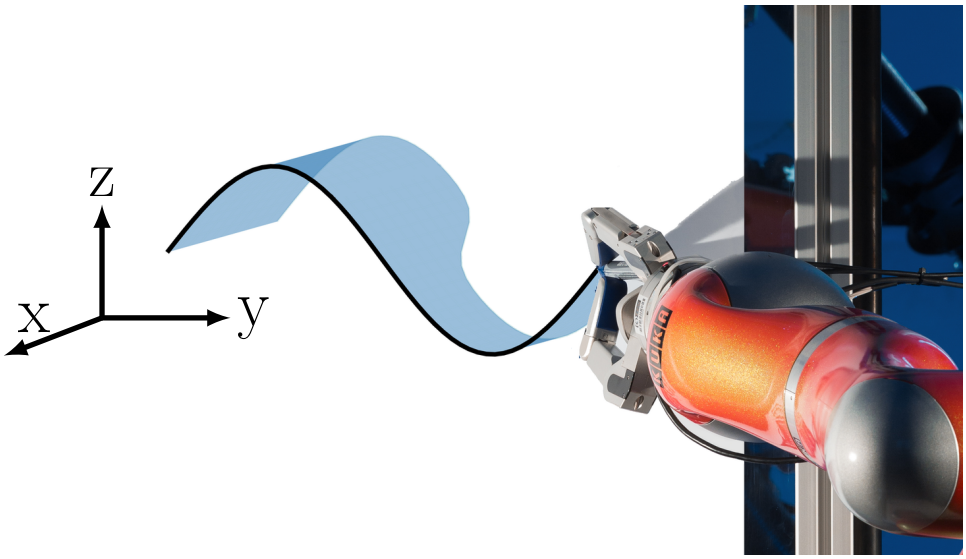


Figure 3.4: Illustration of the base frame and the reference for the robot writing task.

tip) is following a two-dimensional position reference while applying varying forces along it on the whiteboard, i.e. writing with different stroke width. For this purpose, the reference consists of a position-controlled sine curve along the whiteboard surface and a composition of exponential functions for the desired force perpendicular to the whiteboard surface. The whiteboard placement accounts for this task. The base frame y- and z-axes are parallel to the whiteboard, while the x-axis is normal to the surface, see also Figure 3.4. The position reference is depicted as a black line in Figure 3.4 and the force reference is indicated via a blue shaded area along the x-axis. Since the reference consists of three directions, we consider the control of three joints of the robot to obtain a square input-output structure $n_{\mathrm{y}} = n_{\mathrm{u}} = 3$. Robotic writing tasks have, for example, been considered in [61, 154, 234]. In each of these references, some notion of path following is applied. The reference can be adjusted by the controller to react on unforeseen disturbances and to increase control performance. Pure position control with MPC is used in [60, 61] such that interaction forces act as disturbances or model-plant mismatch. In [234] an active compliance control, i.e. an indirect force control method, is used. In contrast to [234] we assume a fixed board position and MPC similar to [61, 154], which allows to consider constraints. In [36], a writing task is performed with a quadcopter. There, a position-controlled MPC is used which includes force constraints to prevent from contact loss, while the positional reference for the pen tip is 5 cm behind the undeformed contact surface. By applying the hybrid position and force controller proposed in this chapter, we directly control the interaction forces with varying desired contact forces. This task is defined via trajectory tracking and path following, such that a comparison between the two approaches is possible. In the following, the description of the involved software and hardware components for simulation and lab experiments is given before discussing the achieved results.

3.3.1 Software and Hardware Setup

We use a KUKA Lightweight robot as depicted in Figure 3.1, with active actuation of the joints one, two and four. To measure the contact forces, a 6DOF force-torque sensor is attached to the wrist of the robot. Moreover, a three-fingered robotic hand is attached to hold the pen. Specifications of the sensor and the robotic hand can be found in [12]. The fast research interface establishes the communication between the robot and the work station PC [206], and a CAN-bus builds the connection to the wrist sensor. The communication interface for the control of the robot from MATLAB was designed in [11]. The sensor interface was set up in [18]. The controller implementation in Matlab uses the code-generation toolkit of ACADO to solve the involved optimisation problem in real-time [7]. The computations are performed on a Linux workstation PC with Ubuntu 12.04 and an Intel Xeon(R) X5675 processor with 3.07 GHz x6.

3.3.2 Control Setup and Parameters

We compare the performance of two model predictive controllers for tracking and path following designed in accordance with (2.11) and (2.16). The system model used in (2.11b) and (2.16b) is (3.2), with $x = (q_1, q_2, q_4, \dot{q}_1, \dot{q}_2, \dot{q}_4)^\top$. The robot parameters were identified in [11] and adjusted by [18] to include the Barrett hand and force-torque sensor. This dynamical model is used for simulations as well as for prediction in the model predictive controller. However in contrast to [11], $N(x) = 0$ is considered. Several design choices justify this simplification. First, the gravity is compensated by an internal controller provided by the KUKA control setup. Second, friction and Coriolis effects only have a minor influence at the considered small velocities. Third, a compensation of external torques outside of the predictive controller is added to the optimal inputs just as the torques for gravity compensation. The contact force model was designed and parametrised in [18] and corresponds to the linear spring model (3.4). Damping effects have been neglected due to noisy sensor readings in the joint velocities as well as small penetration velocities.

In the path following case, the double integrator $\dot{z}_1 = z_2, \dot{z}_2 = v$ is used as virtual system (2.16c). The system outputs are given via $y_{\mathrm{tt}} = (p_{\mathrm{e,y}}, p_{\mathrm{e,z}}, F_{\mathrm{n}})^\top$ (see also (3.3)) and $y_{\mathrm{pf}} = (p_{\mathrm{e,y}}, p_{\mathrm{e,z}}, F_{\mathrm{n}}, z_1)^\top$, where $p_{\mathrm{e,y}}$ and $p_{\mathrm{e,y}}$ are the end-effector (pen tip) position in y- and z-direction, while F_{n} denotes the normal contact force between the pen and the whiteboard in x-direction of the base frame. Note that in path following, an additional output is included, which represents the path parameter $\theta = z_1$.

We refer to the outputs of the system as variables of interest, which might or might not be directly measurable. For instance, the Cartesian positions of the end-effector are not measured directly. As our robot is equipped with a force-torque sensor at the wrist, we would like to incorporate these measurements besides the joint sensor readings on positions and velocities. In the nominal case, this additional information

is unnecessary. In the experimental setup, however, a model-plant mismatch exists. Incorporating additional information can lower its effect. If the contact force would have been modelled via a dynamic equation, this information is naturally included via the initial conditions. Inspired by this principle, the model-plant mismatch for the contact force will be used to update the output model such that for the optimisation starting at time t_k the prediction model for the contact force model is

$$\bar{F}_\mathrm{n} = K_e h_\delta(\bar{q}) + \tilde{F}_\mathrm{n}(t_k). \tag{3.9}$$

Here, $\tilde{F}_\mathrm{n} \in \mathbb{R}$ denotes the model-plant mismatch based on measurements at t_k which is given by $\tilde{F}_\mathrm{n}(t_k) = h_\mathrm{n}(\tilde{F}(t_k)) - K_e h_\delta(\tilde{q}(t_k))$. The predictions inside the controller are denoted by $\bar{\cdot}$ while $\tilde{\cdot}$ indicate sensor readings. The map $h_\mathrm{n} : \mathbb{R}^6 \to \mathbb{R}$ denotes the transformation of the measured end-effector wrench into the normal contact force. This formulation is equivalent to a dynamic interpretation of the contact force model and its differentiation to include the force as a part of the system states. At the same time, it does not require the solution of an artificially added dynamic, which is not necessary to describe the physical relations.

The mathematical equations for the reference definition in the path following case is given in Appendix A.4. For trajectory tracking, the reference is defined according to the first three components of r_pf. As discussed in Chapter 2, this includes the specification of a timing law. To make a fair comparison between path following and tracking, the same timing law as in the path following case is assigned. In other words, the timing law provided by path-following MPC is used for the tracking MPC as an a priori planned, timed trajectory. The cost functions of the involved predictive controllers are given by $L_i = e_i^\top Q_i e_i + u_i^\top R_i u_i$, $E_i = \varepsilon_i^\top Q_{\mathrm{E},i} \varepsilon_i$, with $i \in \{\mathrm{tt}, \mathrm{pf}\}$. The weighting matrices, controller parameter values, and constraints are given in Appendix A.4.

3.3.3 Simulation Results

Figure 3.5 depicts the system outputs and corresponding output errors for the implementation of a trajectory tracking MPC (dashed black line) and path following MPC (solid blue line). In path following, the evolution of the reference (thin black line) is determined by the predictive controller. This is done simultaneously to the control of the robotic system following the reference. The optimal evolution of the virtual system state $z_1 = \theta$ is shown in Figure 3.6, top left. It is determined via the choice of the virtual input v in Figure 3.6, bottom left and its double integration. The virtual state $z_2 = \dot{\theta}$ shows the reference velocity profile. The same optimised profile is chosen for the tracking trajectory. Hence in Figure 3.5, both the trajectory tracking controller and the path following controller achieve the same accuracy. However, this is only true in the nominal case when no disturbances or uncertainties occur. In the following, a reproducible input disturbance of $-3\,\mathrm{Nm}$ is added to the first joint

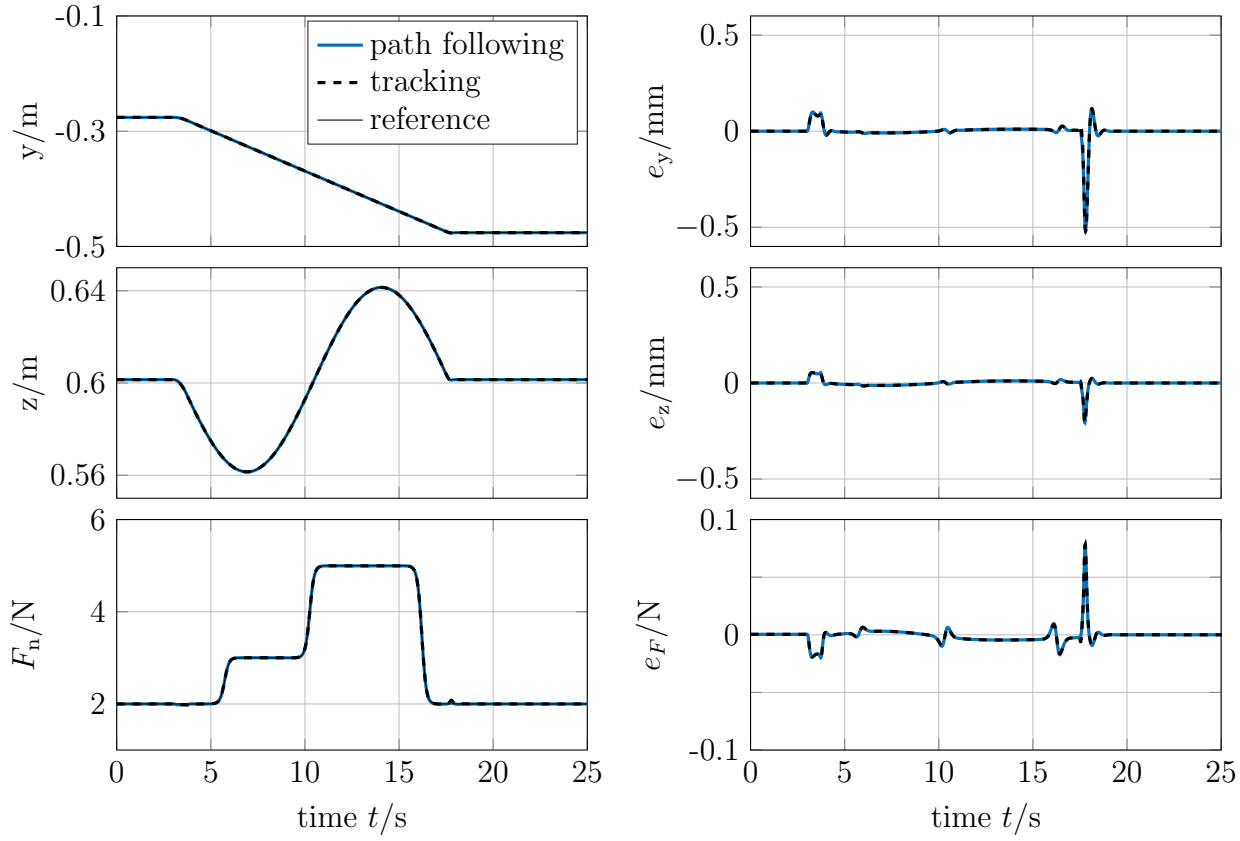


Figure 3.5: System outputs and control errors for tracking and path following MPC.

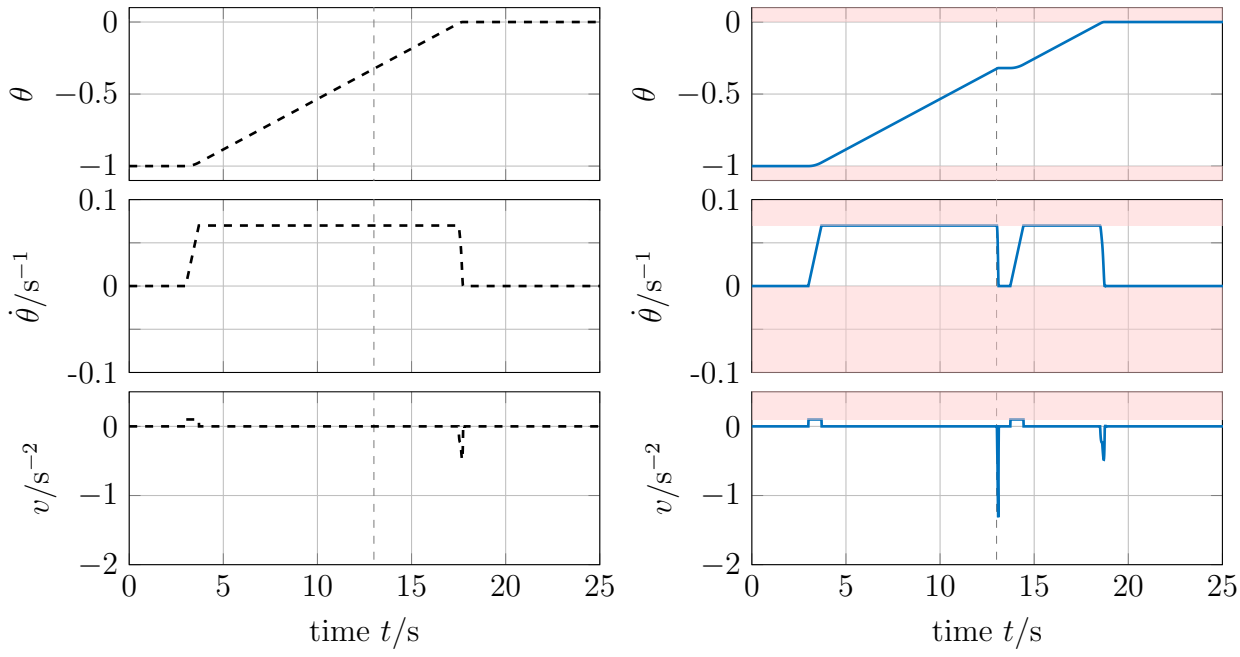


Figure 3.6: The timing law of the reference for trajectory tracking equals the virtual system evolution for path following without disturbance (left). Virtual system evolution for path following with disturbance (right, blue).

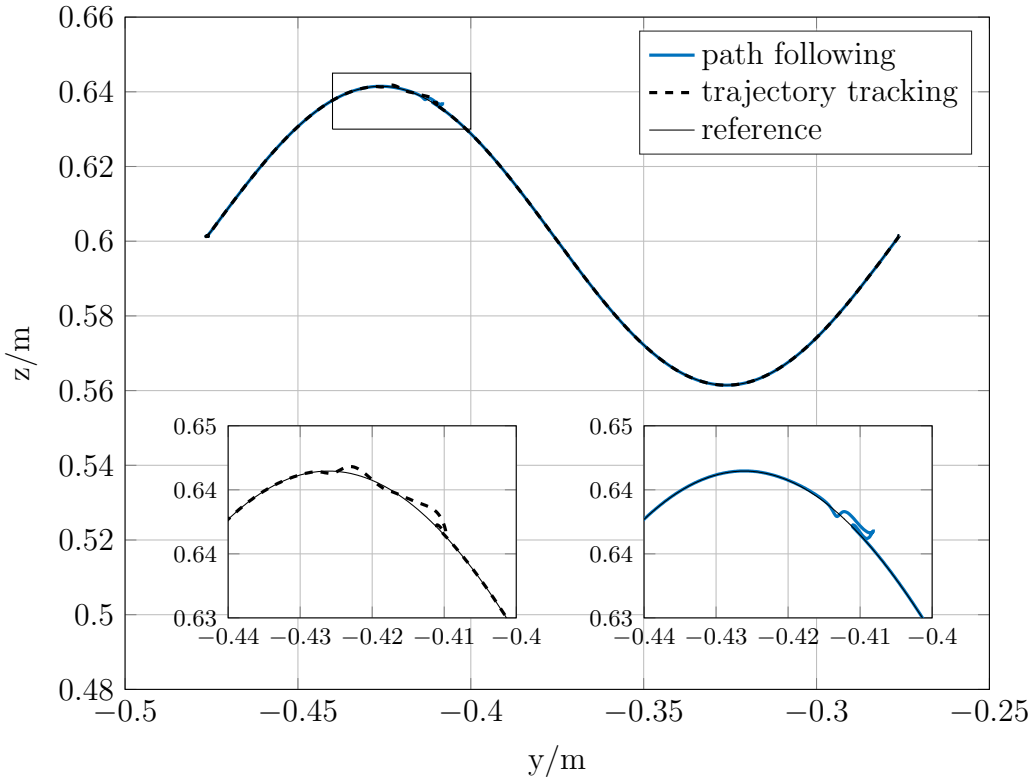


Figure 3.7: End-effector position in Cartesian plane for tracking and path following MPC.

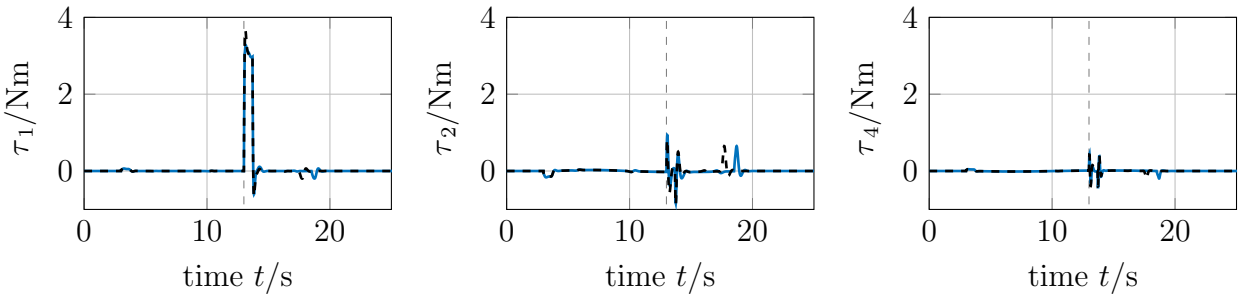


Figure 3.8: Control input torques of the considered joints for tracking and path following MPC.

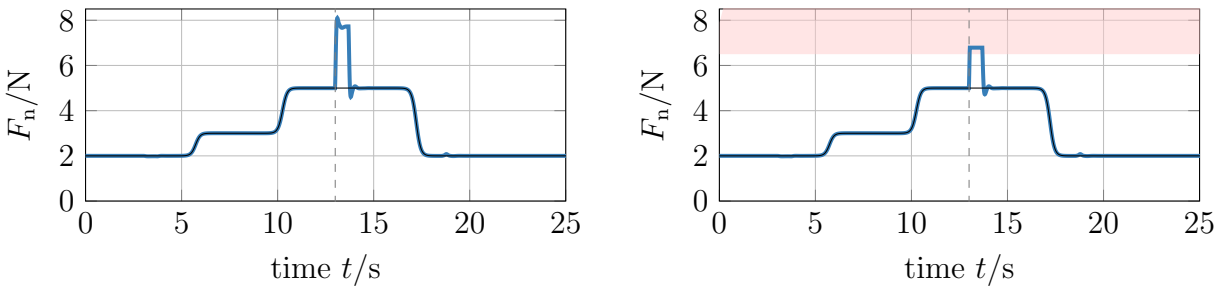


Figure 3.9: Contact force in disturbed path following MPC without (left) and with (right) constrained maximum force.

Table 3.1: Maximum and average computation times for simulations and experiments.

	Total time		OCP		Wrist Sensor	
trajectory tracking	max.	aver.	max.	aver.	max.	aver.
in simulation	1.2 ms	1.0 ms	0.8 ms	0.6 ms	-	-
path following	max.	aver.	max.	aver.	max.	aver.
in simulation	8.7 ms	2.1 ms	8.5 ms	1.8 ms	-	-
path following	max.	aver.	max.	aver.	max.	aver.
in experiment	10 ms*	6.5 ms	10 ms*	3.7 ms	7.2 ms	2.5 ms

during $t \in [13\,\mathrm{s}, 13.7\,\mathrm{s}]$ for both cases. In such a situation, the path following MPC and trajectory tracking MPC behave differently, which is shown in Figure 3.7. While the disturbance acts on the system, precise tracking of the reference becomes more difficult. The path following controller can adjust the speed of the reference via the virtual input to compensate for the effect of the disturbance. In the presented case, it slows down the reference, cf. Figure 3.6 right. In contrast, the tracking MPC cannot adjust the predefined reference and tries to follow it. This inflexibility results in larger Cartesian deviations in Figure 3.7 compared to the path following case. The torque control inputs in both cases are close to each other, cf. Figure 3.8. The main difference arises from the additional degree of freedom of the path following MPC to adapt the reference.

Besides a deviation from the Cartesian path, the input disturbance has a large effect on the contact force. Figure 3.9 (left) shows a drastic increase of the contact force during disturbance. Large forces are often undesired to prevent tool or environment damage. In the case of the robotic writing task, the pen tip is very fragile and pressed flat if the contact forces become too high. One of the benefits of direct force control with MPC is the possibility to include force constraints. Hence, a constraint on the maximum force is included besides the lower constraint to prevent from contact loss. The output constraints are chosen to be $\mathcal{Y} = [0, 6.5\,\mathrm{N}]$ in a subsequent trial, which is shown in Figure 3.9, right. A significant reduction of the contact force despite the disturbance is achieved. However, a small violation of the constraints occurs. This violation occurs as the disturbance directly acts at the input of the robot and is unknown to the controller. Hence, a mismatch between predictions of the controller, which satisfies the constraints, and the actual contact forces appears. From the practical side, a backoff of the constraints to robustify the closed-loop control performance can cope with this effect. All in all, safe performance with reduced and limited contact forces (besides consideration of multiple other constraints) can be achieved.

*package drop for times greater than $T_\mathrm{s} = 10\,\mathrm{ms}$

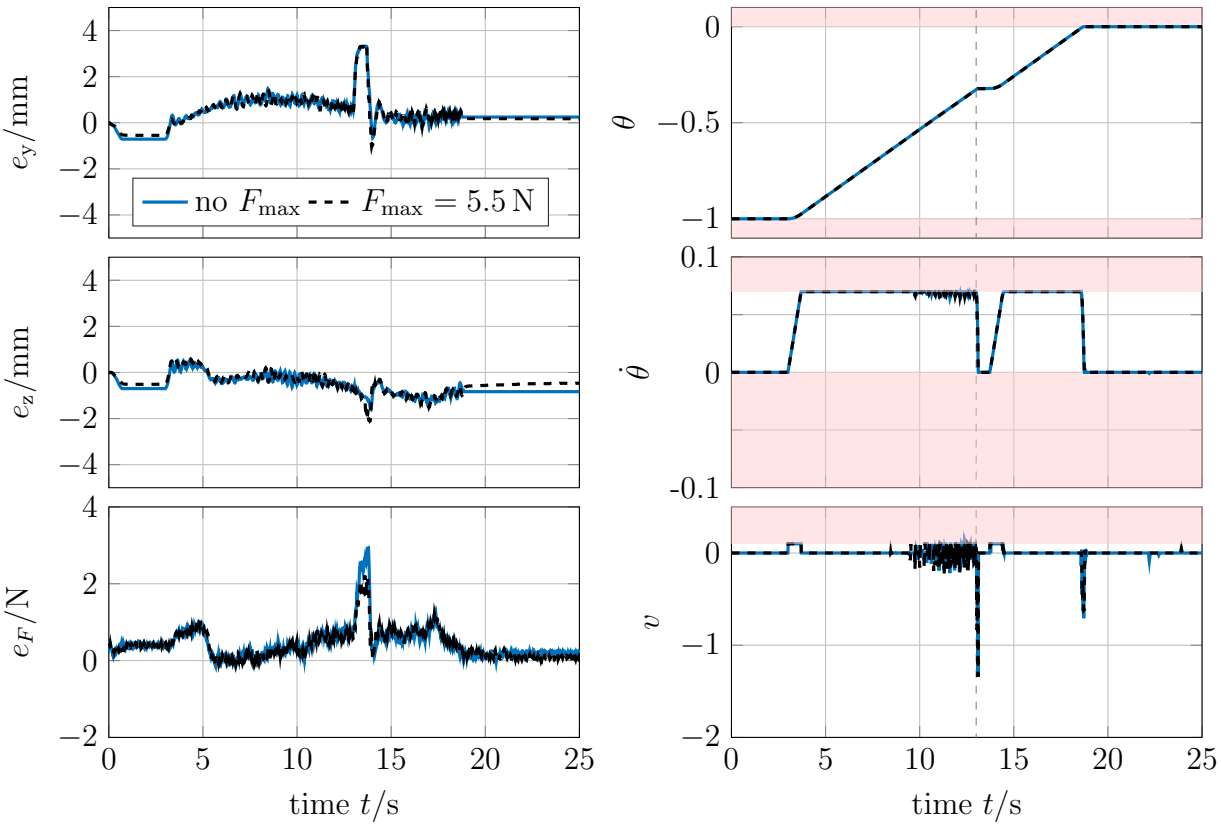


Figure 3.10: Control errors (left) and virtual system evolution (right) for path following MPC in experiments with and without actively constrained maximum force.

3.3.4 Experimental Results

Based on the successful evaluation of the proposed hybrid position and force control MPC scheme in simulations in the previous section, lab experiments on the robot have been performed. Preliminary work for these experiments, especially setting up the communication, models and a preliminary implementation of a predictive, hybrid position-force controller, can be found in [18]. The results presented here build upon these preliminary works and were published in [154]. The same controller setup is used as in the previous section. Only the Cartesian reference is slightly shifted in z-direction compared to the simulations due to imprecise manual board placement. To evaluate the real-time feasibility of the simulations and experiments, the average and maximum computation times are listed in Table 3.1. This table shows the total computation times as well as the time needed to solve the optimal control problem (OCP) and the time for the sensor communication, which build the most extensive parts of the total times. In all simulations, the total maximum computation times lie below the sampling time of $T_\mathrm{s} = 10\,\mathrm{ms}$, even though the path following implementation has higher calculation times. This difference originates mainly from the inclusion of constraints for the virtual system, which are active most of the time. Since the path following case is superior to the tracking formulation in terms of performance

when disturbances occur, we present and compare two path-following MPC schemes in the following. In both cases the same input disturbance of $-3\,\mathrm{Nm}$ is applied at $t \in [13\,\mathrm{s}, 13.7\,\mathrm{s}]$ to the first joint. In the second trial, an additional force restriction to $\mathcal{Y} = [0, 5.5\,\mathrm{N}]$ is added. The respective computation times are shown in Table 3.1. Even though the average total computation time is below T_s, the maximum total computation time occasionally would have exceeded this threshold. This resulted in up to five communication package drops by the fast research interface to the robot in the experiments. These outliers always occur at the end of the path when the path parameter θ becomes zero. However, stable communication despite the small amount of dropped packages was achieved. The achieved control errors are shown in Figure 3.10 (left) for the case with and without this additional force constraint. In Figure 3.10 left, solid blue lines indicate the case without active force constraints, while the black dashed line shows the case with active maximum force constraint of 5.5 N. The control errors are noisy and significantly larger than in the ideal simulations. This originates from several aspects: First, the sensor data is corrupted with noise, which is significant especially in the joint velocities, joint torques, and in the force sensor, see also [12, 18]. Second, there exists a model-plant mismatch between the prediction model also used for simulation and the real system. For example, Coriolis and friction effects have been neglected during prediction. Moreover, the considered force model is linear. Nevertheless, the designed MPC allows the robot to follow the reference path with control errors smaller then 1.53 mm and 1.04 N in the undisturbed case and with 3.38 mm and 2.94 N in the disturbed case. When the disturbance occurs, the virtual system is adjusted by the controller to lower the effect of it, cf. Figure 3.10, right. Loosely speaking, the reference is forced to wait for the disturbed system to minimise the control errors. At the same time, the chosen torque control inputs lower the effect of the disturbance further, see also Figure 3.11. The position-controlled subspace of the experimental path following MPC is shown in Figure 3.12. In both cases (with and without active maximum force constraint), the deviation from the path is restricted to a small area in the Cartesian space thanks to the additional degree of freedom in path-following MPC. A more visible difference between the two experiments can be seen in the force-controlled subspace depicted in Figure 3.13. The disturbance leads to a significant increase in the contact force. When adding the force constraints this effect is lowered by 0.73 N, which corresponds to a reduction of 24.8 %, cf. Figure 3.13. However, as in the simulations, there exists a small violation of the constraints. This results from the model-plant mismatch, that is significantly increased during the disturbance and is unknown to the controller. Nevertheless, we could show that the proposed hybrid position and force controller can successfully be applied to simulation and experimental setups. It runs in real-time, achieves proper tracking and path-following performance, and allows to limit the occurring contact forces next to the satisfaction of input torque and state constraints.

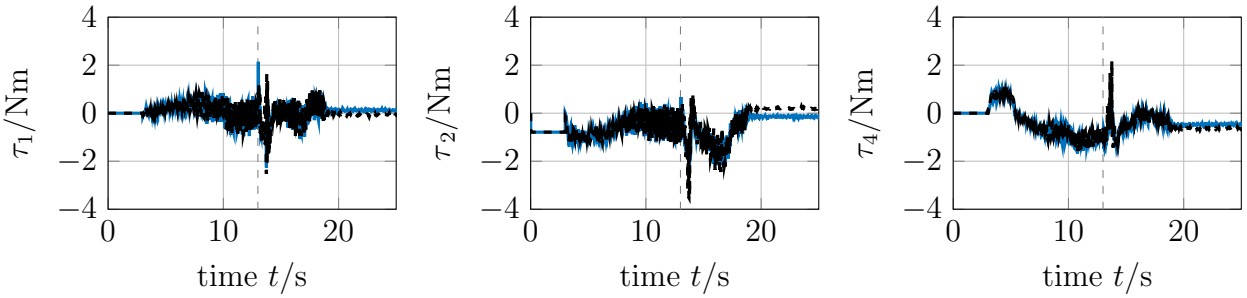


Figure 3.11: Control inputs for path following MPC in experiments with (black) and without (blue) active maximum force constraints.

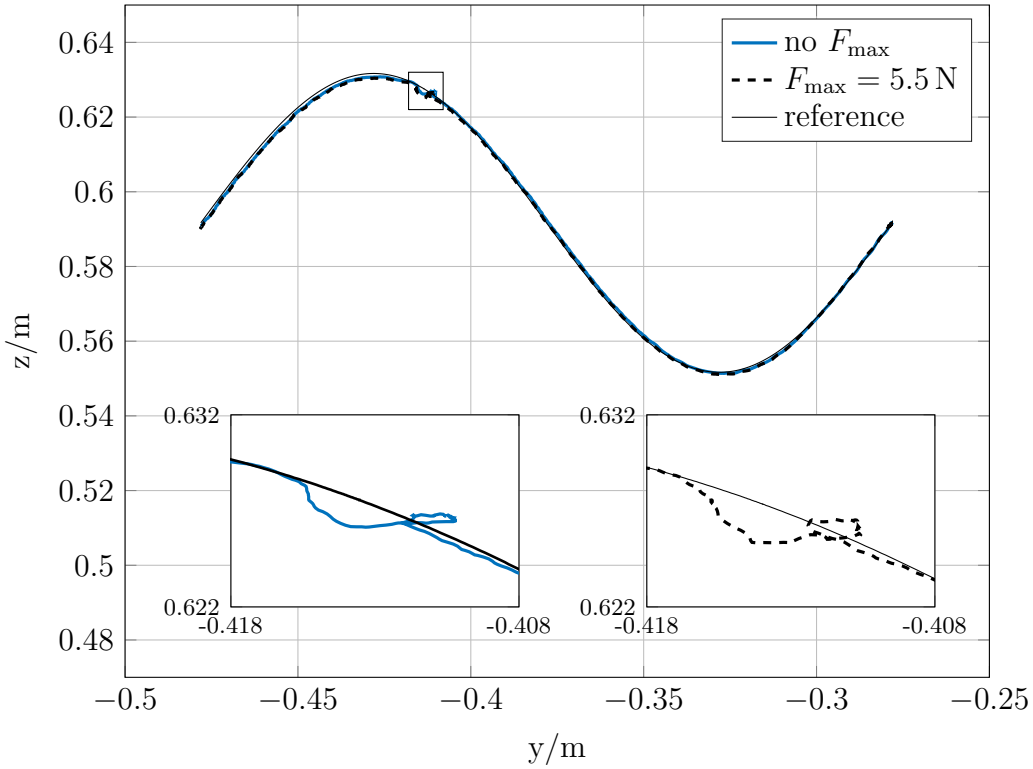


Figure 3.12: Cartesian end-effector position for disturbed path following MPC in experiments with (dashed) and without (solid blue) active maximum force constraint.

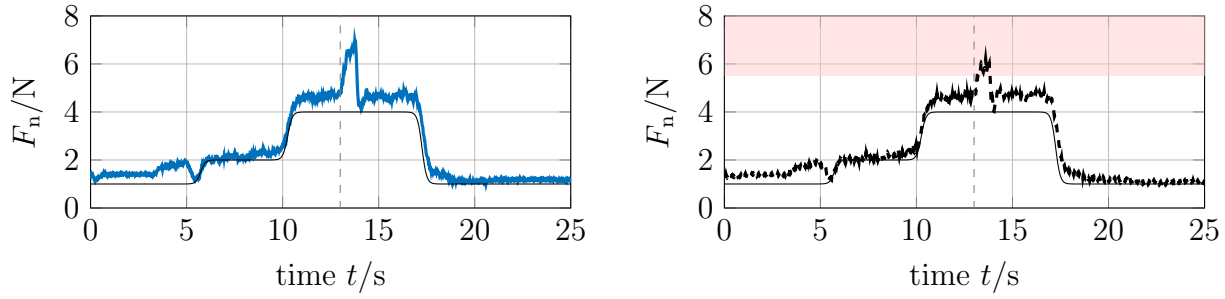


Figure 3.13: Normal contact force for disturbed path following MPC in experiments with (dashed black, right) and without (blue, left) active maximum force constraint.

3.4 Summary

In this chapter, a new direct force controller based on MPC has been proposed. The concept makes the benefits of MPC accessible for interaction tasks in robotics. Among others, we achieved the explicit consideration of constraints, the reduced tendency of oscillations and overshooting thanks to prediction, and the applicability to nonlinear MIMO systems. As an application example, a robot writes on a compliant surface in simulations and experiments. We achieved real-time-feasible optimal control for nonlinear prediction models in these implementations. In the nominal case, we guarantee closed-loop stability in robotic interaction tasks. At the same time, the experimental validation verified additional inherent robustness of the proposed approach concerning sensor noise, disturbances, and modelling inaccuracies. The implementations show that the proposed direct force control via MPC entails a considerable potential for robotic applications. It can increase the flexibility in robotic manufacturing since the precise positioning of the manipulated objects is not required anymore. At the same time, fragile objects can be handled while guaranteeing force limitation such that new fields of applications for robots are possible. Moreover, safety for human-robot interactions is enhanced by the explicit consideration of contact forces allowing for closer integration of robotic and human co-workers.

All in all, the proposed approach for direct force control with MPC allows for increased safety and autonomy of dynamical systems that interact with the environment.

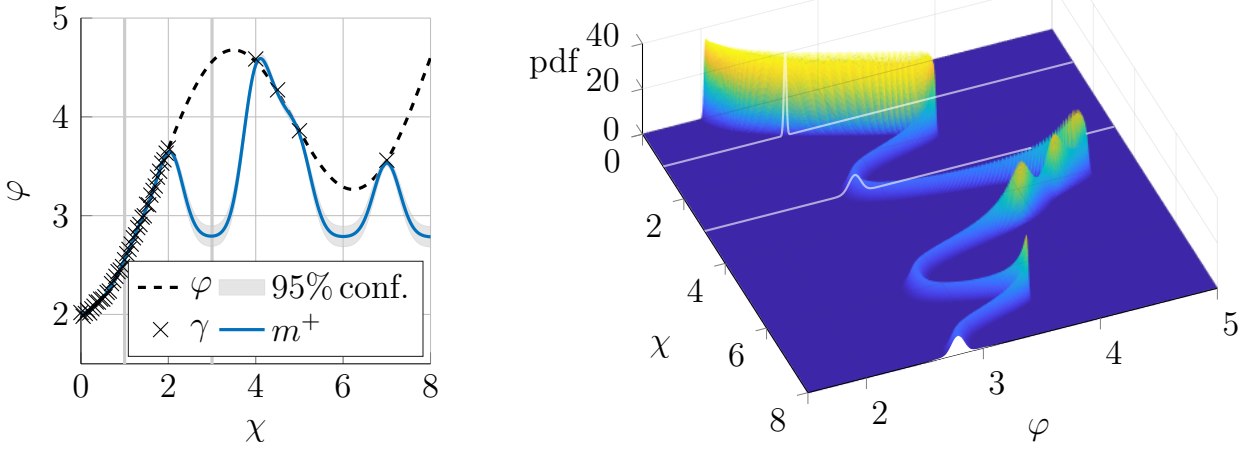
4 Gaussian Processes

In the previous chapters, knowledge of all components in the predictive controller, such as system dynamics, references, and possible disturbances, is assumed. This assumption is often not fulfilled, especially when operating a system in dynamical or changing environments. In such cases, machine learning can support the predictive controller and provide information about the system and its environment based on data. One specific machine learning approach, which is frequently used to do so, are Gaussian processes (GPs).

This chapter provides a general introduction to Gaussian processes. We will use GPs as elements to extend the first-principles based controllers to learning-supported MPC. For this purpose, we outline the basic concepts of data-based learning and regression using Gaussian processes, tailored for use in feedback systems. Loosely speaking, GPs calculate conditional probability distributions at the points of interest, for instance to obtain predictions of states, disturbances, or other model components, while taking data and the corresponding measurement noise into account. This allows us to obtain a prediction via the conditional mean and a reliability measure of these predictions via the associated variances. We outline the influence of prior knowledge, design decisions, and data on a trained Gaussian process. Furthermore, the control-relevant properties of Gaussian process models, such as differentiability and predictive capabilities, are discussed. In the following, we assume that the reader is familiar with basics in stochastics [213, 235].

4.1 Introduction to Gaussian Processes

Gaussian processes are a data-based, stochastic machine learning approach. GPs can be used for system identification, regression, as well as disturbance prediction, and have gained increasing attention in the control community for the modelling of static and dynamic systems, cf. [13, 16, 114, 119, 150, 157, 158, 180, 235]. This popularity arises from several appealing properties of GPs compared to first-principle models and other machine learning concepts such as neural networks: One benefit of Gaussian processes is the natural handling of uncertainty in the data. Noise in observations is explicitly considered in GP formulations. Moreover, GPs allow expressing uncertainty in models and predictions via distributions and confidence intervals. So to say, Gaussian processes provide an assessment of their approximation and prediction quality while avoiding overfitting noisy data. An additional advantage of GPs is that they allow for incorporating prior knowledge. This allows, if done suitably, that the involved



(a) Prediction with a GP trained on data (×) which should model the unknown function φ (black, dashed). The predicted mean m^+ is depicted in blue solid line and a confidence interval around it is shown in grey.

(b) Corresponding probability density function (pdf) of the GP. Areas with many data points (e.g. $\chi < 2$) show higher probability density then regions with few data (e.g. $2 < \chi < 4$). Each cutting plane parallel to the φ axis shows a Gaussian distribution (cf. white lines at $\chi = 1$ and $\chi = 3$).

Figure 4.1: Illustration of a Gaussian process.

parameters and structural assumptions are physically interpretable and enables the fusion of first-principles modelling and machine learning. For example, prior covariance functions can be chosen to encode smoothness or periodicity of the system and signals to be learned. Furthermore, prior mean functions can be built using first-principle models. For example, the prior mean can be represented by a linear first-principle model, while the GP is used to capture the underlying system's nonlinearities.

Let us consider the following static system

$$\gamma = \varphi(\chi) + \eta \tag{4.1}$$

mapping from the input $\chi \in \mathbb{R}^{n_\chi}$ to the one-dimensional output $\gamma \in \mathbb{R}$, where this is corrupted by the uncertainty or noise η. This noise is assumed to follow an independent, identically distributed Gaussian distribution $\eta \sim \mathcal{N}(0, \sigma^2)$ with zero mean and variance σ^2. The underlying functional relationship between the independent variables or input data χ and dependent variables or output data γ is represented by the mapping $\varphi : \mathbb{R}^{n_\chi} \to \mathbb{R}$.

We use Gaussian processes to model the unknown or only partially known underlying function φ in (4.1), cf. Figure 4.1. In other words, we consider regression problems to find the relationship between the input and output data. This relation is used to predict or infer outputs for previously unseen input data. GPs allow to estimate not only a deterministic value for the input but provide a Gaussian probability distribution

over all possible values of the output. A formal definition of GPs is [235]:

Definition 3. *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

Therefore, Gaussian processes can be interpreted as an extension from normally distributed variables to distributions over functions, see also Figure 4.1. They are characterised by a mean function $m : \mathbb{R}^{n_\chi} \to \mathbb{R}$ and a covariance function $\kappa : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_\chi} \to \mathbb{R}$ and we will shortly denote them by

$$\varphi(\chi) \sim \mathcal{GP}\left(m(\chi), \kappa\left(\chi, \chi'\right)\right). \tag{4.2}$$

Hereby, the covariance function κ is a measure of the joint variability of two random variables. It is positive semi-definite and symmetric $\kappa(\chi, \chi') = \kappa(\chi', \chi)$.

Our overall goal is to predict or infer the distribution of the output $\varphi(\chi_*)$ at (possibly unseen) points χ_*. Formulating a GP model to make those predictions involves the following steps: First, one needs to choose the structure of the GP. This involves the formulation of the prior belief about the mean and covariance functions m and κ. Second, a collection of input-output-observation should be selected to build data sets D_ϕ and D_{pred} used for the training and prediction or estimation as outlined later. Here, the index ϕ stands for the hyperparameters, which result from the training with the data set D_ϕ. Whether these data sets are fixed or adapted once new data becomes available distinguishes between offline and online learning of GPs. Third, the GP trains or learns based on the data set D_ϕ via optimising the so-called hyperparameters ϕ of the mean and covariance function. Finally, inference or prediction via the GP utilises the structure, the prior, the determined hyperparameters, and the data D_{pred}.

These four topics, including structure, data, hyperparameters, and predictions, are outlined in the following sections.

4.2 GP System Structure and Prior Belief

In the design of a GP to represent a (dynamic) system, the model structure, the regressors or inputs, the prior mean function, and the prior covariance functions must be chosen. In this step, knowledge about the underlying structure of the system can be incorporated if available. This possibility of including prior knowledge discriminates GPs from pure data-driven black-box modelling. Even with no or only few a priori information, GPs can produce accurate inference and function approximation performance. Several model structures can be used depending on whether a static or a dynamic mapping should be modelled via the GP. As GPs are widespread in data analysis, input-output-data-based formulations of dynamic systems like autoregressive models are common, see [117, 149]. Furthermore, state-space models learned via Gaussian processes can be considered as in [184, 223, 226]. We use GPs to learn and

model external signals such as references (Section 5.4) and output maps of dynamical systems (Section 5.3). In both cases, the GP reflects a static mapping such that neither delayed input regressors, as in autoregressive models, nor recursive predictions based on previous GP evaluations must be performed, see also Section 4.6.

Additionally to selecting a model structure, the choice of prior mean and covariance functions m and κ allows us to reflect assumptions and knowledge over the modelled system (4.1). If knowledge of the underlying system is available, inclusion via the prior allows for improved inter- and extrapolation quality. Two specific types of covariance functions will be considered in the following chapters which are:

Definition 4. *A covariance function $\kappa : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_\chi} \to \mathbb{R}$ is called periodic if there exists a period $\Delta\chi \in \mathbb{R}^{n_\chi}$ such that $\kappa(\chi, \chi') = \kappa(\chi, \chi' + i\Delta\chi)$ with $i \in \mathbb{N}_0$.*

Definition 5. *A covariance function $\kappa : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_\chi} \to \mathbb{R}$ is called a decaying covariance function if it fulfils:*

(i) The covariance function κ is stationary and monotonously decreasing $\kappa(\chi_1, \chi') \leq \kappa(\chi_2, \chi')$ for all $|\chi_1 - \chi'| \geq |\chi_2 - \chi'|$.

(ii) There exists a finite bound $\varsigma : \mathbb{R}^{n_\phi} \to \mathbb{R}_0^+$ for which the absolute values of the time derivatives $\kappa^{(\beta)} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, \kappa^{(\beta)} = \dfrac{\partial^\beta \kappa(\cdot, \chi')}{\partial \chi^\beta}$ up to order β of the covariance function κ are monotonously decreasing such that $|\kappa^{(\beta)}(\chi_1, \chi')| \leq |\kappa^{(\beta)}(\chi_2, \chi')|$ at least for all $|\chi_1 - \chi'| \geq |\chi_2 - \chi'| \geq \varsigma(\phi)$.

For instance, the so called squared exponential covariance function

$$\kappa_{\mathrm{se}}(\chi, \chi') = \sigma_{\mathrm{se}}^2 \exp\left(-\frac{(\chi - \chi')^2}{2l_{\mathrm{se}}^2}\right)$$

with hyperparameters $\phi = (\sigma_{\mathrm{se}}, l_{\mathrm{se}})$ is a decaying covariance function for order $\beta = 2$ with $\varsigma(\phi) = \sqrt{3} l_{\mathrm{se}}$, which is illustrated in Figure 4.2.

The prior mean and prior covariance functions involve hyperparameters $\phi \in \mathbb{R}^{n_\phi}$, where n_ϕ depends on the selected functions m and κ. The hyperparameter values can be learned based on a hyperparameter training set, as outlined in Section 4.4. The selection of the involved data sets is discussed in the following section.

4.3 Data Collection

The number and location of the data points influence the approximation capability and the computational complexity of the GPs. While large training data sets can improve the approximation quality, they increase the computational load of the inference or prediction. Therefore, data pre-processing to obtain useful training data sets is an essential step in GP modelling and design, just as in any other machine learning

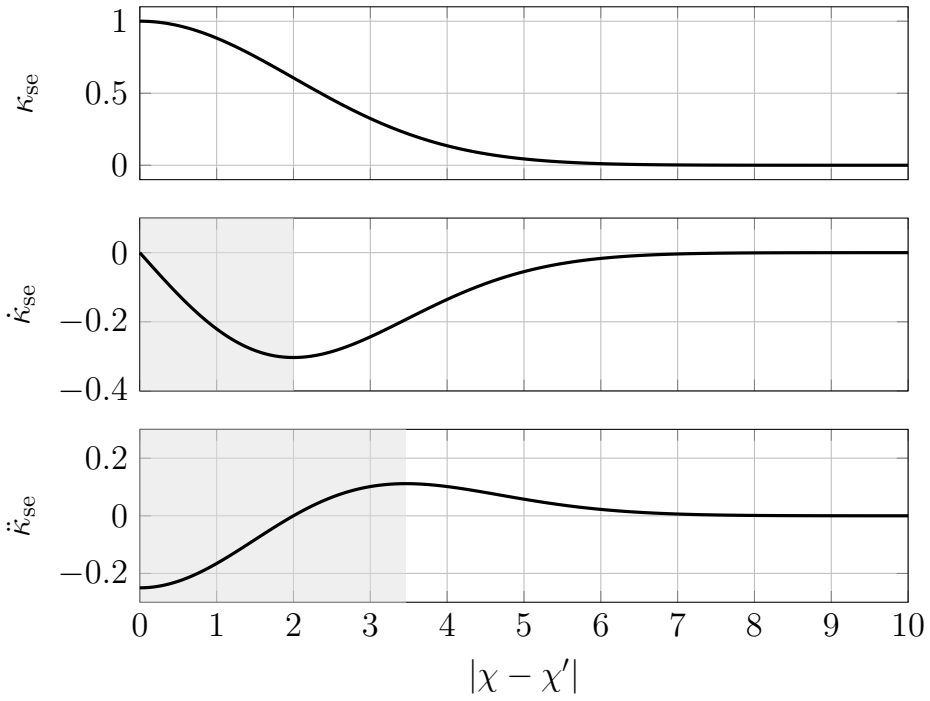


Figure 4.2: Squared exponential covariance function and its derivatives with hyperparameters $\sigma_{\text{se}} = 1$ and $l_{\text{se}} = 2$. The covariance is stationary and strictly monotonously decreasing for all $|\chi - \chi'|$, while the absolute values of its first and second derivative are strictly monotonously decreasing for $|\chi - \chi'| > l_{\text{se}} = 2$ and $|\chi - \chi'| > \sqrt{3} l_{\text{se}} \approx 3.46$, respectively.

algorithm. A tradeoff between information quality and computational complexity must be found, such that a reduced amount of data with high informative content is obtained for training. While a detailed discussion on data selection, pre-processing, and the curse of dimensionality is beyond the scope of this work, interesting approaches using Gaussian processes are sparse GPs [34], latent variable GPs [221] and GP autoencoder [105]. Moreover, data selection for dynamical systems' modelling via GPs in multiple tasks was recently addressed in [31]. An analysis of the effect of the data distribution on controller performance was performed in [127], while [225] identifies the number of training points to learn asymptotically stable GP state-space models.

In general we distinguish two types of training data sets in GPs: The hyperparameter training set D_ϕ, and the training data set used for inference or predictions D_{pred}. They are defined via

$$D_\phi := \left\{ (\chi_{\phi,i}, \gamma_{\phi,i}) \in \mathbb{R}^{n_\chi} \times \mathbb{R} \,|\, i = 1, 2, \ldots, n_{D_\phi} \right\} \tag{4.3}$$

and

$$D_{\text{pred}} := \left\{ (\chi_{\text{pred},i}, \gamma_{\text{pred},i}) \in \mathbb{R}^{n_\chi} \times \mathbb{R} \,|\, i = 1, 2, \ldots, n_{D_{\text{pred}}} \right\}, \tag{4.4}$$

where n_{D_ϕ} and $n_{D_{\text{pred}}}$ are the number of data points for hyperparameter estimation

and prediction/inference, respectively. For ease of notation, we define

$$\boldsymbol{\chi}_\phi := (\chi_{\phi,1}, \ldots, \chi_{\phi,n_{D_\phi}}), \tag{4.5}$$

$$\boldsymbol{\gamma}_\phi := (\gamma_{\phi,1}, \ldots, \gamma_{\phi,n_{D_\phi}}), \tag{4.6}$$

$$\boldsymbol{\chi} := (\chi_{\text{pred},1}, \ldots, \chi_{\text{pred},n_{D_{\text{pred}}}}), \tag{4.7}$$

$$\boldsymbol{\gamma} := (\gamma_{\text{pred},1}, \ldots, \gamma_{\text{pred},n_{D_{\text{pred}}}}), \tag{4.8}$$

such that $D_\phi = (\boldsymbol{\chi}_\phi^\top, \boldsymbol{\gamma}_\phi^\top)$ and $D_{\text{pred}} = (\boldsymbol{\chi}^\top, \boldsymbol{\gamma}^\top)$. Especially in offline learning, these two data sets are often assumed to be identical. However, different data sets for prediction and hyperparameter optimisation can be used in both offline and online learning. For example, an online update of the prediction training data set D_{pred} can be performed separately from an offline determination of the hyperparameters via D_ϕ as was done, e.g., in [157].

4.4 Hyperparameter Determination

The mean and covariance functions involve hyperparameters $\phi \in \mathbb{R}^{n_\phi}$. For example, if we have a constant mean function $m(\chi) = \mu$ and a squared exponential covariance function $\kappa_{\text{se}}(\chi, \chi') = \sigma_{\text{se}}^2 \cdot \exp\left(-\frac{(\chi-\chi')^2}{2l_{\text{se}}^2}\right)$ we have in the scalar case in total three hyperparameters $\phi = (\mu, \sigma_{\text{se}}, l_{\text{se}})$, which are the constant value of the mean μ, a vertical scaling σ_{se}, and a horizontal length scale parameter l_{se}. Often also the measurement noise standard deviation σ is considered as one of the hyperparameters. Note that, given the data $\boldsymbol{\chi}_\phi, \boldsymbol{\gamma}_\phi$ and the functions κ and m, the hyperparameters θ will be uncertain and posses distributions. However, their distribution can be learned based on the hyperparameter training set D_ϕ. Using Bayes rule the posterior probability distribution $p(\phi|\boldsymbol{\chi}_\phi, \boldsymbol{\gamma}_\phi)$ of the hyperparameters can be inferred by

$$p(\phi|\boldsymbol{\chi}_\phi, \boldsymbol{\gamma}_\phi) = \frac{p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi, \phi)p(\phi)}{p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi)}. \tag{4.9}$$

Here, $p(\phi)$ is the prior belief about the hyperparameter distribution, $p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi, \phi)$ is the evidence or marginal likelihood, and $p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi)$ is a normalising constant. Unfortunately, analytic calculation of $p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi)$ is in most cases intractable [117]. Numerical approximation methods as Monte Carlo simulations can be used to approximate the distribution, which are computationally expensive. Therefore, often a point estimate $\hat{\phi}$ instead of a full distribution is used. This estimate can, for example, be obtained by maximising the logarithmic marginal likelihood. Under the assumption of a uniformly prior hyperparameter distribution, the hyperparameter posterior distribution is proportional to the marginal likelihood [117, 235]. Thus, maximising the logarithmic marginal likelihood corresponds to finding the most likely hyperparameters $\hat{\phi}$. Assum-

ing a deterministic constant prior mean with unknown value μ and let $\boldsymbol{\mu}$ be a vector with appropriate length composed of μ, the logarithm of the marginal likelihood is given by

$$\begin{aligned}\ln p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi,\phi) = &-\frac{1}{2}\ln(|(K_\phi+\sigma^2 I)|) - \frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\gamma}_\phi)^\top (K_\phi+\sigma^2 I)^{-1}(\boldsymbol{\mu}-\boldsymbol{\gamma}_\phi)\\ &-\frac{n_{D_\phi}}{2}\ln(2\pi).\end{aligned} \tag{4.10}$$

Here, K_ϕ is the covariance matrix of the training data with size $n_{D_\phi}\times n_{D_\phi}$ whose entries are calculated based on the covariance function κ. Specifically, $K_\phi^{i,j} = \kappa(\chi_{\phi,i},\chi_{\phi,j})$ with $i,j \in \{1,\dots,n_{D_\phi}\}$. While the first term in (4.10) penalises the complexity of the model and thereby reduces overfitting, the second term fits the model to the data. The third term is a normalisation constant. By maximising (4.10) the most likely hyperparameters $\hat{\phi}$ are obtained by finding a tradeoff between model complexity and data consistency. The hyperparameter optimisation can, thus, be written as

$$\hat{\phi} := \arg\max_{\phi} \ln p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi,\phi) \tag{4.11}$$

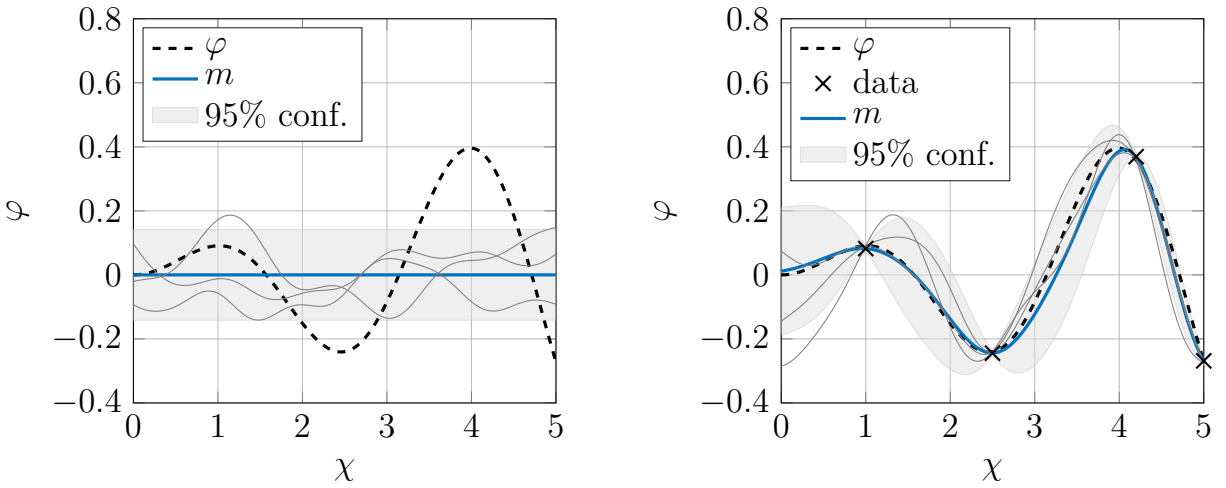
with the logarithmic marginal likelihood $\ln p(\boldsymbol{\gamma}_\phi|\boldsymbol{\chi}_\phi,\phi)$ as objective function. The evaluation of the objective function includes the inversion of K_ϕ, which relates to a computational complexity of $\mathcal{O}(n_{D_\phi}^3)$. In Chapter 5, we will derive extensions to this way of hyperparameter determination such that additional constraints in the optimisation can be considered. Though the addition of constraints in general increases the computational complexity of the optimization, often hyperparameters are determined offline or at certain events and not at every iteration of a control loop [157, 224].

4.5 Inference and Prediction

The overall goal in supervised learning with GPs is to obtain a model of (4.1) such that we can infer or predict the distribution of its output at possibly unseen points χ_*. This prediction is based on the prior belief, the training data D_{pred}, and the hyperparameters ϕ (or their estimates $\hat{\phi}$) as introduced in the previous sections. The joint distribution of the noisy training data output $\boldsymbol{\gamma}$ and the desired predicted output φ_* which estimates $\varphi(\chi_*)$ is given by

$$\begin{pmatrix}\boldsymbol{\gamma}\\ \varphi_*\end{pmatrix} \sim \mathcal{GP}\left(\begin{pmatrix}\boldsymbol{m}(\boldsymbol{\chi})\\ m(\chi_*)\end{pmatrix}, \begin{pmatrix}K+\sigma^2 I & \mathrm{k}\\ \mathrm{k}^\top & \kappa(\chi_*,\chi_*)\end{pmatrix}\right),$$

where $\boldsymbol{m}(\boldsymbol{\chi}) := (m(\chi_{\text{pred},1}),\dots,m(\chi_{\text{pred},n_{D_{\text{pred}}}}))^\top$. The entries of the covariance matrix K are calculated using the covariance function κ such that K is of dimension $n_{D_{\text{pred}}} \times n_{D_{\text{pred}}}$ and specifies the covariance between all of the training data points. The entries of the vector k (with dimension $n_{D_{\text{pred}}}\times 1$) are calculated with the covariance function



(a) Prior GP with zero prior mean (solid blue) and 95% confidence interval (grey area) representing $\pm 2\sqrt{\kappa^+(\chi,\chi)}$ with constant hyperparameters. From the distribution three sample functions are drawn in solid, thin lines. The function we want to learn is φ depicted as dashed line.

(b) Conditioned on data D_{pred} (black crosses) the posterior mean function m^+ (blue) and the 95% confidence interval $m^+ \pm 2\sqrt{\kappa^+(\chi,\chi)}$ (grey area) are inferred. Random sample functions (grey lines) drawn from the posterior show conditioning on the data.

Figure 4.3: Inference or prediction with a GP.

using test and training data points such that $\mathbf{k}^i = \kappa(\chi_i, \chi_*)$ with $i \in \{1, \ldots, n_{D_{\text{pred}}}\}$. It defines the cross correlation between test and training data points. The scalar $\kappa(\chi_*, \chi_*)$ is the auto covariance of the test data. Given this joint probability distribution, the conditional posterior distribution of φ_* given the data $D_{\text{pred}} = \{\boldsymbol{\chi}, \boldsymbol{\gamma}\}$ is

$$p(\varphi_* | \boldsymbol{\chi}, \boldsymbol{\gamma}) = \mathcal{N}(m^+(\chi_*), \kappa^+(\chi_*)), \tag{4.12}$$

where the posterior mean function $m^+ : \mathbb{R}^{n_\chi} \to \mathbb{R}$ is defined by

$$m^+(\chi_*) := m(\chi_*) + \mathbf{k}^\top (K + \sigma^2 I)^{-1} \big(\boldsymbol{\gamma} - \boldsymbol{m}(\boldsymbol{\chi})\big) \tag{4.13}$$

and the posterior covariance $\kappa^+ : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_\chi} \to \mathbb{R}_0^+$ is given by

$$\kappa^+(\chi_*, \chi_*) := \kappa(\chi_*, \chi_*) - \mathbf{k}^\top \big(K + \sigma^2 I\big)^{-1} \mathbf{k}.$$

Please note that inherently these posteriors depend on the hyperparameters ϕ via the involved prior covariance and mean as well as on the involved data D_{pred}. Whenever beneficial, we will denote this dependency by $m^+(\chi_*; D_{\text{pred}}, \phi)$ and $\kappa^+(\chi_*, \chi_*; D_{\text{pred}}, \phi)$. However, in most of the cases this notation is omitted for brevity. An illustration of GP inference given the hyperparameters is displayed in Figure 4.3. The computational complexity of inference is mainly influenced by the inversion of K, such that is scales with $\mathcal{O}(n_{D_{\text{pred}}})$.

4.6 Gaussian Processes and Control

When using Gaussian processes in a control context, some properties of them need special consideration or can be of particular use. Three of those, namely the prediction for multiple steps into the future, the prediction of multi-dimensional outputs, and the derivatives of GPs, will be briefly discussed in the following.

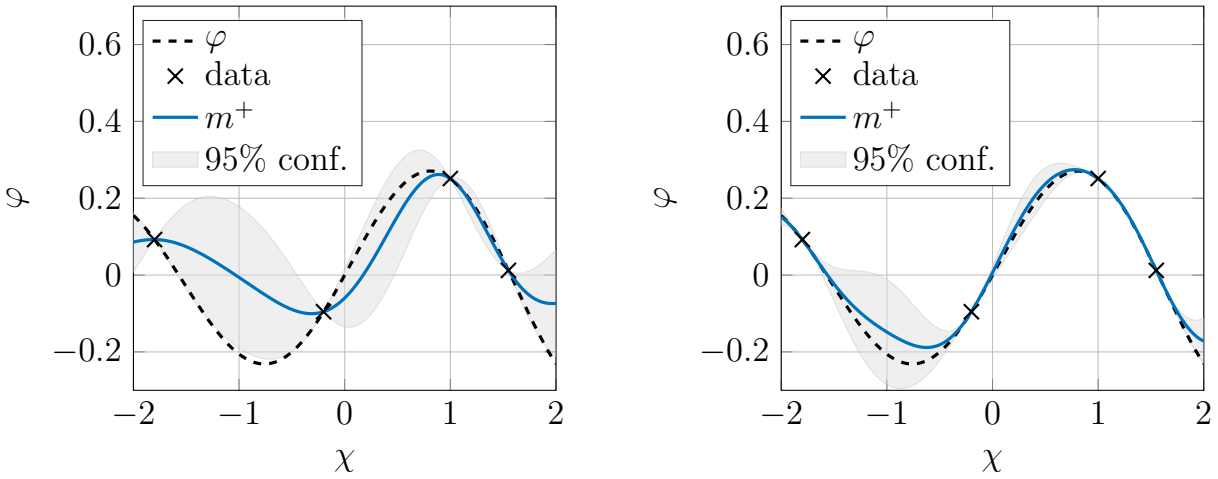
Multi Step Ahead Predictions

When using Gaussian processes in a model predictive control framework, they are often utilised to model (parts of) the system dynamics, see e.g. [29, 93, 94, 118–120, 135, 149, 171, 178, 180, 223, 225, 237]. This requires a recursive evaluation of the GP to obtain multiple-step-ahead predictions that cover the prediction horizon. This recursive evaluation requires a GP prediction where the input to the GP is the predicted output distribution of that GP from a previous time step. This input has a Gaussian distribution and is not a deterministic value. Calculating a posterior distribution from uncertain inputs is in general intractable, and the posterior distribution then is non-Gaussian [93]. Nevertheless, there are approximation techniques to overcome this issue. One straightforward strategy is to use only the predicted mean value as a deterministic input. It was shown in [79, 94] that this leads to a poor estimation of the actual uncertainty of the predictions. Numerical Monte Carlo simulations can approximate the actual distribution, which is associated with a high computational load. Alternatively, analytical approximations as exact moment matching and Taylor series approximations can be used to obtain Gaussian distributions, which approximate the actual non-Gaussian distribution, see [78, 93]. Additionally, a recursive evaluation of a GP to obtain multi-step ahead predictions is often based on an independence assumption that ignores the recursive nature. This assumption can also lead to an underestimation of the real uncertainty in the predictions. To overcome this issue, in [92] correlation between iterations is included to obtain more realistic predictions and uncertainty estimates.

In this work, we do not use GPs to model dynamical system mappings. Instead, we use Gaussian processes to model static mappings, which describe either the reference or the system outputs. This way, no recursive GP calls are necessary, which eliminates the aforementioned issues in our context.

Multiple Outputs

In this chapter, we assume the map φ in (4.1) maps into a one-dimensional space. As we use GPs, among others, as reference predictors, a one dimensional GP could model the reference of a SISO system under output feedback. However, in general, we are interested in multi-dimensional predictions via GPs, such as modelling a reference in a higher dimensional state space. In the following chapters, we will train multiple



(a) GP posterior without derivative information at training data points.

(b) GP posterior with derivative information at training data points.

Figure 4.4: GP posterior with and without derivative observations.

GPs to cope with those higher dimensional mappings, also known as multi-kriging [23]. The covariance between the individual processes is not considered during prediction. This approach possesses an essential advantage over fully coupled GPs, as the implementational cost is significantly lower. This results from the inversion of the covariance matrix K, which causes a computational load that is cubic in its size, i.e. in the number of the training data points. For n_y uncoupled GPs with n_{D_pred} data points, the computational complexity of prediction scales as $\mathcal{O}\left(n_\mathrm{y} \cdot n_{D_\mathrm{pred}}^3\right)$, whereas coupled GPs have $\mathcal{O}((n_\mathrm{y} n_{D_\mathrm{pred}})^3)$. However, extensions to include dependencies, which might occur due to correlated measurement noise or correlations in the system itself, exist, see [23, 32, 235]. In geostatistics, the field of correlated multi-dimensional GPs is known as co-kriging [33]. Correlations between the multiple outputs can also appear in the form of constraints, such as linear and quadratic dependencies, see [106, 201, 202]. In this work, constraints are included in the learning phase to express correlations between multiple GPs. Coupled higher dimensional GPs with decoupled covariances are obtained. This setup increases the computational complexity of the hyperparameter optimisation while lowering the computational demand in the predictions. For offline learning with online data updates, this shifts the computational burden from online to offline calculations. Hence, real-time-feasible implementations are enabled by efficient usage of available resources.

Derivatives of Gaussian Processes

When using differentiable covariance functions, such as the squared exponential covariance, periodic covariances, or specific parametrisation of the Matérn covariance [235], derivatives of Gaussian processes can be obtained. As differentiation is a linear operation, the derivative of a GP is also a GP [214]. This property brings two ad-

vantages: On one side, derivative information can be included in the learning phase. This additional information is beneficial as it allows for better predictions with a smaller number of data points, see also Figure 4.4. On the other side, we can predict derivatives that might be desired in (dynamical) systems modelling. The posterior distribution of the derivative of GP (4.2) with respect to the a^{th} component of χ_* is defined via the derivative of the posterior mean

$$\left.\frac{\partial m^+(\cdot)}{\partial \chi^a}\right|_{\chi_*} := \left.\frac{\partial m(\cdot)}{\partial \chi^a}\right|_{\chi_*} + \left.\frac{\partial \mathbf{k}^\top(\chi,\cdot)}{\partial \chi^a}\right|_{\chi_*} \left(K+\sigma^2 I\right)^{-1}\left(\boldsymbol{\gamma}-\boldsymbol{m}(\boldsymbol{\chi})\right) \tag{4.14}$$

and the derivative of the posterior covariance

$$\left.\frac{\partial \kappa^+(\chi_*,\cdot)}{\partial \chi^a}\right|_{\chi_*} := \left.\frac{\partial^2 \kappa(\chi_*,\cdot)}{\partial \chi^a \partial \chi^a}\right|_{\chi_*} - \left.\frac{\partial \mathbf{k}^\top(\chi_*,\cdot)}{\partial \chi^a}\right|_{\chi_*} \left(K+\sigma^2 I\right)^{-1} \left.\frac{\partial \mathbf{k}(\chi_*,\cdot)}{\partial \chi^a}\right|_{\chi_*}. \tag{4.15}$$

In a setup where GPs are used to model references for a controller, derivative predictions are useful when the controlled dynamic system includes e.g. integrator chains. In Section 6 equations (4.14) and (4.15) will be used to predict velocities in multiple coordinates which build the basis for the reference definition. The prediction of the derivative of the GP (4.2) adds no additional hyperparameters to be determined. Furthermore, it only includes observations of the function itself and not of its derivatives. However, if derivative observations are available, they can be included in the training data set, which leads to an augmented covariance matrix. This augmented covariance matrix takes the covariances between the original observations and the derivative observations as well as the autocorrelation of the derivative observations into account, see also [185]. Including derivative observations can highly increase the approximation capability. Thus, it can also reduce the number of training data points while maintaining the prediction quality. Moreover, instead of actual derivative measurements, monotonicity assumptions can be included in a similar way [192]. We exploit this fact in the constrained hyperparameter optimisation in Chapter 5.

4.7 Summary

This chapter gives a basic introduction to Gaussian processes and shows their applicability in a control context. While the posterior mean of a GP can be used for data-based prediction of static and dynamic systems, the posterior variance allows for evaluation of the prediction model quality, which can be useful both in offline modelling and online control. These characteristics will be used in the following chapters to support model predictive controllers with learning-based GP models. This combination allows for explicit consideration of noise in measurements and the inclusion of prior knowledge in the training process.

5 Learning-supported Model Predictive Control

In this chapter, we merge the outlined MPC and GP concepts to obtain a model predictive control formulation that is supported by learning. In contrast to the nominal MPC cases of Chapter 2, we do not assume perfect knowledge of all components in the predictive controller. In particular, the reference and the output of the system are assumed to be unknown a priori. We will use Gaussian processes to obtain a representation of these model components from data and supply the predictive controller with this information. Based on the discussions and outlined properties of MPC and GPs from the previous chapters, we analyse the stability properties of the learning-supported controllers. Furthermore, application examples illustrate the proposed concepts in simulations and experiments. The robotic force control example from Chapter 3 is extended to include a learning- and data-based force model.

5.1 Introduction to Machine Learning and Control

Model predictive control is a popular control strategy due to its versatility, conceptual simplicity, and superior performance to linear or decoupled controllers in many cases. Most importantly, MPC can handle constraints directly during control, which makes it attractive for delicate control tasks. Additionally, a comprehensive theory exists, which allows us to give stability guarantees, see also Chapter 2. The nominal stability guarantees, however, often assume perfect knowledge about the system and its environment. Robust MPC strategies extend this scope to give guarantees despite uncertainty in the model and signals [121, 136, 162, 164, 167, 187, 189]. However, if the uncertainty is considerable, robust concepts become quite conservative or even infeasible. Moreover, the performance of nominal and robust MPC highly depends on its prediction quality. This includes the quality of the system model as well as knowledge about external signals as disturbances and references. Therefore, it is beneficial to minimise the uncertainty in the predictive control framework in nominal as well as in robust approaches. Machine learning methods are suited strategies to do so. Including information about the system and its environment in a learning-based fashion can lower the needed effort to obtain models for dynamics and external signals and allows to adapt the controller online. Consequently, we will use Gaussian processes to support model predictive controllers. However, an arbitrary update of the controller, e.g. via an update of the prediction model or the reference, can lead to a loss of guarantees as stability and does not guarantee performance improvement. Consequently, development of trustable strategies on how to incorporate learning in MPC is needed.

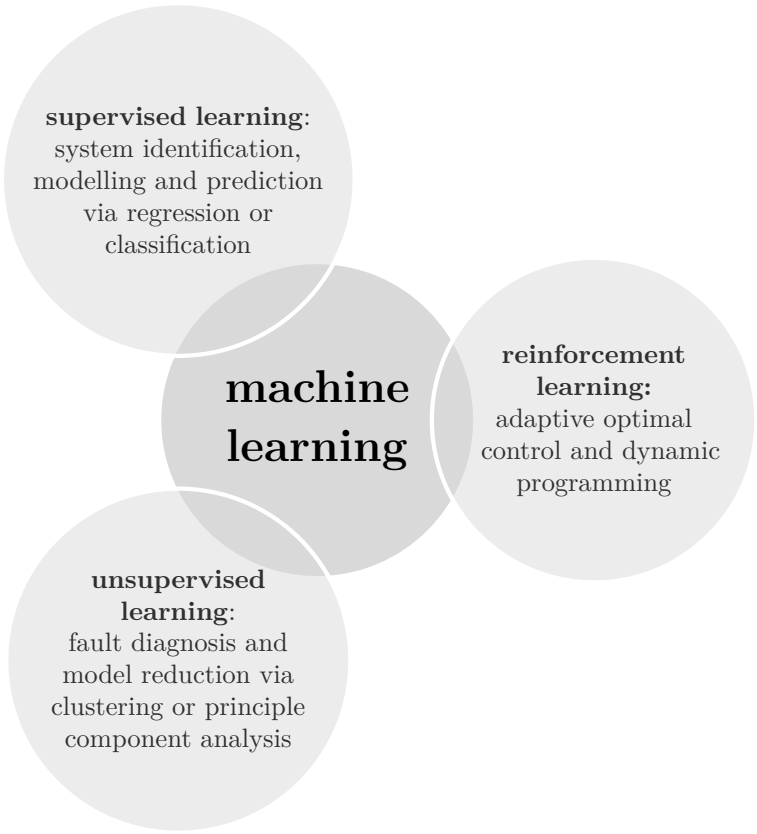


Figure 5.1: Classification of machine learning algorithms and their possible fields of application in control.

In the next sections, we address these demands by supporting MPC with learning of prediction models as well as by supporting and tailoring the learning of GPs by dynamic system properties. In order to better classify these concepts, we give a short overview of the combination of machine learning and control in the following.

5.1.1 Machine Learning for Control

Machine learning is a way to achieve artificial intelligence of computer systems where data is used to find coherence and generalise from it automatically. Often, machine learning algorithms are classified based on how the learning task is achieved, cf. Figure 5.1. In supervised learning, for each input data, the corresponding output data is given. The task is to find a mapping between the input and the output. Once this mapping is identified for the given training data, generalisation for unseen input data points is possible. Thus, supervised learning can be used for system identification and modelling. Gaussian-process-based learning is a supervised learning approach. In unsupervised learning, a given set of input data should be analysed. Often, similarity in the data should be found, which can be represented via clustering of the data. In that case, the learning task is not to find a relation between inputs and outputs but to

analyse and structure the data. Consequently, unsupervised learning is useful for fault detection via finding outliers in the data or model reduction via finding redundancies. Reinforcement learning represents a third category of learning, which is closer to the way humans or animals learn. The task is to find an optimal strategy of performing a quest by learning from the interaction with the system and its environment via trial and error. Input and output data becomes available with an assigned reward. Maximising this reward relates to learning the best strategy. Therefore, some reinforcement learning strategies resemble optimal control, adaptive control and iterative learning control, as they try to find an optimal way to operate systems.

In this work, supervised learning is used to support model predictive control using GPs. We do not consider model reduction, fault diagnosis, or learning a controller. Instead, we use supervised learning of Gaussian processes to obtain prediction models for the controller. Specifically, learning of static prediction models for reference, signals, and output maps is considered.

5.1.2 Learning of Different Components in MPC

Supervised learning can be used to learn various components in a predictive controller. In [29, 93, 94, 118–120, 135, 171, 178, 180, 223, 237] the dynamical system is learned partially or completely, while [196–199] use supervised learning to iteratively update the terminal cost and constraint. When disturbance models are learned and used in a robust predictive controller such as tube MPC, they can also be used to adjust the tightening of constraints. This way, an additive part in the system model (the disturbance) and the constraints are adjusted simultaneously via learning, see, for example, [231]. Similar connections exist between the learning of the system model and the adaptation of constraints in robust and stochastic control, cf. [93, 135, 215, 232].

Specifically, MPC with GP system models allows quantifying the uncertainty or trustability of the model predictions via the variance. This measure can be incorporated as hard or soft constraint to avoid regions of the state space for which only limited information or measurements are available. Consequently, the constraints or the cost function of the predictive controller are learned, see [10, 118, 171].

The focus of this chapter lies on learning external signals, such as references and disturbances, as well as static output maps of the prediction model. References and disturbances are two particular types of external signals, which we want to model, learn, and predict. They differ with respect to the control perspective we embed them in. While references should be tracked or followed by the system, disturbances should be rejected. In the following, a short overview of related works that focus on the learning of external signals is given. Learning a dynamical system that serves as a reference prediction model is done in [143] for platooning, where the velocity of the leading vehicle is learned and used as a reference for the following vehicle. A periodic reference is learned via Gaussian processes in [114], and the approach is applied to

the precise positioning of telescopes. A similar idea is used in [177] for blood glucose control. However, the learned signal is interpreted as a disturbance. A dynamic disturbance model to predict demands in water networks is considered in [231, 233] including GPs in robust MPC.

Besides uncertainty in external signals, uncertainties can affect parts of the prediction models. In particular, the output map (2.1b) might not be perfectly known while the system dynamics (2.1a) are well understood. In such a case, the output map can be learned separately from the dynamics. For example, kinematics in flexible or cable-driven robots are hard to model based on first principles but can be learned via Gaussian processes, cf. [55, 148]. Moreover, contact with objects might be hard to model based on first principles due to unknown material properties or positions. In [160], learning-supported MPC is designed based on a Gaussian process that models the contact force as system output in a robotic system. Here, the system dynamics of the robot are known, while external contact forces, which depend statically on states, are learned.

5.1.3 Online, Offline, and Iterative Learning

Nominal MPC strategies, as outlined in Chapter 2, rely on a priori knowledge about the system and the control task. This knowledge takes the form of the system model, the cost function, the constraints, or the reference. These components are traditionally designed based on first-principles models and are physically interpretable. However, machine learning can be used to obtain them, as outlined in the previous subsection. If this learning task is performed separately from the controller execution, it refers to *offline learning*. Offline learning entails the benefit that the computational complexity needed to obtain a learned model is detached from real-time demands of the closed-loop controller. Furthermore, it allows for the evaluation of model accuracy before controller execution, such that approximation errors can be estimated and bounded. These benefits are used, for example, in [29, 117, 119, 120, 135] where offline learning with Gaussian processes for MPC is considered.

When the same or a similar task is performed repeatedly, machine learning can be used to incorporate knowledge from previous executions. In between each execution machine learning based on the gathered data can be used to adapt learning-supported controllers. We refer to this type of learning as *iterative learning* while being aware that classically iterative learning control (ILC) is not restricted to the use of machine learning algorithms and often directly updates the control input instead of components in a predictive controller. In contrast, the works [178, 180] update the prediction model in an MPC based on data from a previous execution of a repetitive task. The concept is applied to an autonomous vehicle on a particular route. The GP models a part of the system dynamics, which includes the effect of the environment. As in offline learning, the computational demand of the learning is detached from online execution

times. The works [196–199] use iterative learning to update the terminal cost and constraint after each batch execution. They use a reference-free MPC formulation, i.e., they consider cases where the desired reference to be followed is unknown. Instead of learning the reference, learning of a safe set from state trajectories of previous executions is performed, which builds the basis for the terminal constraint set. The reference for a model predictive controller is updated via iterative learning in [236]. In contrast, iterative learning of a cost function is proposed in [219]. In [208], a disturbance is learned from previous executions. This learned disturbance is used in the system model of the MPC at each consecutive iteration. Moreover, it iteratively updates a reference generator in between the control executions.

Online learning builds a third possibility to combine machine learning and control, where the controller is updated during runtime. Online learning-based control is, therefore, closely related to adaptive control techniques. In online machine learning-supported MPC, an update of the system model or other components in the optimal control problem during runtime is performed. This requires special care to maintain the feasibility of the control problem as well as to ensure closed-loop stability guarantees. A drawback of online learning is the increased computational complexity, which has to be handled online while meeting real-time requirements. At the same time, online learning allows for direct incorporation of new measurement data, which can improve the prediction accuracy. Furthermore, it allows the controller to adapt to time-varying systems or changing environmental conditions as disturbances. Online learning of MPC system models with GPs is done, for example, in [171] where online training data updates, as well as hyperparameter optimisations, are performed. In [114] and [177], external signals, interpreted as references and disturbances respectively, are learned online via GPs.

In this work, offline and in principle also iterative learning of system model outputs for predictive control is proposed. The derived formulation allows us to provide guarantees for nominal stability. Moreover, the predictive controller can use its full potential when the learning-supported system representation is known from the first control iteration. Furthermore, offline and online learning of references for MPC is proposed, which provides closed-loop guarantees of the controller. While offline reference learning provides nominal stability, the online learning of references is designed to be robust against online training data updates.

5.2 Guarantees despite Learning

Even though the incorporation of learning into control allows for performance improvements, adaptation to changing situations, and can decrease manual effort, one crucial aspect is the potential loss of provable properties. In general, machine learning does not provide guarantees for closed-loop behaviour. It should, therefore, be equipped with additional features when used in a control framework to obtain those. In the

context of learning-supported predictive control, desirable guarantees are constraint satisfaction, recursive feasibility, convergence, stability, and robustness.

Combination of nominal MPC with Gaussian processes does not consider the stochastic nature of the GP. In those cases, often only the mean function of a GP is used [150, 157, 178]. For stability guarantees, the assumptions outlined in Chapter 2 can often not be used by the learning-supported MPC. To overcome this, combinations of learning-supported MPC with additional safety layers have been proposed. For example, in [3, 113, 123], and [66], a subset of the state space is considered to be safe. Any control action that keeps the system inside this set is assumed to be safe as well. As inside the safe set any control action is permitted, learning-supported MPC can be used. As soon the learning-supported controller steers the system to the boundary of the safe set a safe controller is employed to steer the system back into the interior of the safe set. This way, certain constraints in the state space are always fulfilled, which is, however, not guaranteed by the learning-supported MPC itself.

Unlike nominal MPC, stochastic MPC allows incorporating the probabilistic nature of GPs directly. If the states of the system model are stochastic (e.g. due to a GP system model) chance constraints instead of deterministic hard constraints on the states can be considered, see [93, 135, 232]. This way, probabilistic guarantees for constraint satisfaction are given instead of deterministic guarantees.

Recently, realisations of functions from a GP (see also Figure 4.3) were interpreted as scenarios such that scenario MPC can be used [223]. This formulation allows for deterministic handling of the uncertainty via the sampled functions from the GP distribution, which are deterministic. For the linear case, probabilistic performance guarantees of the controller are provided in [223].

Instead of handling uncertainties in a probabilistic way, set-based methods rely on bounded uncertainty set descriptions. Robust MPC utilises those set based uncertainty descriptions to obtain robust stability guarantees, i.e. stability despite bounded uncertainty. A learning-supported robust MPC formulation based on tube MPC was proposed in [8], where constraint tightening is used. This allows decoupling performance improvement via learning and safety via robust constraint satisfaction. Unlike the outlined safe set methods, this separation is achieved inside the MPC formulation instead by an additional encasing safety layer. Two system prediction models inside the predictive controller are considered, where one model ensures robust stability and constraint satisfaction. In contrast, the other model is learned online and used inside the cost function for performance improvements. The authors prove robust asymptotic stability with the learning-based predictive controller independently from the used learning algorithm. The approach was, for instance, applied to heat ventilation systems and quadcopters in [9] and [22]. Extensions of this idea are proposed in [138, 151] for nonlinear systems and in [19] for multi-mode systems. Alternative robust learning-based MPC formulations are given in [215] for state dependent uncertainties, and [91] for input uncertainties.

In Section 5.4, we will show how nominal MPC stability can be guaranteed despite learning of external reference signals offline or online. Moreover, we outline how to design a Gaussian process used as an output model to meet the requirements for nominal stability in Section 5.3. Furthermore, we discuss possibilities on how to extend these guarantees to a stochastic setting by incorporating the model uncertainty.

5.3 Learning of Outputs

In the following, we extend the presented nominal model predictive control schemes described in Chapter 2 by incorporating GP output models. We outline differences which occur compared to the case of first-principles model predictive control in theory and practise for the force control example presented in Chapter 3.

5.3.1 Basic Idea and Motivation

We want to use learning techniques to obtain components of the system model (2.1) in a model predictive controller using Gaussian processes. In contrast to existing works, such as [29, 93, 94, 118–120, 135, 171, 178, 180, 223, 237], we do not aim at learning the full dynamical system, the right-hand side of the differential equation (2.1a), or an input-output representation of the system. Instead, the output map of the system (2.1b) should be learned. These output maps can, for instance, incorporate nonlinear coordinate transformations that might be uncertain due to unknown parameters. For example in robotics, the direct kinematics might not be entirely known due to manufacturing inaccuracies or flexibility of certain components. Moreover, maps for the inverse kinematics are only locally defined and challenging to identify based on first principles such that learning-based methods might be helpful tools. Especially for flexible, soft or cable-driven robots, the kinematics are often hard to model based on first principles. Gaussian processes were used, for instance, in [148] to obtain a kinematic correction for a surgical robotic system, while in [55], the GP models the inverse kinematics. Other examples are output models describing the interaction of a robot with its environment, which might involve significant uncertainties stemming from unknown object positions, friction, damping or stiffness coefficients. In all these cases, machine learning can be used to obtain these mappings instead of using parametric models, cf. [27, 160, 183, 229].

One key difference in using learned output models in MPC instead of learned system dynamics is their independence of previous predictions. Unlike for dynamic system equations, no recursive evaluation of predictions must be carried out. In the context of stochastic Gaussian process models, multiple-step-ahead predictions with a (partially) learned dynamical system involves the evaluation of the GP at uncertain inputs. This recursion leads to predictions that are not normally distributed and thus are not entirely in line with the Gaussian process framework, see also Section 4.6. Different

techniques have been investigated to approximate the distributions via Gaussian distributions, see [78, 93]. However, when the GP model represents (parts of) the output map (2.1b) no uncertainty in the input occurs despite multiple evaluations over the prediction horizon. At each prediction step, the input to the GP output model is the deterministic state x obtained via the dynamic system equation (2.1a) which does not involve uncertainties. Consequently, variance information of the GP output prediction model can be determined exactly. Moreover, uncertainty does not grow over time, but it only depends on the current position in the state space.

5.3.2 Mathematical Formulation

We assume that the output equation $y = h(x)$ of a dynamical state space model is not perfectly known. Instead, noisy observations $D = \left(\boldsymbol{x}^\top, \hat{\boldsymbol{y}}^\top\right)$ with $\boldsymbol{x} = (x_1, \ldots, x_{n_D}), \hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_{n_D})$ are available describing the relation

$$\hat{y}_i = h(x_i) + \eta, \tag{5.1}$$

where we assume that $\eta \sim \mathcal{N}(0, \sigma^2)$ for $i \in \{1, 2, \ldots, n_D\}$. To identify or closely approximate $h(x)$ based on D, the following output model structure

$$\tilde{h}(x) := h_{\text{fp}}(x) + h_{\text{ml}}(x) \tag{5.2}$$

is used. Here, $h_{\text{fp}} : \mathbb{R}^{n_{\text{x}}} \to \mathbb{R}^{n_{\text{y}}}$ describes a first-principles output model which might encode prior knowledge about $h(x)$. The term $h_{\text{ml}} : \mathbb{R}^{n_{\text{x}}} \to \mathbb{R}^{n_{\text{y}}}$ denotes a data-driven model part obtained via machine learning. So to say, we use a hybrid modelling approach merging the benefits from first principles such as the quality of extrapolation with advanced learning strategies allowing for more detailed and specific models. Each of the n_{y} components of h_{ml} will be obtained via a Gaussian process trained on a modified subset of the data set D while only taking the corresponding component of the output data $\hat{\boldsymbol{y}}$ into account. In particular, the prediction data set $D_{\text{pred},j}$ and the training data set $D_{\phi,j}$ for the jth component are

$$D_{\circ,j} \subseteq \left\{ (x_i, \hat{y}_{i,j} - h_{\text{fp},j}(x_i)) \in \mathbb{R}^{n_{\text{x}}} \times \mathbb{R} \,|\, i = 1, 2, \ldots, n_D \right\} \tag{5.3}$$

with $\circ \in \{\text{pred}, \phi\}$, with $j \in \{1, 2, \ldots, n_{\text{y}}\}$ pointing to the components of the output observation $\hat{y}_i = \left(\hat{y}_{i,1}, \hat{y}_{i,2}, \ldots, \hat{y}_{i,n_{\text{y}}}\right)^\top$ and with the first-principles model $h_{\text{fp}} = \left(h_{\text{fp},1}, h_{\text{fp},2}, \ldots, h_{\text{fp},n_{\text{y}}}\right)^\top$. For a more compact notation we define

$$\check{\boldsymbol{y}}_j = \left(\hat{y}_{1,j} - h_{\text{fp},j}(x_1), \ldots, \hat{y}_{n_D,j} - h_{\text{fp},j}(x_{n_D})\right)^\top.$$

The resulting data-based model part becomes

$$h_{\text{ml}}(x) = (y_{\text{ml},1}(x), y_{\text{ml},2}(x), \ldots, y_{\text{ml},n_{\text{y}}}(x))^\top \tag{5.4}$$

with

$$y_{\mathrm{ml},j}(x) \sim \mathcal{N}(m_j^+(x), \kappa_j^+(x,x)) \tag{5.5}$$

and the posterior means $m_j^+(x)$ and variances $\kappa_j^+(x,x)$. As a consequence, the modelled system output is not deterministic, but follows a normal distribution. Hence, the model predictive control formulations from Chapter 2 have to be reformulated to be able to deal with this type of model. Specifically, the mean and the variance of the predicted output will be explicitly considered in the control problem formulation.

Unlike [10, 29, 166, 171, 238], we do not include the variance in the cost function to penalise or exploit areas where the prediction model is uncertain. Instead, the mean of the output is used in the cost function as in [178–180]. For linear systems and quadratic costs, this is equivalent to the use of the expected value of the stochastic cost, which is often used in stochastic MPC, see [88, 145]. For nonlinear systems and quadratic cost function, the variance of the output would influence the optimal solution, see also [29, 93]. However, we use the variance to modify the original system constraints to ensure the safe performance of the learning-supported model predictive controller similar to [82, 179]. The output constraints are reformulated as chance constraints and tightened by a set which includes most of the uncertainty. This strategy is based on:

Assumption 20. *There exists a finite bound $\mathscr{b}_j : \tilde{\mathcal{X}} \to \mathbb{R}$ for each approximation error $\left(h_j(x) - \mathbb{E}\left(\tilde{h}_j(x)\right)\right)$ on a compact set $\tilde{\mathcal{X}} \subset \mathbb{R}^{n_x}$ such that*

$$p\left(|h_j(x) - \mathbb{E}(\tilde{h}_j(x))| < \mathscr{b}_j(x)\right) < 1 - \epsilon \tag{5.6}$$

for all $x \in \tilde{\mathcal{X}}$ and $\epsilon \in (0,1)$.

As outlined in [122], Assumption 20 can not be fulfilled for arbitrary functions h. Nevertheless, error bounds for functions stemming from reproducing kernel Hilbert spaces are derived in [122]. In [128] the probabilistic error bounds $\mathscr{b}_j$ are derived from posterior Gaussian distributions under some mild assumptions including Lipschitz continuity of the involved covariance function and h. For a more detailed review of existing approaches and underlying assumptions to calculate error bounds, we refer to [128] and references therein.

In practice, often the posterior variance of a GP is assumed to approximate these error bounds $\mathscr{b}_j$ directly, cf. [29, 82, 215], without scaling and shifting the resulting variance in accordance to the system properties encoded for instance via the Lipschitz constants as proposed in [128]. It has been reported that this posterior variance can be a poor approximation of the actual approximation error if underlying assumptions on the GP prior or hyperparameters are wrongly made [205]. Moreover, an underestimation of the uncertainty can occur, especially in multi-step-ahead predictions as reported, for instance, in [92]. Nevertheless, the posterior variance of a GP is the main ingredient in the derived error bounds in [128]. It, therefore, is a valuable tool to

obtain a relative measure related to the error bounds if reasonable prior assumptions are posed. Moreover, in our setup, no approximations of the posterior variances are involved due to multiple-step-ahead predictions. For simplicity of presentation, we exploit multiples of the posterior standard deviation as a measure for the approximation quality. Consequently, the optimal control problems (2.5), (2.11) and (2.16) are slightly modified. In particular the constraints (2.5c), (2.11c), and (2.16d) for the control error are changed into

$$e_{\mathrm{s}}(\tau) = r_{\mathrm{s}} - \mathbb{E}\left(\tilde{h}(x(\tau))\right), \tag{5.7a}$$

$$e_{\mathrm{tt}}(\tau) = r_{\mathrm{tt}}(t_k + \tau) - \mathbb{E}\left(\tilde{h}(x(\tau))\right), \tag{5.7b}$$

$$e_{\mathrm{pf}}(\tau) = r_{\mathrm{pf}}\left(l(z(\tau))\right) - \mathbb{E}\left(\tilde{h}\left(x(\tau)\right)\right). \tag{5.7c}$$

This way, the control errors and the objective function are deterministic. Additionally, the output constraints in (2.5f), (2.11f), and (2.16f) are altered into chance constraints

$$p\left(\tilde{h}(x(\tau)) \in \mathcal{Y}\right) \geq p_{\mathcal{Y}}, \tag{5.8}$$

where $p_{\mathcal{Y}} \in (0,1)$ denotes a chosen probability for the satisfaction of the constraints. As the output is following a normal distribution, this chance constraint can be reformulated into a deterministic constraint for the mean of the output distribution via the shrinking of the original constraint set $\mathcal{Y}$. In particular, (5.8) is equivalent to

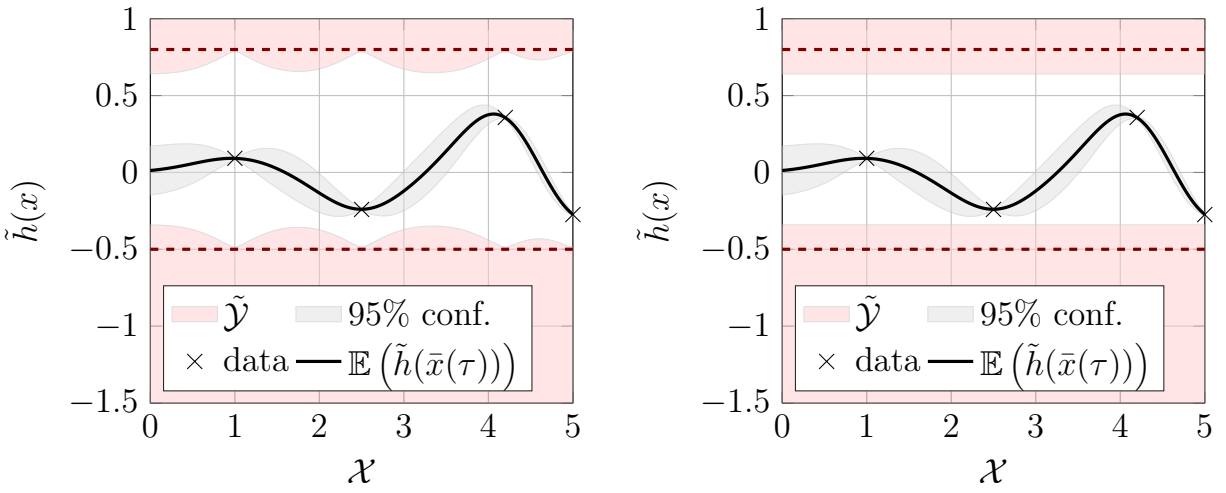
$$\mathbb{E}\left(\tilde{h}(x(\tau))\right) \in \tilde{\mathcal{Y}}, \tag{5.9}$$

where the modified constraint set $\tilde{\mathcal{Y}}$ is defined via

$$\tilde{\mathcal{Y}} := \mathcal{Y} \ominus \mathcal{R}. \tag{5.10}$$

The set $\mathcal{R}$ is constructed via the respective confidence level or error bound that belong to the desired reliability $p_{\mathcal{Y}}$, Note that $\ominus$ denotes the Pontryagin set difference, cf. Appendix A.3. For example, $\mathcal{R} = [-2\sigma_y(x), 2\sigma_y(x)]$ for a one dimensional output with $p_{\mathcal{Y}} \approx 95.45$, where σ_y is the standard deviation of the learning based output $\tilde{h}(x)$. In general the standard deviation $\sigma_y(x) = \sqrt{\kappa^+(x,x)}$ depends on the system state. Hence the tightening of the constraints can lead to state dependency of the output constraint set $\tilde{\mathcal{Y}}$ even if the original constraint set $\mathcal{Y}$ does not depend on x. Alternatively, the constraint tightening can consider the worst case realisation of the uncertainty over a compact set $\tilde{\mathcal{X}}$ such that $\sigma_{y,\max} := \max_{x \in \tilde{\mathcal{X}}} \sigma_y(x)$ is used in the construction of $\mathcal{R}$ instead of σ_y. Figure 5.2 illustrates the constraint tightening. In both cases, we need to ensure feasibility of the resulting optimal control problem with tightened constraints. Therefore, we rely on the following assumptions.

Assumption 21. *The shrunken output constraint set $\tilde{\mathcal{Y}}$ is closed, nonempty and contains the references $r_{\mathrm{s}}, r_{\mathrm{tt}}$ or r_{pf} for setpoint regulation, tracking or path following.*



(a) Based on the posterior variance (gray area) of the output model the original constraints (red dashed line) are tightened.

(b) Based on the maximum value $\sigma_{y,\max}$ of the posterior variance (gray area) the original constraints (red dashed line) are tightened robustly.

Figure 5.2: Constraint tightening.

Assumption 22. *The intersection of the state constraint set $\mathcal{X}$ and the preimage $h^{-1}\left(\tilde{\mathcal{Y}}\right)$ of the shrunken output constraint set $\tilde{\mathcal{Y}}$ is closed and nonempty.*

These assumptions limit in a way the tolerable degree of uncertainty in the map $\tilde{h}$. A proper selection of training data in a specific region of the state space can be utilised to fulfil Assumptions 21 and 22. So to say, the variance information provided by the GP allows to "evaluate" the model quality and builds a basis to decide which and how much data should be incorporated to obtain a reliable model.

5.3.3 Stability of MPC with Learned Outputs

Recursive feasibility and convergence of the optimal control problem formulations for setpoint regulation (2.5), tracking (2.11), and path following (2.16) for the general system representation (2.1) are given in Theorems 1, 2, and 3. We want to establish similar recursive feasibility and convergence for systems with a (partially) learned output model (5.2). Let us consider the nominal case, where the posterior mean of the GP together with the first-principles model part is a perfect representation of the output map. In this nominal case, $\mathbb{E}\left(\tilde{h}(x)\right) = h(x)$ and no uncertainty or noise occurs such that $\sigma_y = 0$. Hence, the output constraint set is given by $\tilde{\mathcal{Y}} = \mathcal{Y}$. Consequently the optimal control problem formulations (2.5), (2.11), and (2.16) are directly applicable for systems with the dynamics (2.1a) and the output (5.2). For this nominal case, we can state the following theorem.

Theorem 4. *A model predictive controller of the form* (2.5), (2.11) *or* (2.16) *with a (partially) learned output model* (5.2) *is recursively feasible and the respective control*

error converges to zero if $\mathbb{E}\left(\tilde{h}(x)\right) = h(x)$ *and the prior mean function* m*, prior covariance function* κ*, and the first-principles model part* h_{fp} *are continuously differentiable.*

Proof. Recursive feasibility and convergence according to Theorems 1, 2, and 3 rely upon the satisfaction of Assumption 2. To apply the same reasoning for the modified setup with the output model (5.2), differentiability and local Lipschitz continuity needs to be verified for h. The output function h is given by $h(x) = \mathbb{E}(\tilde{h}(x)) = h_{\mathrm{fp}}(x) + m(x) + \mathbf{k}^\top (K + \sigma^2 I)^{-1}\big(\boldsymbol{\gamma} - \boldsymbol{m}(\boldsymbol{\chi})\big)$, where the entries of K and $\mathbf{k}$ are $K^{i,j} = \kappa(x_i, x_j)$ and $\mathbf{k}^i = \kappa(x_i, x)$ with $i, j \in \{1, \ldots, n_{D_{\mathrm{pred}}}\}$ cf. (5.2) and (4.13). Differentiability of h is guaranteed via differentiability of the summands h_{fp}, m and κ. Local Lipschitz continuity of h on each compact set $\tilde{\mathcal{X}} \subset \mathcal{X}$ follows directly since the restriction of any continuously differentiable function on a compact set is Lipschitz. Consequently, Assumption 2 is fulfilled for $h(x) = \mathbb{E}(\tilde{h}(x))$ and recursive feasibility and convergence according to Theorems 1, 2, and 3 can be concluded. □

Consequently, standard MPC stability/convergence proofs as outlined in Chapter 2 can be applied in the nominal case for common prior mean and covariance functions, such as the squared exponential covariance, periodic covariances, or specific parametrisation of the Matérn covariance [235]. When considering the uncertainty in the output via shrunken constraints, the following can be concluded from Theorem 4.

Corollary 1. *Consider a model predictive controller for setpoint regulation* (2.5)*, trajectory tracking* (2.11) *or path following* (2.16) *with a (partially) learned output model* $\tilde{h}(x) = h_{\mathrm{fp}}(x) + h_{\mathrm{ml}}(x)$ *included via the control error formulations* $e_{\mathrm{s}}, e_{\mathrm{tt}}$ *and* e_{pf} *from* (5.7) *instead of the original error formulations* (2.5c) *for setpoint regulation,* (2.11c) *for tracking, or* (2.16d) *for path following. Furthermore, consider the shrunken output constraint set* $\tilde{\mathcal{Y}}$ *instead of* $\mathcal{Y}$ *in the optimal control problem constraints* (2.5f)*,* (2.5g)*,* (2.11f)*,* (2.11g)*,* (2.16f)*, and* (2.16h)*. Given Assumptions 21 and 22 on the shrunken constraint set* $\tilde{\mathcal{Y}}$*, the respective predictive controller is recursively feasible with the probabilistic satisfaction of the constraints and the corresponding control errors* $e_{\mathrm{s}}, e_{\mathrm{tt}}$ *and* e_{pf} *converge to zero if the prior mean* m *and covariance* κ *for the output* $\tilde{h}(x)$ *are continuously differentiable.*

Since the Assumptions 21 and 22 guarantee non-emptiness of the output and terminal constraints despite a backoff, the nominal MPC guarantees directly apply to the control errors defined in (5.7) following the same reasoning as in Theorem 4. Consequently, the Theorems 1, 2, or 3 apply directly to the posed learning-supported MPC with constraint backoff. When the backoff set $\mathcal{R}$ reflects a reliable error bound as, for instance, proposed in [128], constraint satisfaction of the true outputs $h(x)$ can be guaranteed via satisfaction of the shrunken constraints by $\tilde{h}(x)$ with a probability related to $p_{\mathcal{Y}}$. Since the posterior variance and the backoff set $\mathcal{R}$ depend on the used prediction training data D_{pred} and the involved hyperparameters ϕ an offline constraint

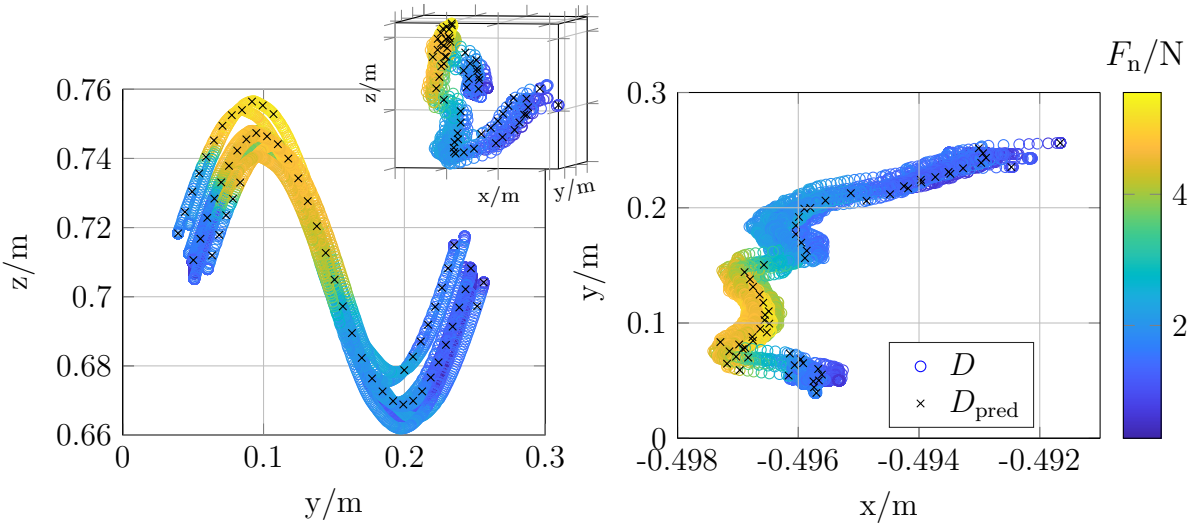


Figure 5.3: Data from the baseline controller performing multiple sinusoidal motions, where the line colour describes the normal force F_n in N.

backoff can be calculated if the output model is not adapted online. As stated in the previous subsection, we do not include the variance in the cost as we do not want to obtain a risk-seeking behaviour. Hence, our main objective in this learning-supported controller design is not to explore the uncertain areas of the state space to improve the model during runtime. However, extensions to online learning and an online update of the shrunken constraints is possible.

It should be noted that issues as recursive feasibility in classical stochastic MPC mainly stem from the fact that the predicted states from the previous iteration are not identical to the measured states used as initial conditions at the next iteration. For the output learning case, however, the uncertainty affects the output map rather than the dynamical system states. Hence, the uncertainty is considered mainly via the constraint tightening, such that safe performance and recursive feasibility can be guaranteed. If deterministic instead of probabilistic constraint satisfaction is desired, extensions of GP models with truncated multinormal distributions can be considered. For instance, in [35] posterior mean and variance for GPs with truncated distributions are derived. These could be used in set-based and robust MPC approaches, which is, however, beyond the scope of this work.

5.3.4 Illustrative Example: Learning-supported Force Control

For manipulators such as soft robots or manipulators with pneumatic actuators, dynamic models can be hard to obtain via first principles. Gaussian processes can be used to acquire the dynamic robot model for prediction inside an MPC [76, 99]. We assume that the robot dynamics are known as they have been identified to the desired degree of precision based on first principles, cf. Chapter 3. While the direct

kinematics in many setups are also known, a model of the environment is often not available. It is subject to frequent changes when the robot handles different objects with a varying size or material. Therefore, in this section, Gaussian processes are used to model the interaction force. We do not explicitly consider grasping tasks, such as [133, 229], where friction and grasping forces are learned via GPs. However, our models for the contact force can also be extended to these type of contacts. A force sensor is modelled via GPs in [97] to map between the tactile sensor values and the contact forces. In our setup, we use a 6DOF-force torque sensor instead of tactile arrays and predict upcoming contacts instead of post-processing the sensor data. Contact forces which occur in human-robot-interactions are modelled in [165] via GPs. These contact force predictions rely on impedance-based modelling of a human arm and a prediction of desired human motions. The posteriors of the desired human motions and the expected contact forces form a joint distribution. In our setup, static but unknown environments are modelled. Hence, our GP force prediction depends on the robot's motions instead of the motions of a human arm. These robot motions are free for the controller to choose and do not have to be estimated along with the contact forces as in [165, 166]. In contrast to [160], a hybrid modelling approach for the contact force is used instead of a purely learning-based contact force model. The robotic force control example from Chapter 3 is used to illustrate the proposed learning of output models via GPs. To this end, the MPC from Section 3.3 is used as a baseline controller. It is applied in a similar setup with a different environment. Since the environment material is softer than before, the previously used force model does not represent the true interaction with satisfactory accuracy. The baseline MPC is used to obtain data for system identification and machine learning.

Data Acquisition

The robot with the baseline controller performs several sinusoidal motions comparable to the reference tracking in Chapter 3. For this purpose, the Cartesian and force reference is shifted manually, resulting in varying forces over a multitude of positions. Based on the measured joint angles, the Cartesian end-effector position can be calculated via the forward kinematics. Figure 5.3 shows the performed motions in Cartesian space together with the captured force data. The measured normal contact force lies between 0.19 N and 5.58 N and is depicted via the line colour in dependence of the end-effector position in Cartesian space. Based on this experiments, a data set $D := \{((q_1, q_2, q_4)_i^\top, F_{\mathrm{n},i}) | i = 1, 2, \ldots, n_D\}$ is defined which is composed of the joint angles q with corresponding normal force F_n consisting of $n_D = 26013$ data points.

Contact Force Model Identification

The parameters of two first-principles models are identified using this data set and the foreward kinematics of the robot. Due to small penetration velocities the linear

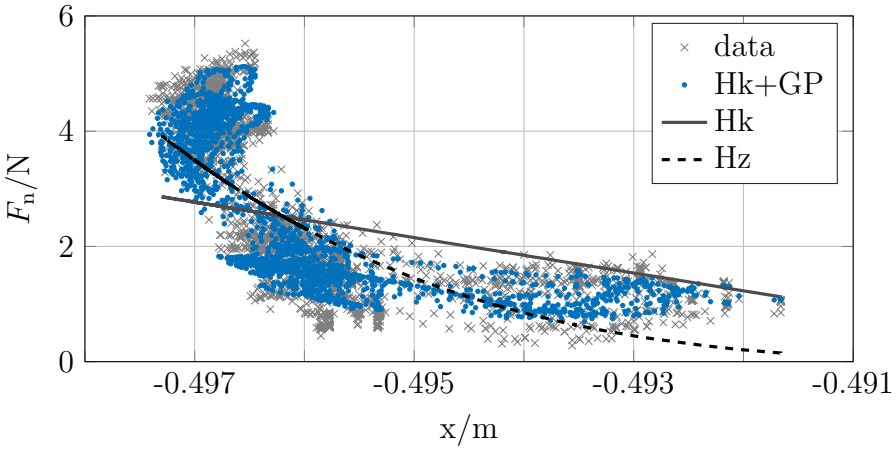


Figure 5.4: Measured and modelled forces plotted over Cartesian x-direction.

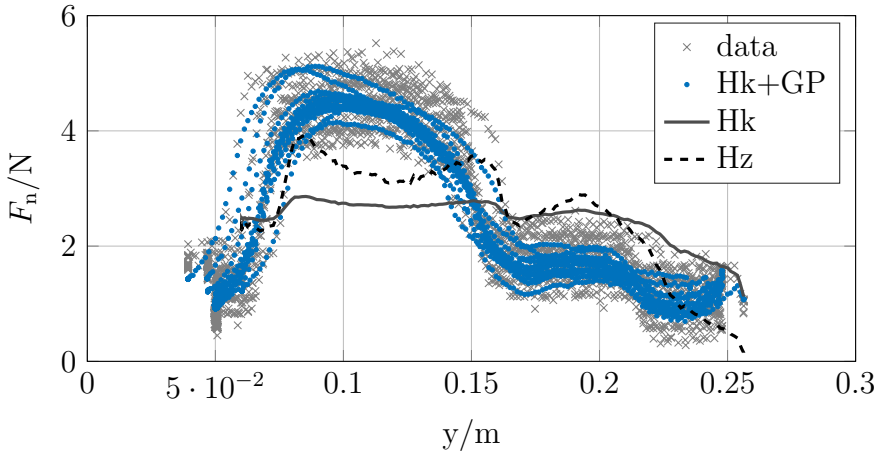


Figure 5.5: Measured and modelled forces plotted over Cartesian y-direction.

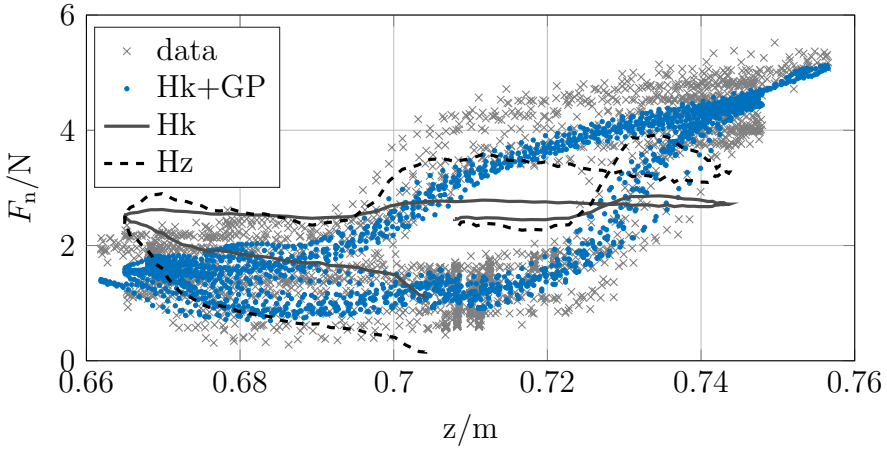


Figure 5.6: Measured and modelled forces plotted over Cartesian z-direction.

spring model of Hook (3.4) and the nonlinear spring model of Hertz (3.5) are chosen. The undeformed board position in x-direction used for identification is $p_0 = -0.488\,\mathrm{m}$ and the parameters of (3.4) and (3.5) are obtained via least squares optimisation and a two stage identifaction for the nonlinear spring model similar to [39]. The determined optimal exponent for the nonlinear Hertz model is $\alpha = 3.49$ and the stiffness parameters are $K_{e,\mathrm{Hk}} = 307.6\frac{N}{m}$ and $K_{e,\mathrm{Hz}} = 4.9090 \cdot 10^7 \frac{N}{m^\alpha}$. These linear and nonlinear spring models (3.4) and (3.5) result in predictions of the contact normal force which are displayed in Figure 5.4, 5.5, and 5.6. Here, the data (depicted as grey crosses) and the prediction of the models are shown in dependence of the Cartesian x-, y- and z-direction separately for illustrative purposes, though the data is inherently coupled. The contact force obtained via the linear spring model of Hook is depicted as a solid grey line, while the nonlinear spring model of Hertz is shown as the black dashed line. Figure 5.4 shows the most intuitive model representation over the Cartesian x-direction which directly links to the penetration depth δ.

As shown in Figures 5.4, 5.5 and 5.6 both the Hertz and Hook models cannot capture the true relation between robot position and contact force precisely. This deviation results from the inflexible model structure with only a few adjustable parameters as well as from underlying assumptions such as that the whiteboard surface is a perfect plane. The latter is not true as small irregularities are present mainly due to the attachment of a whiteboard foil on the soft sponge material. The proposed hybrid model $\tilde{h}_{F_{\mathrm{HkGP}}}$, combining the linear first-principles model with a Gaussian process, is trained using the available data. The GP captures all nonlinear effects reflecting material properties as well as unevenness of the surface. The hybrid model is given by $h_{F_{\mathrm{HkGP}}}(q) = h_{F_{\mathrm{Hk}}}(q) + m_F^+(q)$, where $m_F^+(q)$ denotes the posterior mean of a Gaussian process residual force model with the joint angles q as regressor. A squared exponential covariance function κ_{se} and a zero prior mean $m = 0$ with hyperparameters $\sigma_{\mathrm{se}} = 23\,\mathrm{N}, l_{\mathrm{se}} = 0.1\,\mathrm{rad}, \sigma_{\mathrm{n}} = 0.5\,\mathrm{N}$ are used. For computational reasons the training data set D_{pred} is based on a subset of the full data set D with $n_{D_{\mathrm{pred}}} = 60$. The data point locations for D_{pred} were chosen based on a threshold of $0.015\,\mathrm{rad}$ minimum Euclidean distance between the angular positions. They are displayed as black crosses in Figure 5.3 in Cartesian space. The output training data is constructed based on the measured forces at these positions and the subtraction of the linear spring model values at these points according to (5.3). The resulting posterior mean plus the first-principles linear spring model force is depicted in Figures 5.4, 5.5 and 5.6 as blue dots. Even though the training input data of the GP are joint angles, the predicted force is shown in the Cartesian space for illustrative reasons. As can be seen, the hybrid model consisting of a linear first-principles model and the GP can represent the measured forces without overfitting the noise. A comparison of the maximum and root-mean-square error of the linear, nonlinear, and hybrid model (Hook+GP) is given in Table 5.1. As can be seen, the nonlinear spring results in smaller maximum and root-mean-square errors than the linear one. However, the hybrid model outperforms

Table 5.1: Validation of force models over the full data set D with $n_D = 26013$ via maximum and root-mean-square error (RMSE).

	linear spring (Hook)	nonlinear spring (Hertz)	Hook+GP
maximum error	2.9568 N	2.6717 N	1.4031 N
RMSE	1.1843 N	0.9125 N	0.3754 N

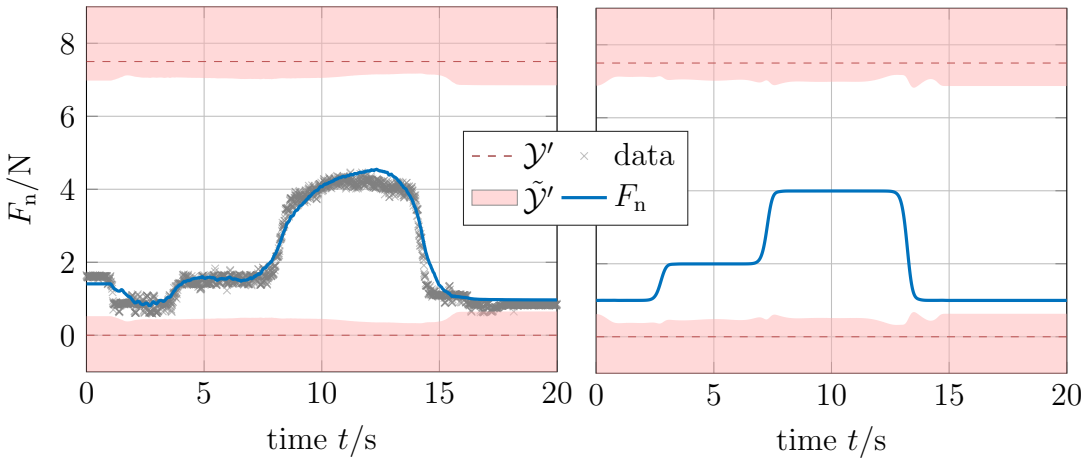


Figure 5.7: Constraint tightening based on posterior variance.

both first-principles models by a root-mean-square error reduction of around 68 % compared to the linear model and around 58 % compared to the nonlinear model. The spatial distribution of the model error is shown in Appendix A.5. For further illustration of the model quality the $2\sigma_y$ confidence regions are depicted in Figure 5.7. The left plot in Figure 5.7 shows the evaluation of the posterior GP prediction at test points from the data set obtained by the baseline controller. The right plot shows the posterior of the GP along the corresponding reference. The force constraints $\text{proj}_3(\mathcal{Y}) = [0\,\text{N}, 7.5\,\text{N}]$ shown as dashed line in Figure 5.7 are tightened by the $2\sigma_y$ confidence bounds. The complement of the tightened set is shown as a red shaded area. In both plots, the posterior mean of the trained GP (solid blue line) does not violate or activate the tightened constraints. Moreover, the variance along the path and in its vicinity shows only small variation, which indicates a suitable choice of the training data. For this setup, a constant over-approximation of the $2\sigma_y$ bound with 0.75 N seems possible. However, this over-approximation only is suitable in the vicinity of the reference path and not in the full state space. Due to the additional computational complexity for high numbers of training data and the used compilers of the ACADO code-generation toolbox, a state-dependent online constraint tightening was not implemented. Validation of each simulation and experiment with tightened constraints was performed instead. As will be shown, these tightened constraints are never active or violated, cf. Appendix A.5.

Controller Design

Based on the derived models, three different model predictive controllers for a path following task are designed. Their setup is equivalent to the path following controller used in Section 3.3. Hence, we use the same reference path to define a desired motion and contact force to encode the interactive writing task. A slight modification in the hardware setup (the pen is held by a small rigid penholder instead of the Barrett hand) leads to minor changes in the kinematic and dynamic parameters of the robot model. The corresponding transformations are given in Appendix A.2. The system states, outputs and inputs are equal to Section 3.3. The output map for the contact force, however, will be selected among the previously identified linear spring, nonlinear spring, and hybrid learning-based model.

A simulation study compares these three output models used in a path following controller. The dynamic system simulation model used inside the optimal control problems as well as to simulate the real system measurements is identical. Only the force model inside and outside of the predictive controller will deviate. In all three simulations, the real interaction force is simulated with a Gaussian process that was trained directly on the measurements in Figure 5.3. This model has no hybrid structure and uses a prediction data set of $n_D = 419$ data points. This setup allows obtaining an exact model of the real interaction, which is however not tailored towards the use in a real-time optimisation-based predictive controller, see also Appendix A.5.

Simulation Results

In Figure 5.8, a comparison of the closed-loop performance of three different controllers is given, where each of them uses one of the identified contact force models from Figure A.2 and Table 5.1. Details of the chosen tuning parameters and further simulation results are given in Appendix A.5. The controlled Cartesian output direction and the corresponding path following errors are shown in Figure 5.8 top and middle. As can be seen, the path reference (thin black line) is followed precisely with tracking errors under 0.1 mm each. While the controller with the linear spring (Hook) and the hybrid GP model perform similarly well, slightly larger tracking errors occur for the nonlinear spring (Hertz) especially in the beginning of the path between 2 s and 3 s. The path following performances in the force-controlled output direction is shown in Figure 5.8, bottom. The linear spring shows the worst behaviour resulting in control errors up to 6.58 N. These errors originate from the large model-plant mismatch between the simulation of the real interaction and the linear model in the MPC. They cause constraint violation of both the maximum allowed force of 7.5 N as well as the minimum allowed force of 0 N to prevent from contact loss. The root-mean-square control error is 3.24 N. In comparison, the nonlinear spring model performs significantly better in the closed-loop. No constraint violation occurs, the maximum control error is 1.63 N, and the root-mean-square control error is 0.85 N. The hybrid model consisting of

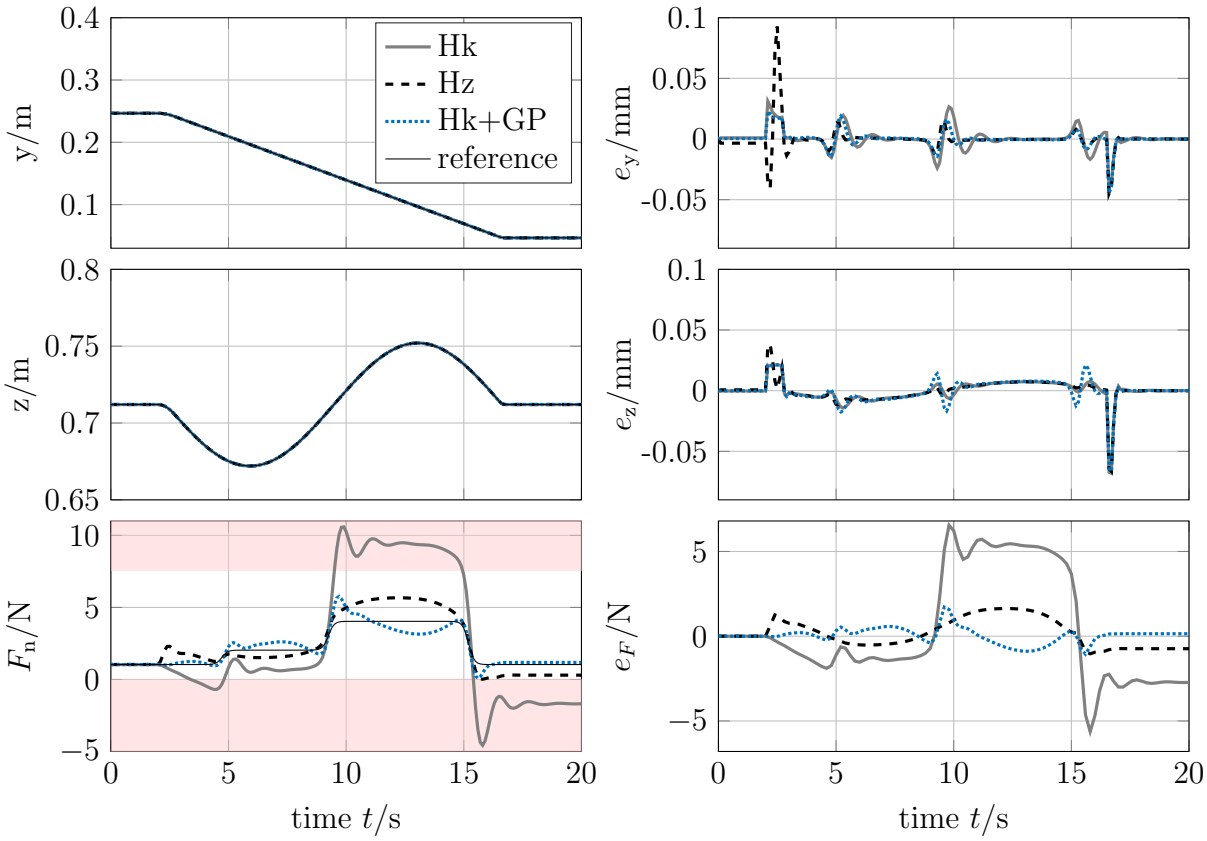


Figure 5.8: System outputs and control errors for path following MPC.

the linear spring plus a GP model outperforms the nonlinear spring in terms of the root-mean-square control error which is 0.45 N. Hence it reduces this error by around 47 % compared to the Hertz model. Nevertheless, large maximum errors up to 1.79 N occur. Consequently, this controller setup should be improved before performing experiments. We propose to include online measurements in the predictive controller to adapt the force models online as also outlined in Chapter 3.

Adaptive Force Models

We want to adapt the force model online. The model-plant mismatch $\tilde{F}_{\mathrm{n},k} \in \mathbb{R}$ at time t_k is calculated via $\tilde{F}_{\mathrm{n},k} = h_{\mathrm{n}}(\tilde{F}_k) - h_{F_i}(\tilde{q}(t_k))$. The simulated force $h_{F_i}(\tilde{q}(t_k))$ is subtracted from the normal contact force based on the actual measured contact wrench $\tilde{F}_k$ at t_k. In this example h_{F_i} is either the linear spring, the nonlinear spring or the learned hybrid model, i.e. $i \in \{\mathrm{Hk}, \mathrm{Hz}, \mathrm{HkGP}\}$. The output is $y_{\mathrm{pf}} = \tilde{h}(x) = (p_{\mathrm{e,y}}, p_{\mathrm{e,z}}, F_{\mathrm{n}}, z_1)^\top$ where the prediction of the normal contact force starting at time t_k is $\bar{F}_{\mathrm{n}}(\tau) = h_{F_i}(\bar{q}(\tau)) + \tilde{F}_{\mathrm{n}}(t_k)$ for all $\tau \in [t_k, t_k + T]$. The predictions inside the controller are denoted by $\bar{\cdot}$ while $\tilde{\cdot}$ indicate sensor readings. As discussed in Chapter 3, this formulation is equivalent to an update of the initial conditions in a differential equation. A comparison between a dynamic interpretation of the static force model and this adaptive setup is provided in Appendix A.6. Including the available measure-

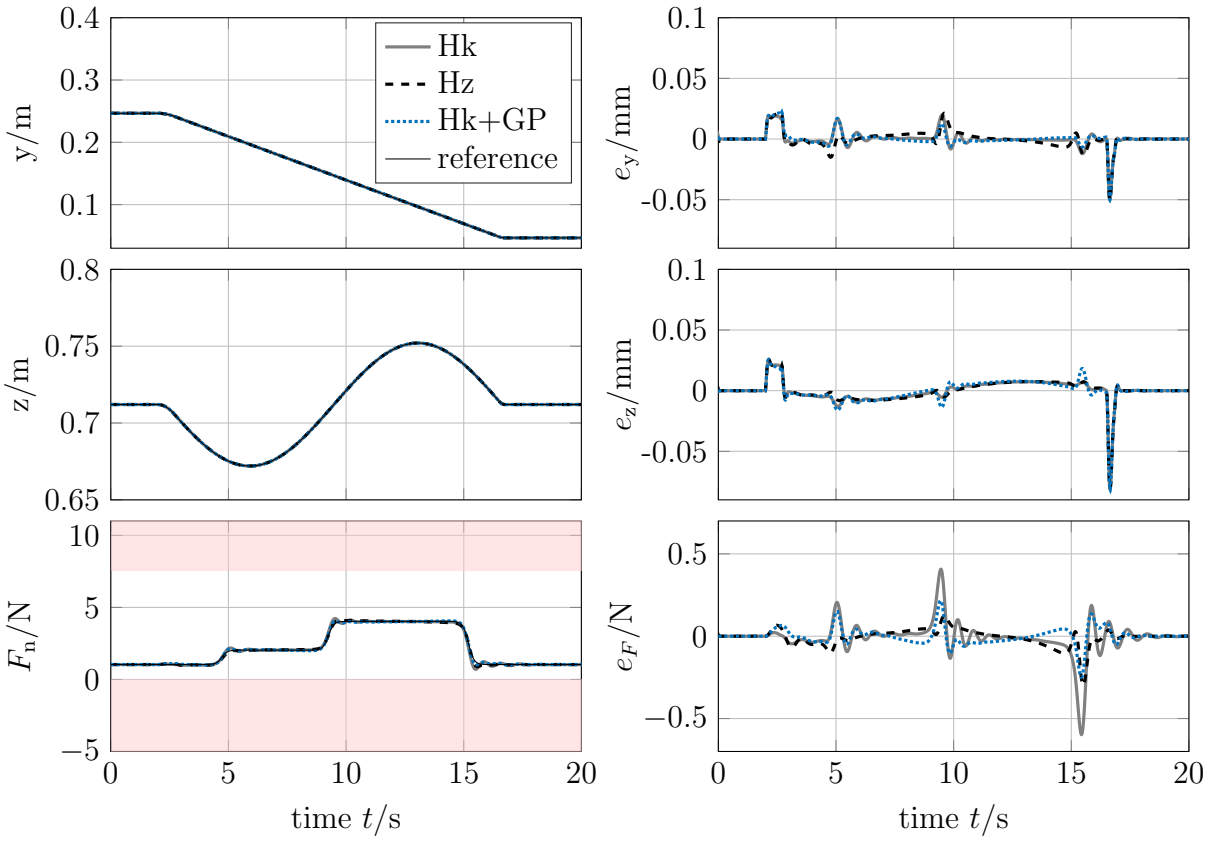


Figure 5.9: System outputs and control errors for adaptive path following MPC.

ments via the model adaptation similar to an initial condition allows for a significant increased closed-loop control performance, as shown in Figure 5.9. The same controller parameters as in Figure 5.8 are used. The effect of the model-plant mismatch is drastically reduced in all three model scenarios. The maximum control errors in the force direction are 0.60 N, 0.30 N, and 0.25 N for linear, nonlinear and hybrid model respectively. The respective root-mean-square errors are 0.09 N, 0.05 N, and 0.05 N. Hence, the root-mean-square error for the nonlinear spring is equivalent to the hybrid model in the adaptive case. The adaptation and online feedback of the force sensor information compensate the substantial model-plant mismatch for the nonlinear spring model adequately. This performance improvement also shows the inherent robustness of the model predictive controller, which can cope with non-perfect models. Still, it is not always trivial to come up with a suitable (nonlinear) first-principles model. In these cases, machine learning approaches like the proposed Gaussian-process-based force model provide an appealing strategy as they allow to obtain precise output representations. While the proposed hybrid model outperforms the nonlinear first-principles model in the non-adaptive case, it performs equally well in terms of the root-mean-square error. It results in smaller maximum errors when both are used in an adaptive setting. In both cases, a significant improvement compared to the linear case is visible. Since these simulation results show promising controller performance, a validation of

Table 5.2: Evaluation of experimental closed-loop performance via maximum and root-mean-square error (RMSE).

	e_F		e_y		e_z	
	max.	RMSE	max.	RMSE	max.	RMSE
Hook	1.28 N	0.49 N	1.54 mm	1.09 mm	1.79 mm	0.76 mm
Hertz	0.71 N	0.26 N	1.33 mm	0.96 mm	1.47 mm	0.73 mm
Hook+GP	0.79 N	0.23 N	1.55 mm	1.13 mm	1.40 mm	0.71 mm

the approach was performed in experiments.

Experimental Results

The adaptive control setup allows constraint satisfaction that is critical for safe evaluation of the different controller concepts in experiments. Hence, the adaptive strategy for the MPC with the linear, nonlinear, and hybrid model is applied to the real robot. Again, the same controller setup and parameters as in the simulations are used. The resulting closed-loop behaviour of the output is shown in Figure 5.10, while further figures are provided in Appendix A.5. In all three cases, the closed-loop controller is real-time feasible and satisfies the state, input, and output constraints. Moreover, a comparable qualitative closed-loop performance is obtained in all cases as the path following errors in the Cartesian and the force-controlled subspaces lie in the same order of magnitudes, cf. Figure 5.10. In Table 5.2, a detailed numerical comparison of the control errors in the linear, nonlinear, and hybrid case is provided. The root-mean-square errors for the Cartesian y-direction differ only up to 0.17 mm in magnitude between the three control setups, cf. Table 5.2 fifth column. In the Cartesian z-direction the average performance of all three force model setups deviates even less with 0.05 mm maximum difference (Table 5.2 last column). However, the control errors in Cartesian y- and z-direction are more than a magnitude larger than in the simulations. In Figure 5.10, a clear trend for the control error evolution depending on the path can be seen. While the control error e_y grows along the path, e_z changes its sign when the movement changes from upwards to downwards motion. A model-plant mismatch exists, which causes this effect depending on the motion along the path. We assume that the unmodelled friction along the path as well as other unmodelled components in the contact wrench cause this effect. Hence, future work should consider these additional contact effect besides the normal contact force that was investigated here. The normal contact force F_n and the corresponding control errors e_F are depicted in Figure 5.10, bottom. Here the MPC with a linear contact force model shows a deviation from the reference with maximum control errors of 1.28 N and root-mean-square error of 0.49 N. As in the simulations, the predictive controllers with nonlinear and hybrid force model outperform the linear case with a maximum force reduction of

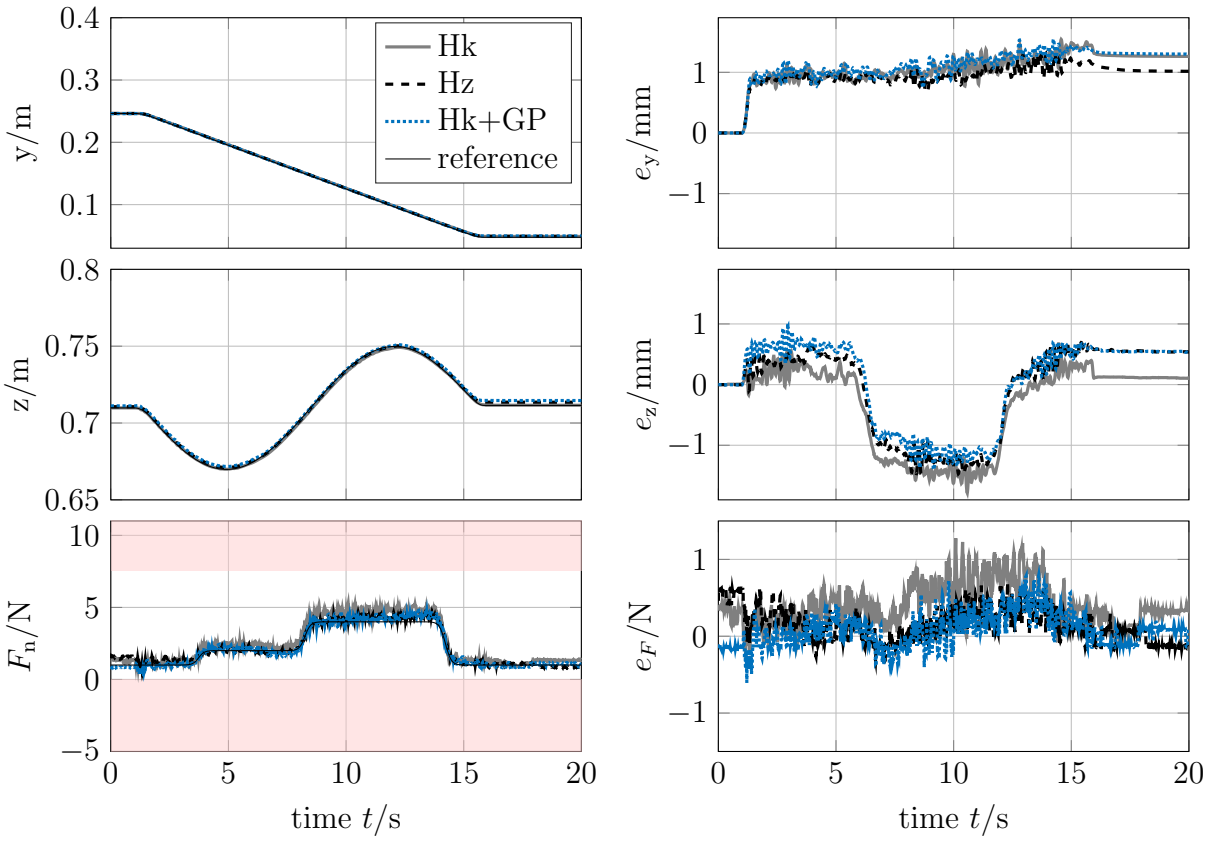


Figure 5.10: Experimental results for adaptive path following MPC with linear spring model (Hk), nonlinear spring model (Hz), and hybrid model (Hk+GP).

around 44.5 % and 38.2 % each, cf. Table 5.2. The controllers with nonlinear spring and learning-based hybrid model both show a significant reduction of the contact force error compared to the linear spring MPC, as shown in Table 5.2. The MPC with the hybrid model based on the Gaussian process exhibits the best control performance regarding the root-mean-square error of 0.23 N.

As a conclusion, we can state that the proposed learning-supported model predictive controller enables the direct control of contact forces in robotics. The hybrid GP model allows for a significantly improved model quality. Moreover, we have shown that the hybrid model can be used in an experimental, real-time feasible, adaptive setting where it performs at least equally well compared to detailed nonlinear first-principles models while using less prior knowledge and it outperforms the baseline controller with linear contact force model.

5.4 Learning-supported MPC for Reference Tracking

In this section, we derive a way to combine machine learning via Gaussian processes for reference learning with model predictive control while giving stability guarantees.

There exist many applications where the reference for a controlled system is obtained

via interaction of the system with the environment. One example are chemical reactors in a plant in which the reference of one process depends on the operation of other processes, e.g. due to changing compositions of the feed streams. Another example is hand guiding of robots by human operators to teach the robot a desired motion or the cooperative manipulation of objects by multiple robots. Motion compensation in medical robotics also requires a synchronised movement to human motions, see Chapter 6. Moreover, autonomous cars in a platooning scenario can improve their performance by learning a desired motion or velocity given by the leading vehicle.

In all these cases, the reference is contained in data, which can be corrupted with measurement noise or suffer from data loss due to communication. This can be tackled by modelling of the reference and using the reference model as a back-up or pre-filter. Moreover, reference models allow predicting likely future reference values. This prediction of the future can improve the performance of model predictive control. Naturally, the control inputs in MPC are results of an optimisation over the prediction horizon. When changing references for trajectory tracking or path following are considered, their future values should be available to the controller. Otherwise, performance might deteriorate, and stability might be lost. Consequently, we propose to use a data-based approach to model the reference.

Gaussian processes allow coping with noisy data since their stochastic nature allows to incorporate noise information directly while avoiding overfitting. This is fundamental, as noisy references might not be followable or trackable. Moreover, noisy references should not be followed or tracked exactly since they do not reflect the originally desired evolution. On top, references might not be followable or trackable from the beginning, even in the nominal case. Constraints in the learning of the GP are proposed to handle such situations. Unlike [1, 35] or [202], these constraints do not refer to truncated multi-normal distributions or implicit correlations of multidimensional GPs. Moreover, we do not learn constrained dynamics, as done in [75]. Instead, the constraints are included in the training of the GP via a constrained hyperparameter optimisation. All in all, we use GPs to learn, predict, filter, and if necessary, modify a given reference such that the proposed learning-supported MPC is recursively feasible, and the control error converges to zero. Thereby, tasks and benefits of reference governors [72] and prediction filters [107] are merged into a data-based machine learning framework suitable for MPC. While we focus on continuous-time systems with flatness properties, the results can be generalised, see, e.g. [156] and [158] where we use reachability analysis to inform the learning about system properties for time-discrete systems.

We decompose our task as shown in Figure 5.11. Gaussian processes serve as a reference generator, where the posterior mean of the GP provides the reference. In case of multiple outputs, we will use n_y individual GPs motivated by our results in [157]. The output reference $\hat{r}(t)$ is the function we want to learn. The data $D = \{(t_i, r_i)\}$ with $i \in \{1, 2, \ldots, n_D\}$ is depicted as black crosses in Figure 5.11 and describes the

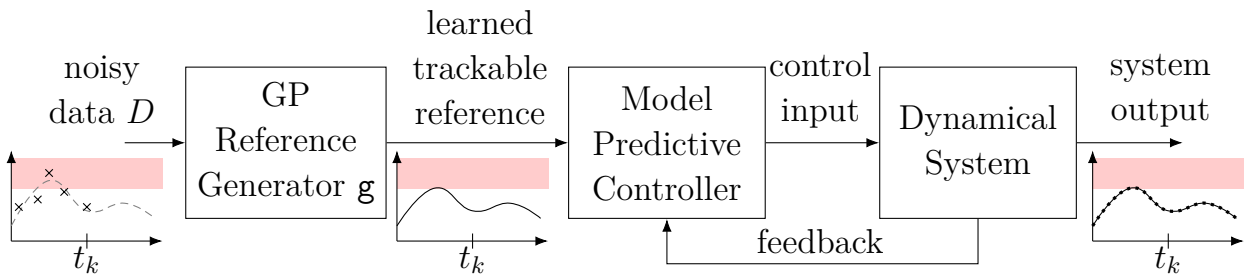


Figure 5.11: Learning-supported model predictive controller with GP reference generator. With the data available up to time t_k the GP learns a trackable reference (solid) enabling prediction ($t > t_k$) and constraint satisfaction. Hence, exact tracking (dotted line) can be achieved.

desired reference $\hat{r}$ (depicted as gray dashed line) with some additive noise η. The hyperparameter training data D_ϕ might use all data in D or can be a subset of it, such that $\boldsymbol{\chi}_\phi = (t_1, t_2, \ldots, t_{n_{D_\phi}})$ and $\boldsymbol{\gamma}_\phi = (r_1, r_2, \ldots, r_{n_{D_\phi}})$. The red shaded area in Figure 5.11 depicts constraints that the reference should fulfil. These can be composed of output constraints represented by $\mathcal{Y}$ and preimages of the state constraint set $\mathcal{X}$ under the output map h. The original reference $\hat{r}$ might or might not fulfil those constraints. Furthermore, due to noise in the training data, constraints might be violated even if the original reference fulfils them. The GP reference predictor should use the data and learn a reference r_{tt} that satisfies the constraints. Moreover, it should take the dynamics of the system into account while learning a reference such that it can be tracked exactly. This learned reference (depicted as a solid black line) enters the model predictive controller. The GP reference generator should be designed such that stability conditions for the predictive controller apply. We use the following formalisation of reference learning for trajectory tracking:

Task 4. *(Reference generator for trajectory tracking with output data).*
Given the system (2.1)*, a prediction horizon T, and data/measurements $D \in \mathcal{D} := \prod_{i=0}^{n_D} \mathbb{R}_0^+ \times \mathbb{R}^{n_y}$ describing the desired reference $\hat{r} : \mathbb{R}_0^+ \to \mathbb{R}^{n_y}$. Design a reference generator $\mathsf{g} : \mathbb{R}_0^+ \times [0, T] \times \mathcal{D} \to \mathcal{Y}$, which at time t_k provides a reference $r_{\text{tt}} : [t_k, t_k + T] \to \mathcal{Y}, (t) \mapsto r_{\text{tt}}(t) := \mathsf{g}(t_k, \tau, D)$, which fulfils:*

(i) Trackability: The reference r_{tt} is trackable, i.e it fulfils the output constraints $r_{\text{tt}}(t) \in \mathcal{Y}$ and can be followed given the system dynamics once starting on it, hence $\exists u_{\text{tt}}(t) \in \mathcal{U}$ such that $r_{\text{tt}}(t) = h(x_{\text{tt}}(t))$.

(ii) Data fitting: The reference r_{tt} finds a trade off between model complexity and data consistency, i.e. $r_{\text{tt}}(t_i) \approx r_i$ for $(t_i, r_i) \in D$.

In the following, we outline how to achieve Task 4 by addressing the prediction of the reference, fitting of data, and ensuring trackability.

Reference Prediction: GPs allow building data-based prediction models, where the posterior mean of the GP can perform predictions and thereby addressing Task 4. However, in the considered case, the prediction involves extrapolation into the future, while the available data lies in the past. In general, extrapolation is a non-trivial task with purely data-based prediction models. Extrapolation with GP models, in particular, is challenging, as the posterior mean approaches its prior if the distance between test and training data points is large. An illustration of this is effect is shown in Figure 4.1a where the posterior mean (depicted in solid blue) approaches the value of the prior mean $m = 2.78$ in areas with few data. Nevertheless, the internal structure of specific references can be exploited in the Gaussian process design to allow for prediction with better approximation quality. The Sections 5.4.1-5.4.3 show extrapolation for asymptotically constant and periodic references with GPs. Different distance measures (included via the prior covariance function κ and the contained hyperparameters) and a priori knowledge is used for this purpose. This highlights a general benefit of Gaussian processes, as GPs allow to include knowledge of the system in a structured way.

Data Fitting: Gaussian processes as reference generators naturally fulfil the filtering properties (Task 4(ii)): The inference of a GP is based on conditioning its distribution on the data. Loosely speaking, it prioritises sample functions which are consistent with the data and assigns low probabilities to functions which are not, see also Figure 4.3. Nevertheless, the measurement noise η, which is encapsulated in the training data, is taken into account during inference via the measurement noise variance σ^2, cf. (4.13). This way, sample functions with exact fit to the noisy data are not necessarily preferred or more probable than sample functions with approximate fitting. Moreover, the hyperparameter optimisation via the logarithmic marginal likelihood (4.11) allows finding hyperparameter while trading off model complexity and data fitting, cf. Section 4.4.

Trackability: The obtained references r_{tt} should be trackable. To address output trackability, which is hard to verify in general, we exploit the concept of differential flatness. Following [56, 131, 200] differential flatness for systems can be defined as

Definition 6. *The system* (2.1) *is differentially flat with respect to the output* $\xi = (\xi_1, \ldots, \xi_{n_\mathrm{u}})^\top$ *if (at least locally) the variable* ξ *can be written as a function of the state* x*, the input* u*, and a finite number of time derivatives of the input variable such that* $\xi = \Pi\left(x, u, \dot{u}, \ldots, u^{(\lambda)}\right)$.

Flatness implies that the system variables x and u can be expressed as functions of the variable ξ and a finite number of time-derivatives of ξ. Hence,

$$x = \Phi\left(\xi, \dot{\xi}, \ldots, \xi^{(\beta-1)}\right) \tag{5.11}$$

and

$$u = \Psi\left(\xi, \dot{\xi}, \ldots, \xi^{(\beta)}\right). \tag{5.12}$$

We exploit this property by assuming:

Assumption 23. *The system* (2.1) *is differentially flat with respect to its output* y.

Differential flatness of system (2.1) with respect to the output y allows to connect the output, state, and input via (5.11) and (5.12). We enforce this connection via constraints in GP learning, which boils down to a constrained hyperparameter optimisation. Moreover, the proposed constraints guarantee the satisfaction of the state, input, and output constraint sets $\mathcal{X}, \mathcal{U}$, and $\mathcal{Y}$. In combination, this allows for trackability of the learned reference. The resulting constrained hyperparameter optimisation problem can be written as

$$\hat{\phi} := \arg\max_{\phi} \ \ln p(\boldsymbol{\gamma}_\phi, |\boldsymbol{\chi}_\phi, \phi) \tag{5.13a}$$

subject to $\forall \tau \in \left[0, \bar{t}\right]$

$$m^+ \left(\tau; D_\phi, \phi\right) \in \mathcal{Y} \tag{5.13b}$$

$$\Phi\left(m^+(\tau; D_\phi, \phi), \dot{m}^+(\tau; D_\phi, \phi), \ldots, m^{+(\beta-1)}(\tau; D_\phi, \phi)\right) \in \mathcal{X} \tag{5.13c}$$

$$\Psi\left(m^+(\tau; D_\phi, \phi), \dot{m}^+(\tau; D_\phi, \phi), \ldots, m^{+(\beta)}(\tau; D_\phi, \phi)\right) \in \mathcal{U}. \tag{5.13d}$$

As before, the cost (5.13a) is the logarithmic marginal likelihood, while additional constraints have been included: Via (5.13b), the posterior mean m^+ of the GP, which models the output reference r_{tt}, is constrained by the output constraint set $\mathcal{Y}$. The corresponding reference state and input expressions are obtained due to flatness via the mappings Φ and Ψ from (5.11) and (5.12). They are constrained to $\mathcal{X}$ and $\mathcal{U}$ via (5.13c) and (5.13d). The necessary derivatives $\dot{m}^+, \ddot{m}^+, \ldots, m^{+(\beta)}$ of the posterior mean with respect to time (which is the regressor) can be obtained via (4.14). Regarding the proposed constrained learning of GPs, we can state the following proposition:

Proposition 1. *If optimisation problem* (5.13) *is feasible, the GP provides a trackable reference* $r_{\mathrm{tt}}(\tau) = m^+\left(\tau; D_\phi, \hat{\phi}\right)$ *for all* $\tau \in \left[0, \bar{t}\right]$.

Note that one can conclude trackability of the reference only up to time $\bar{t}$. In the following, two common types of references, namely asymptotically constant and periodic references, will be considered to establish trackability for all times.

5.4.1 Asymptotically Constant References

Asymptotically constant references change for a limited time span T_{trans} and converge to a constant afterwards, see Figure 5.12. For instance, asymptotically constant references can be used to describe set point transitions. Examples for asymptotically constant references are the transition of a chemical plant from one operating point to another or the parking of a car. The basic idea to show trackability for all times is

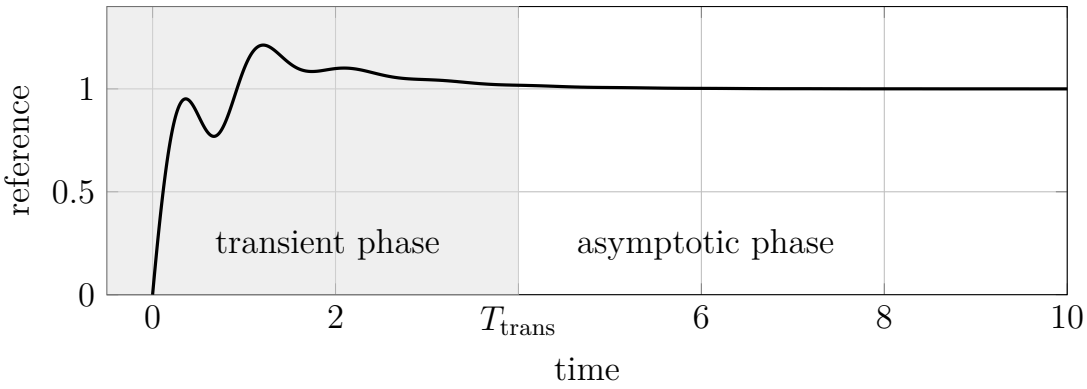


Figure 5.12: Illustration of an asymptotically constant reference. During the transient phase (grey area) the reference varies arbitrary whereas after T_{trans} it converges to a constant.

to split the problem into two phases. The first phase covers the transient phase with the constraints in problem (5.13). At the same time, we propose to encode knowledge about the reference in the prior mean and covariances for the steady state to guarantee trackability for the asymptotic phase. The aim is to find a time $\bar{t}$ for which constraints are satisfied for all times $t > \bar{t}$. To do so, we enforce the learned reference at time $\bar{t}$ to lie in suitably calculated invariant sets as outlined in the following.

We use a decaying covariance function κ and a constant prior mean $m(\tau) = \mu \, \forall \tau \in \mathbb{R}$. Assuming fixed hyperparameters ϕ and a fixed training data set D_ϕ the posterior mean (4.13) can be reformulated as a weighted sum

$$m^+(\tau) = \mu + \sum_{i=1}^{n_{D_\phi}} c_i \kappa(t_i, \tau). \tag{5.14}$$

Here, $t_i \in \boldsymbol{\chi}_\phi$ from D_ϕ. The constant coefficients c_i depend on the fixed hyperparameters ϕ and the training data D_ϕ. A bound on m^+ can be obtained via the triangular inequality such that

$$|m^+(\tau) - \mu| \leq \bar{m}(\tau) := \sum_{i=1}^{n_{D_\phi}} |c_i| \kappa(t_i, \tau), \tag{5.15}$$

where $\bar{m}$ is monotonously decreasing for $\tau \geq t_i$. Similar to (5.14) and (5.15) the β^{th} derivative of m^+ can be expressed via

$$m^{+(\beta)}(\tau) = \sum_{i=1}^{n_{D_\phi}} c_i \kappa^{(\beta)}(t_i, \tau),$$

with a corresponding bound

$$|m^{+(\beta)}(\tau)| \leq \bar{m}^{(\beta)}(\tau) := \sum_{i=1}^{n_{D_\phi}} |c_i||\kappa^{(\beta)}(t_i, \tau)|. \tag{5.16}$$

The bound $\bar{m}^{(\beta)}$ is monotonously decreasing for all $\tau \geq t_i$ and $|\tau - t_i| \geq \varsigma(\phi)$. Based on these bounds, invariant sets for the output, state, and input references can be obtained assuming that the maps Φ and Ψ are bounded:

$$\mathcal{Y}(\tau) := [\mu - \bar{m}(\tau), \mu + \bar{m}(\tau)], \tag{5.17a}$$

$$\mathcal{X}(\tau) := \Big\{x \Big| x = \Phi\left(y, \dot{y}, \ldots, y^{(\beta-1)}\right), y \in \mathcal{Y}(\tau), \dot{y} \in [-\dot{\bar{m}}(\tau), \dot{\bar{m}}(\tau)], \ldots, y^{(\beta-1)} \in [-\bar{m}^{(\beta-1)}(\tau), \bar{m}^{(\beta-1)}(\tau)]\Big\}, \tag{5.17b}$$

$$\mathcal{U}(\tau) := \Big\{u \Big| u = \Psi(y, \dot{y}, \ldots, y^{(\beta)}), y \in \mathcal{Y}(\tau), \dot{y} \in [-\dot{\bar{m}}(\tau), \dot{\bar{m}}(\tau)], \ldots, y^{(\beta)} \in [-\bar{m}^{(\beta)}(\tau), \bar{m}^{(\beta)}(\tau)]\Big\}, \tag{5.17c}$$

Once these sets satisfy the output, state, and input constraints, we can guarantee the satisfaction of those constraints for all times since we use a decaying covariance function. To satisfy these conditions, we propose an iterative learning procedure with sampling time T_s:

Algorithm 1 GP Learning for asymptotically constant references

Init training data set D_ϕ and transition time $\bar{t} > t_i \, \forall i \in \{1, \ldots, n_{D_\phi}\}$
while true **do**
  obtain $\hat{\phi}$ via (5.13) using $\left(\bar{t}, D_\phi\right)$
  if $|t_i - \bar{t}| > \varsigma\left(\hat{\phi}\right)$ **then**
    compute $m^+(\bar{t}), \bar{m}(\bar{t}), \dot{\bar{m}}(\bar{t}), \ldots, \bar{m}^{(\beta)}(\bar{t})$ via (4.13),(5.15),(5.16) using $\hat{\phi}$
    compute $\mathcal{Y}, \mathcal{X}, \mathcal{U}$ via (5.17)
    if $\mathcal{Y} \subseteq \mathcal{Y}, \mathcal{X} \subseteq \mathcal{X}, \mathcal{U} \subseteq \mathcal{U}$ **then**
      break
    end if
  end if
  $\bar{t} \leftarrow \bar{t} + T_\mathrm{s}$
end while
return $\bar{t}, \hat{\phi}$

If Algorithm 1 terminates in finite time, the following Lemma can be stated:

Lemma 1. *Consider a differential flat system* (2.1) *and a Gaussian process trained via Algorithm 1 with constant prior mean m and decaying prior covariance function κ.*

The reference $r_{\mathrm{tt}}(t) = m^+\left(t; D_\phi, \hat{\phi}\right)$ *provided by this GP is trackable according to Definition 1 for system* (2.1) *for all times* $t > 0$.

Proof. Once Algorithm 1 converges, the time $\bar{t}$ and the optimised hyperparameters $\hat{\phi}$ are obtained. The constrained optimisation problem (5.13) guarantees the trackability of the learned reference up to time $\bar{t}$ via imposing constraints (5.13b)-(5.13d) on the learned GP mean m^+ and its derivatives. From time $\bar{t}$ on, the monotonicity of the mean and its derivatives is fulfilled (cf. Line 5 in Algorithm 1). Given monotonicity of $\kappa, |\dot{\kappa}|, \ldots, |\kappa^{(\beta)}|$ the bounds $\bar{m}, \dot{\bar{m}}, \ldots \bar{m}^{(\beta)}$ are monotonously decreasing for all $t \geq \bar{t}$. It follows that $\mathcal{Y}(t) \subseteq \mathcal{Y}\left(\bar{t}\right), \mathcal{X}(t) \subseteq \mathcal{X}\left(\bar{t}\right), \mathcal{U}(t) \subseteq \mathcal{U}\left(\bar{t}\right)$ for all $t \geq \bar{t}$, i.e. the sets $\mathcal{Y}\left(\bar{t}\right), \mathcal{X}\left(\bar{t}\right)$, and $\mathcal{U}\left(\bar{t}\right)$ are invariant. Line 8 in Algorithm 1 implies that $\mathcal{Y}(t) \subseteq \mathcal{Y}, \mathcal{X}(t) \subseteq \mathcal{X}$, and $\mathcal{U}(t) \subseteq \mathcal{U}$ for all $t \geq \bar{t}$. Consequently, trackability of the reference $r_{\mathrm{tt}} = m^+$ is guaranteed via the satisfaction of the output, state, and input constraints for all times $t \in \mathbb{R}_0^+$. □

Example 1. *We consider the control of a well-mixed continuously stirred tank reactor (CSTR) with an irreversible, exothermic reaction* $\mathrm{A} \rightarrow \mathrm{B}$*, see Figure 5.13. The CSTR is equipped with a cooling jacket where the coolant flow rate is the control input to the system* $u = F_{\mathrm{c}}$*. The inlet flow rate to the reactor is assumed to be equal to the outlet flow rate such that the volume inside the reactor is constant. Based on the energy and mass balances, a system of nonlinear differential equations can be derived [152]. Those describe the dynamic behaviour of the two states of the reactor* $x = [c_{\mathrm{A}}, T_{\mathrm{R}}]^\top$ *which are the concentration of species* A *and the reactor temperature. A linearisation of the system around the operating point* $c_{\mathrm{A}} = 0.265\,\frac{\mathrm{kmol}}{\mathrm{m}^3}, T_{\mathrm{R}} = 394\,\mathrm{K}, F_{\mathrm{c}} = 15\,\frac{\mathrm{m}^3}{\mathrm{min}}$ *leads to the following linearised system*

$$\begin{aligned} \dot{x} &= \begin{pmatrix} -7.5763 & -0.0935 \\ 854.9129 & 5.8153 \end{pmatrix} x + \begin{pmatrix} 0 \\ -6.0831 \end{pmatrix} u, \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} x, \end{aligned} \tag{5.18}$$

where the output of the system is the concentration of A*. The task is to bring the system from an operating point of* $c_{\mathrm{A}} = 0.305\,\frac{\mathrm{kmol}}{\mathrm{m}^3}$ *to the considered linearisation point* $c_{\mathrm{A}} = 0.265\,\frac{\mathrm{kmol}}{\mathrm{m}^3}$*. Meanwhile, constraints in the output concentration, the temperature, and the coolant flow rate must be satisfied. These are* $\mathcal{Y} = [0.165\,\frac{\mathrm{kmol}}{\mathrm{m}^3}, 0.365\,\frac{\mathrm{kmol}}{\mathrm{m}^3}], \mathcal{X} = [0.165\,\frac{\mathrm{kmol}}{\mathrm{m}^3}, 0.365\,\frac{\mathrm{kmol}}{\mathrm{m}^3}] \times [389\,\mathrm{K}, 394.1\,\mathrm{K}]$*, and* $\mathcal{U} = [13\,\frac{\mathrm{m}^3}{\mathrm{min}}, 18\,\frac{\mathrm{m}^3}{\mathrm{min}}]$*. The reference trajectory for the transition originates from a downstream process which demands specific amounts of concentrations. This reference is not provided as a smooth, analytic function* $\hat{r}$ *but in terms of noisy data* r_i*, as shown in Figure 5.14. Based on this training data* D_ϕ*, the constrained hyperparameter optimisation* (5.13) *is performed via Algorithm 1. We use the squared exponential covariance function as a prior and a zero prior mean. Algorithm 1 is initialised with* $\bar{t} = 5\,\mathrm{min}$*. At time* $\bar{t} = 7.7\,\mathrm{min}$*,*

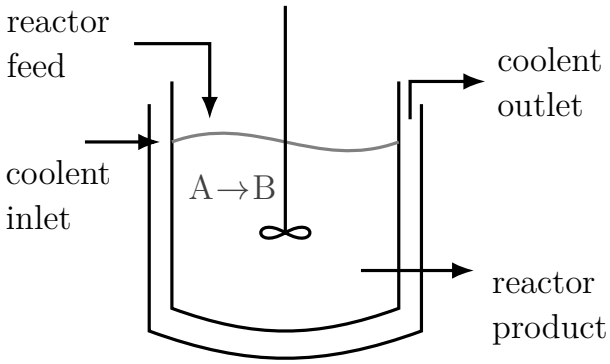


Figure 5.13: Schematic depiction of a continuously stirred tank reactor with a cooling jacket.

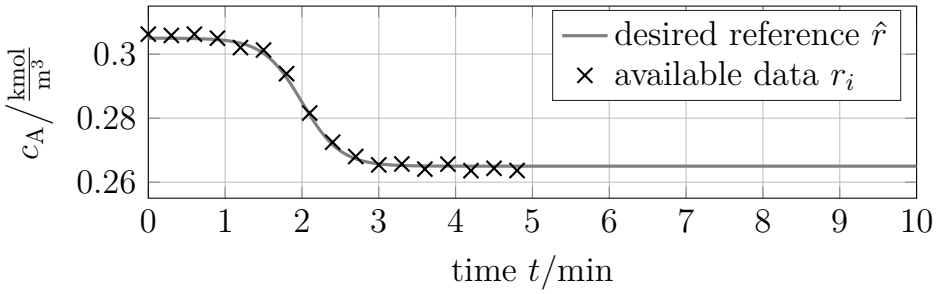


Figure 5.14: The desired reference (solid gray) is unknown analytically, but available in terms of noisy data (black crosses).

all constraints are satisfied such that Algorithm 1 terminates and provides valid hyperparameters $\hat{\phi} = (\hat{\sigma}_{\text{se}}, \hat{l}_{\text{se}}) = (0.014, 0.94)$ *that guarantee trackability of the learned reference for all times. The resulting references* $r_{\text{tt}}, x_{\text{tt}}$ *and* u_{tt}*, which satisfy all constraints (depicted as red shaded areas), are shown in Figure 5.15 in blue solid line. For comparison, a GP is trained on the same data with an unconstrained hyperparameter optimisation and the resulting learned references are depicted in Figure 5.15 in black dash-dotted lines. In contrast to the constrained GP, the references provided by the unconstrained GP violates the state constraints for* $t \in [4.72\,\text{min}, 6.7\,\text{min}]$*. Consequently, this reference would not be trackable for system* (5.18) *and can lead to the infeasibility of a model predictive controller that uses this reference. As a side effect, adding constraints can lead to an improvement in the approximation quality if the original unknown reference* $\hat{r}$ *fulfils these constraints since additional information is included during the learning phase. The approximation errors for the output references are depicted in Figure 5.16 for the constrained and unconstrained case. As can be seen in Figure 5.16, the maximum approximation error is reduced by adding the constraints into the learning.*

Based on the learned references, a model predictive controller according to (2.11) *can be designed. In this example, the cost function has quadratic form* $L_{\text{tt}} = e_{\text{tt}}^\top Q_{\text{tt}} e_{\text{tt}} + w_{\text{tt}}^\top R_{\text{tt}} w_{\text{tt}}$ *and* $E_{\text{tt}} = \varepsilon_{\text{tt}}^\top Q_{\text{E,tt}} \varepsilon_{\text{tt}}$ *with* $Q_{\text{tt}} = 1000, R_{\text{tt}} = 0.1$ *and* $Q_{\text{E,tt}} = \text{diag}(0.1, 0.1)$*. At the prediction horizon* $T = 0.5\,\text{min}$ $x(t_k + T) = x_{\text{tt}}(t_k + T)$ *is chosen as the terminal*

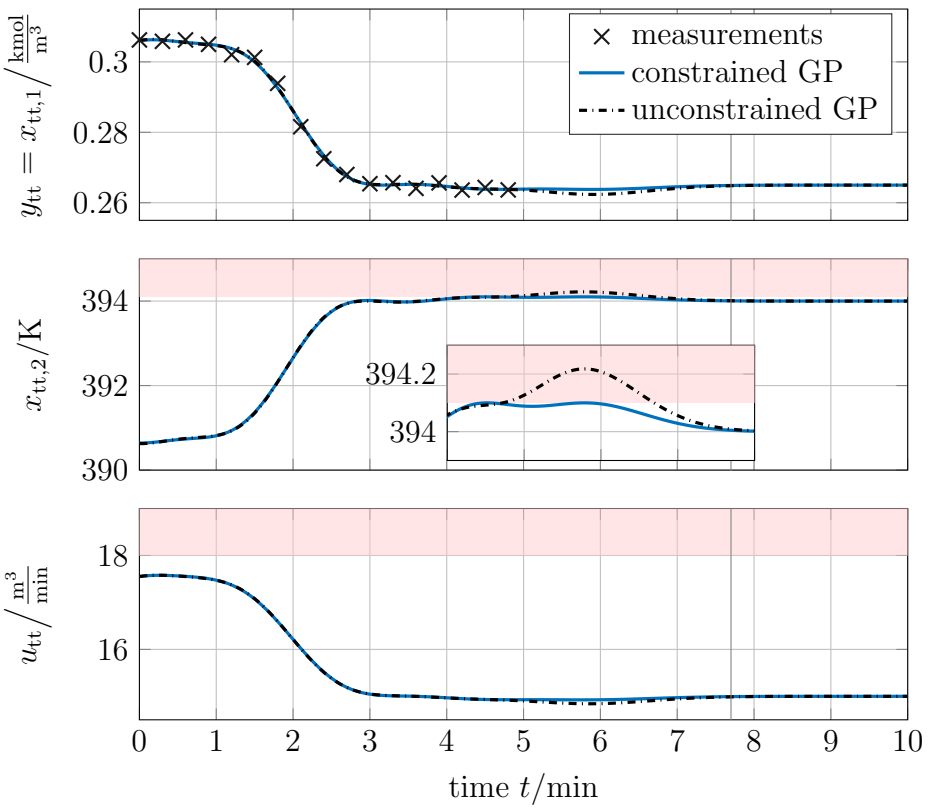


Figure 5.15: Learned references from GP hyperparameter optimisation with and without constraints.

constraint set $\mathcal{F}_{\mathrm{tt}}$. The resulting control errors are shown in Figure 5.17. Therein, the control errors of the same MPC with different reference formulations are shown. When unfiltered noisy observations are used, the control errors (grey dashed lines) become large as the reference is not trackable. Moreover, the OCP is infeasible at several time instances. Using an unconstrained GP leads to smoother references and smaller control errors (black dash-dotted). However, the OCP is infeasible for $t \in [4.35\,\mathrm{min}, 6.4\,\mathrm{min}]$ as the terminal equality constraint violates the state constraints. In contrast to those references, the constrained GP learning leads to a trackable reference and recursive feasibility of the OCP. Moreover, it results in the smallest control errors (solid blue line). The corresponding control inputs, as well as their learned references, are depicted in Figure 5.18. While the control inputs (thin black line) and their references for the unconstrained and constrained case (middle and bottom) show decent values, the unfiltered reference leads to noisy and non-admissible input references (grey dashed). The MPC tries to follow the noisy reference, which leads to a noisy input as well. This overfitting is undesirable in most cases, e.g. due to wear and tear of actuators. Overall this example illustrates the ability of constrained GP learning to predict, filter and modify references such that they are suited for controller design and MPC in particular.

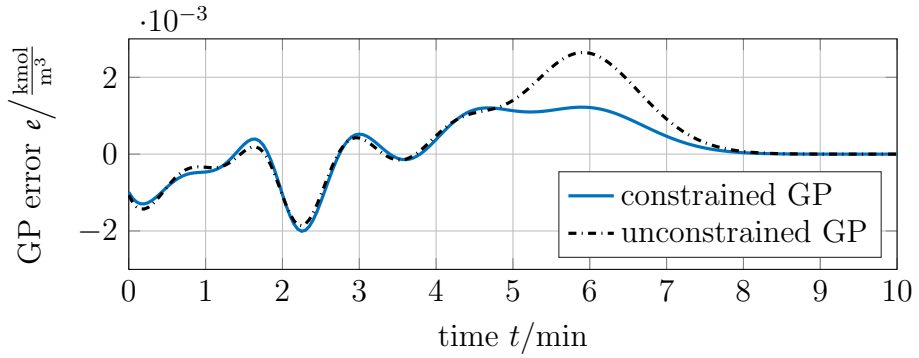


Figure 5.16: Output error to the ground-truth function of the GP.

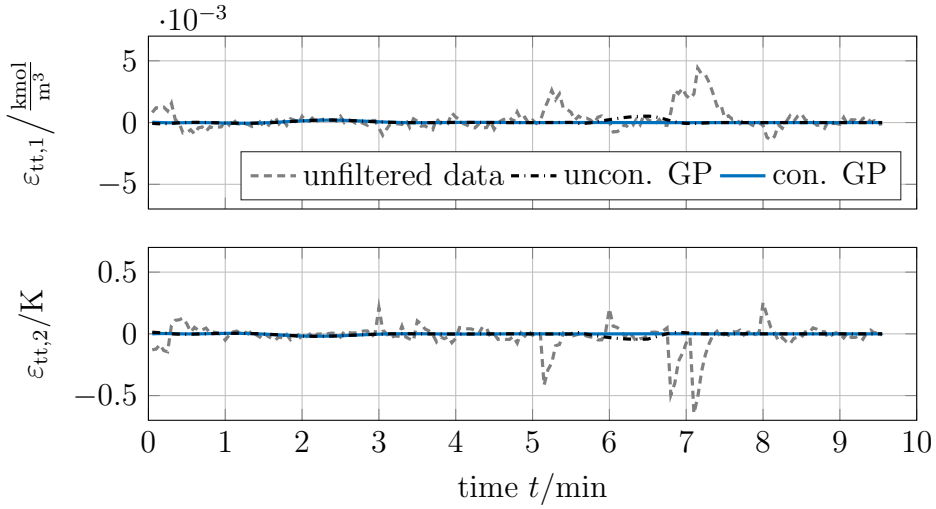


Figure 5.17: Control error to the learned reference of the MPC.

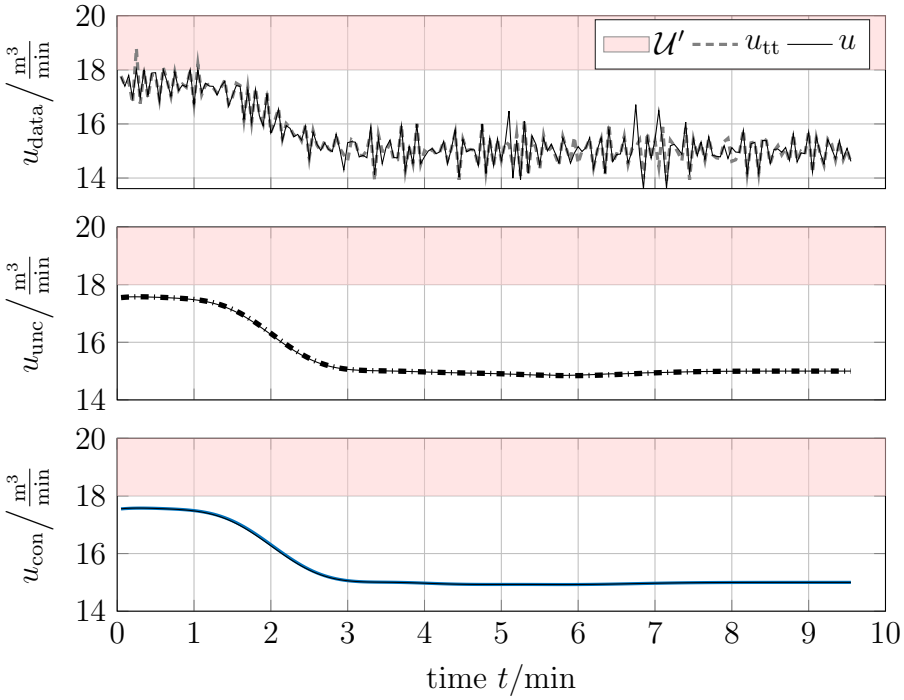


Figure 5.18: Optimal control inputs.

5.4.2 Learning of Periodic References

Periodic references occur for example in cyclic operations of chemical reactors, day-night-cycles in building climate control, or in robot manufacturing, where the same item is produced repeatedly. Gaussian processes are suitable to learn periodic references and to predict them since the periodicity of the signal can be included in the structure of the GP. This can be achieved by choosing periodic covariance functions, cf. Definition 4. An example for a periodic covariance function is

$$\kappa_{\mathrm{per}}(t,t') = \sigma_{\mathrm{per}}^2 \exp\left(-\frac{2}{l_{\mathrm{per}}^2}\sin^2\left(\frac{\pi}{T_{\mathrm{per}}}(t-t')\right)\right),$$

where the hyperparameters ϕ are the period time T_{per}, and the length scales σ_{per} and l_{per}. One can obtain periodicity of the references using a periodic covariance κ and a constant prior mean $m(t) = \mu$ skipping the decaying condition on κ. Trackability can be enforced for all times via solution of (5.13) using $\bar{t}$ larger than the period length T_{p} of the signal $\hat{r}$. As the period length T_{p} is approximated via the hyperparameter T_{per} of the periodic covariance function, this can be realised by adding an additional constraint on the hyperparameters in (5.13). The slightly modified constrained hyperparameter optimisation problem becomes

$$\hat{\phi} := \arg\max_{\phi} \ \ln p(\boldsymbol{\gamma}_\phi, |\boldsymbol{\chi}_\phi, \phi) \tag{5.19a}$$

subject to $\forall \tau \in \left[0, \bar{t}\right]$

$$m^+\left(\tau; D_\phi, \phi\right) \in \mathcal{Y} \tag{5.19b}$$

$$\Phi\left(m^+(\tau), \dot{m}^+(\tau), \ldots, m^{+(\beta-1)}(\tau)\right) \in \mathcal{X} \tag{5.19c}$$

$$\Psi\left(m^+(\tau), \dot{m}^+(\tau), \ldots, m^{+(\beta)}(\tau)\right) \in \mathcal{U} \tag{5.19d}$$

$$H\phi \leq \bar{t}. \tag{5.19e}$$

Here, (5.19b)-(5.19d) enforce similar constraints as in (5.13). Merely, an additional linear constraint on the hyperparameters via (5.19e) is included, where H is a vector of size $1 \times n_\phi$ which is multiplied to the hyperparameter vector ϕ such that $H\phi = T_{\mathrm{per}}$. This additional constraint allows learning of references that are trackable for all times:

Lemma 2. *Consider a flat dynamical system* (2.1) *and a Gaussian process with constant prior mean m and periodic prior covariance function κ. A reference $r_{\mathrm{tt}}(t) = m^+(t; D_\phi, \hat{\phi})$ provided by such a GP trained via* (5.19) *is trackable according to Definition 1 for system* (2.1) *for all times $t > 0$.*

Proof. The constraints in (5.19) use the flatness property of the system to ensure trackability of the reference up to time $\bar{t}$. The constant prior mean and periodic prior covariance guarantee that $m^+(t) = m^+(t+T_{\mathrm{per}}), \dot{m}^+(t) = \dot{m}^+(t+T_{\mathrm{per}}), \ldots m^{+,(\beta)}(t) =$

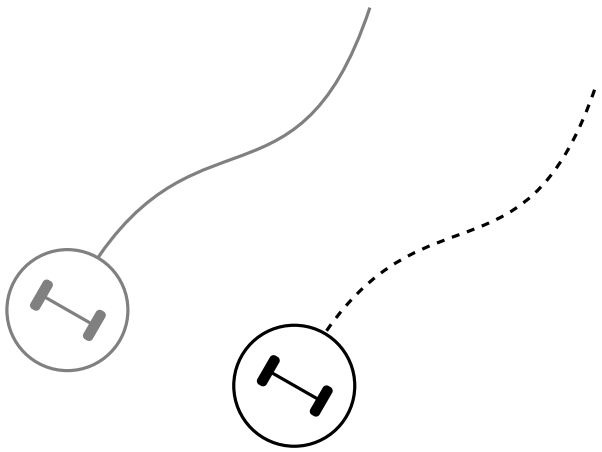


Figure 5.19: Mobile robot (black) in a cooperative task synchronising its motion to a second robot (gray). The desired reference (dashed line) is learned based on sensor data capturing the motion of the second robot (gray solid line) and a desired distance to it.

$m^{+,(\beta)}(t+T_{\mathrm{per}})$ such that $r_{\mathrm{tt}}(t) = r_{\mathrm{tt}}(t+T_{\mathrm{per}}), x_{\mathrm{tt}}(t) = x_{\mathrm{tt}}(t+T_{\mathrm{per}})$, and $u_{\mathrm{tt}}(t) = u_{\mathrm{tt}}(t+T_{\mathrm{per}})$ for all $t \in \mathbb{R}_0^+$. Since $r_{\mathrm{tt}}(t) \in \mathcal{Y}, x_{\mathrm{tt}}(t) \in \mathcal{X}$, and $u_{\mathrm{tt}}(t) \in \mathcal{U}$ for all $t \in \left[0, \bar{t}\right]$ and $\bar{t} \geq T_{\mathrm{per}}$ it can be concluded that $r_{\mathrm{tt}}(t) \in \mathcal{Y}, x_{\mathrm{tt}}(t) \in \mathcal{X}$, and $u_{\mathrm{tt}}(t) \in \mathcal{U}$ for all $t \in \mathbb{R}_0^+$, i.e. the reference is trackable according to Definition 1 for system (2.1). □

Example 2. *A mobile robot is considered which should synchronise its motion to another mobile robot for example to transport heavy items in a cooperative manner. This task leads to periodic motions if the same track is driven repeatedly as for instance in manufacturing processes where items are produced and handled repeatedly. This cooperative task is illustrated in Figure 5.19. The nonlinear dynamics of the mobile robot are given by*

$$\dot{x}_1 = u_1 \sin(x_3) \tag{5.20}$$

$$\dot{x}_2 = u_1 \cos(x_3) \tag{5.21}$$

$$\dot{x}_3 = u_2 \tag{5.22}$$

where the states x_1 and x_2 describe the Cartesian horizontal and vertical position of the robot and x_3 is the heading angle. The system is controlled via the speed u_1 and the turning rate u_2. The inputs are constrained by $\mathcal{U} = [1.88\,\frac{\mathrm{m}}{\mathrm{s}}, 2.1\,\frac{\mathrm{m}}{\mathrm{s}}] \times [-1\,\frac{\mathrm{rad}}{\mathrm{s}}, 2\,\frac{\mathrm{rad}}{\mathrm{s}}]$. The considered flat output of the system is

$$y = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^\top. \tag{5.23}$$

This system has multiple inputs and outputs, such that we need to train multiple GPs to learn the desired reference motion. Even though the two GPs for the two dimensional output do not capture dependencies between the output distributions, the learning of them is coupled by the considered constraints and due to shared hyperparameters. The

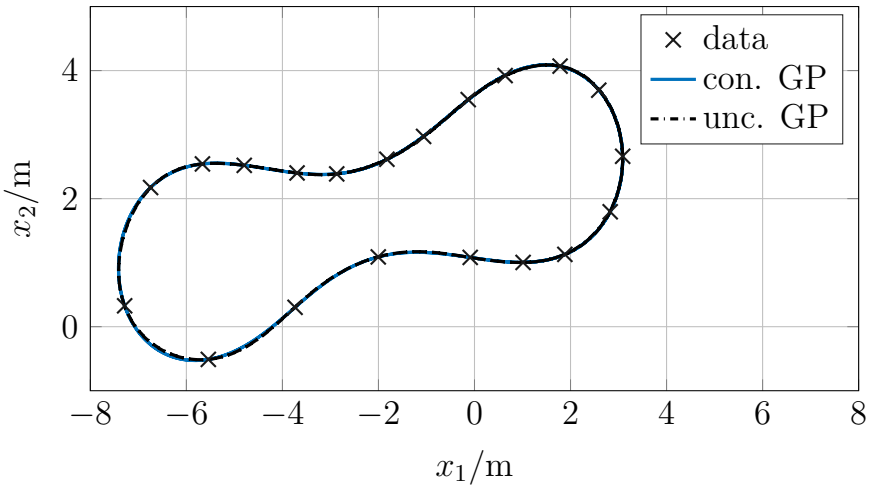


Figure 5.20: Based on the captured reference data (marked with crosses) the reference output trajectories are learned with a constrained (blue, solid line) and an unconstrained (black, dashdotted line) optimisation.

same covariance function κ_{per} *is used for both GPs where the period length hyperparameter* T_{per} *is identical for each, such that the optimisation variables in total are* $\phi = (\sigma_{\mathrm{per},1}, \sigma_{\mathrm{per},2}, l_{\mathrm{per},1}, l_{\mathrm{per},2}, T_{\mathrm{per}})$. *The training data* D_ϕ *for the hyperparameter optimisation with a standard deviation of* $\sigma = 0.001\,\mathrm{m}$ *is depicted as black crosses in Figure 5.20. Based on this data the constrained hyperparameter optimisation* (5.19) *results in the hyperparameter values* $\hat{\phi} = (4.18, 2.14, 0.79, 0.79, 12.5\,\mathrm{s})$. *The system outputs (mean of GPs) with those parameter values are shown in Figure 5.20 in solid blue line. The corresponding state and output evolutions over time are depicted in Figure 5.21 (solid blue line). For comparison, an unconstrained GP is trained on the same data and is depicted in black dash-dotted line in Figures 5.20 and 5.21. Both GPs show a similar evolution at first glance with only small deviations from the true underlying function. These deviations, i.e. the GP approximation errors, are depicted in Figure 5.22. However, the maximum and average errors for both outputs of the constrained GP (solid blue) are smaller than the approximation errors of the unconstrained GP (black dash-dotted line). Furthermore, the unconstrained GP learns an output reference trajectory whose corresponding reference input violates the input constraints of the robot, cf. Figure 5.23 black dash-dotted line. In contrast, the constrained GP learns a reference trajectory which takes all constraints into account, see the solid blue line in Figure 5.23. This constrained learning allows for trackability of the learned reference.*

A model predictive controller is designed to track the references. The stage and terminal cost have quadratic form (as in Example 1) with weightings $Q_{\mathrm{tt}} = \mathrm{diag}(100, 100)$, $R_{\mathrm{tt}} = \mathrm{diag}(0.1, 0.1)$ *and* $Q_{\mathrm{E,tt}} = \mathrm{diag}(0.1, 0.1, 0.1)$. *The prediction horizon is* $T = 0.5\,\mathrm{s}$ *and we use a terminal equality constraint* $x(t_k + T) = x_{\mathrm{tt}}(t_k + T)$. *The resulting control errors of this controller for both references are depicted in Figure 5.24. The same MPC shows different performance while tracking the learned references. While*

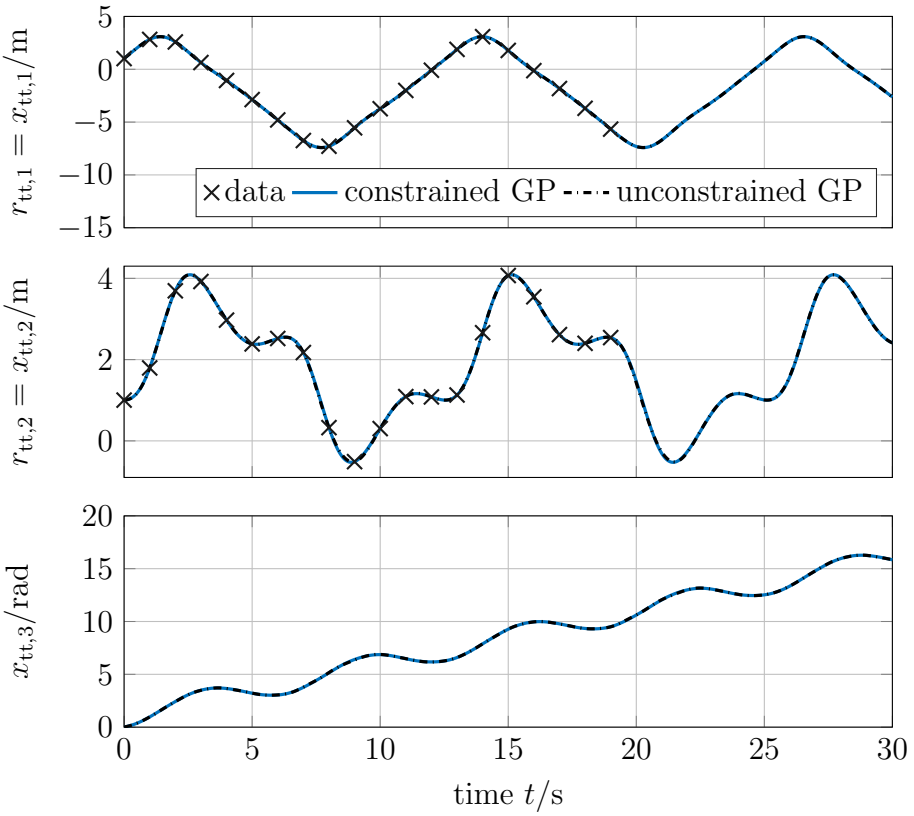


Figure 5.21: Reference trajectories $r_{\mathrm{tt},1}$ and $r_{\mathrm{tt},2}$ are learned from noisy output data (black crosses) and mapped to state reference trajectories via $x_{\mathrm{tt},1} = r_{\mathrm{tt},1}, x_{\mathrm{tt},2} = r_{\mathrm{tt},2}$ and $x_{\mathrm{tt},3} = \operatorname{atan}\left(\frac{\dot{r}_{\mathrm{tt},2}}{\dot{r}_{\mathrm{tt},1}}\right) + i\pi$, where i is the number of periods.

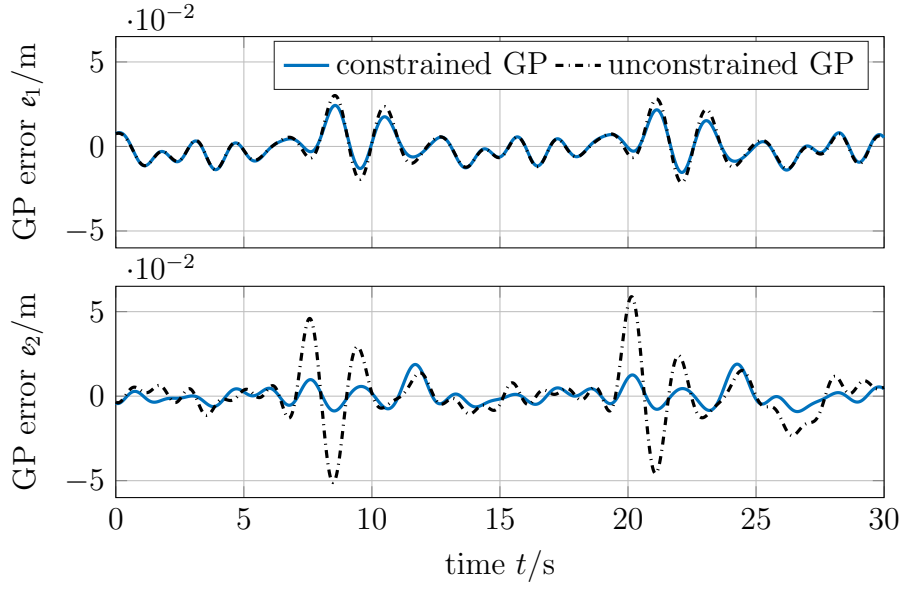


Figure 5.22: Approximation error of a constrained and an unconstrained GP $e := \hat{r} - r_{\mathrm{tt}}$.

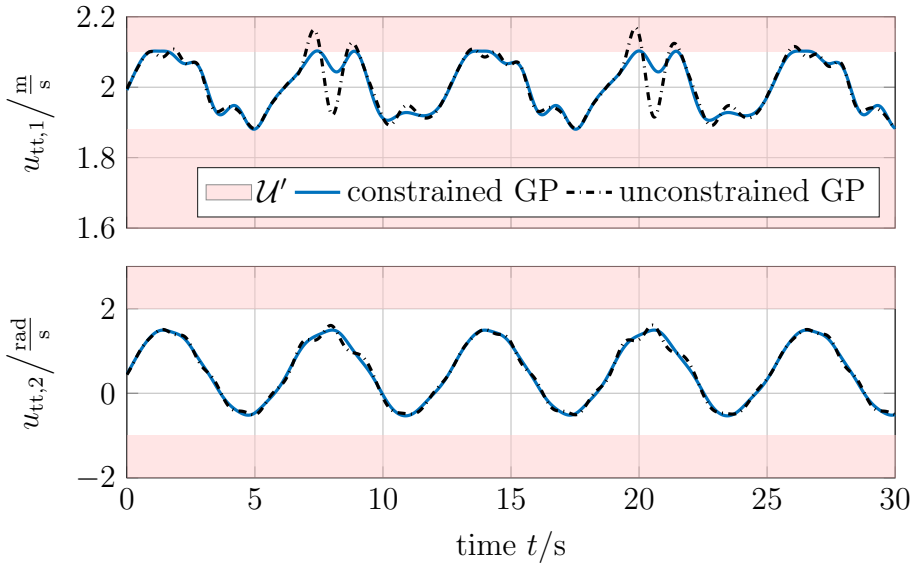


Figure 5.23: Corresponding input references u_{tt} to the learned output references r_{tt} obtained via $u_{\mathrm{tt}} = \Psi\left(r_{\mathrm{tt}}, \dot{r}_{\mathrm{tt}}, \ddot{r}_{\mathrm{tt}}\right)$ which is $\Psi = \left(\sqrt{\dot{r}_{\mathrm{tt},1}^2 + \dot{r}_{\mathrm{tt},2}^2}, \quad \frac{\ddot{r}_{\mathrm{tt},2}\dot{r}_{\mathrm{tt},1} - \dot{r}_{\mathrm{tt},2}\ddot{r}_{\mathrm{tt},1}}{(\dot{r}_{\mathrm{tt},1}^2 + \dot{r}_{\mathrm{tt},2}^2)}\right)^\top$.

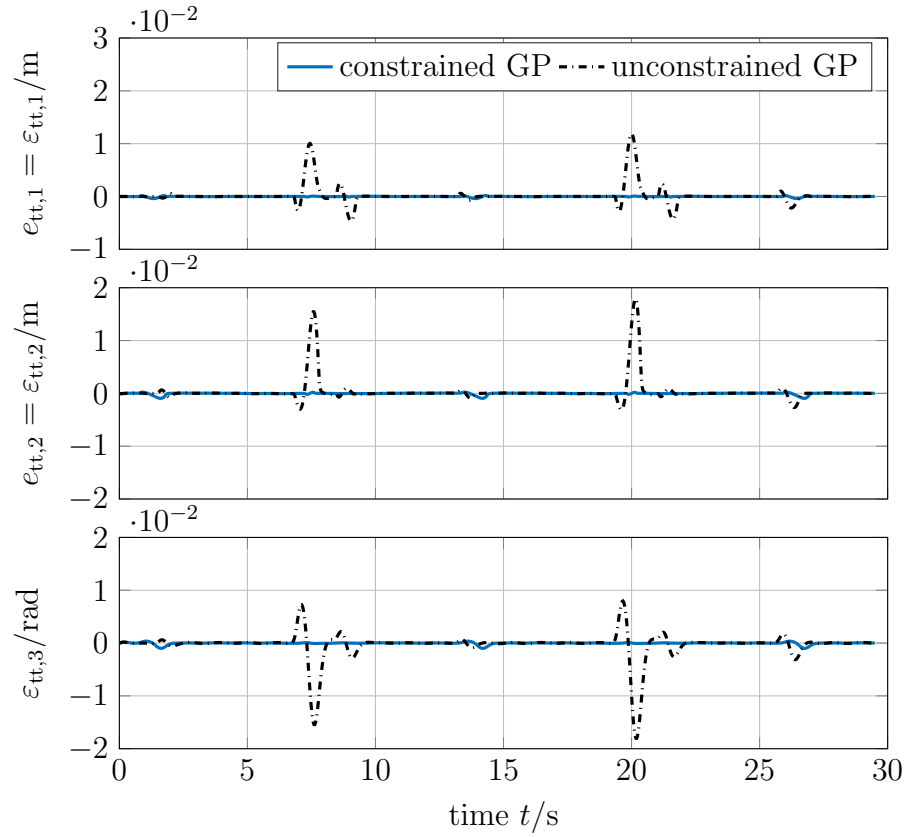


Figure 5.24: Output and state control errors of the same model predictive controller for two different references.

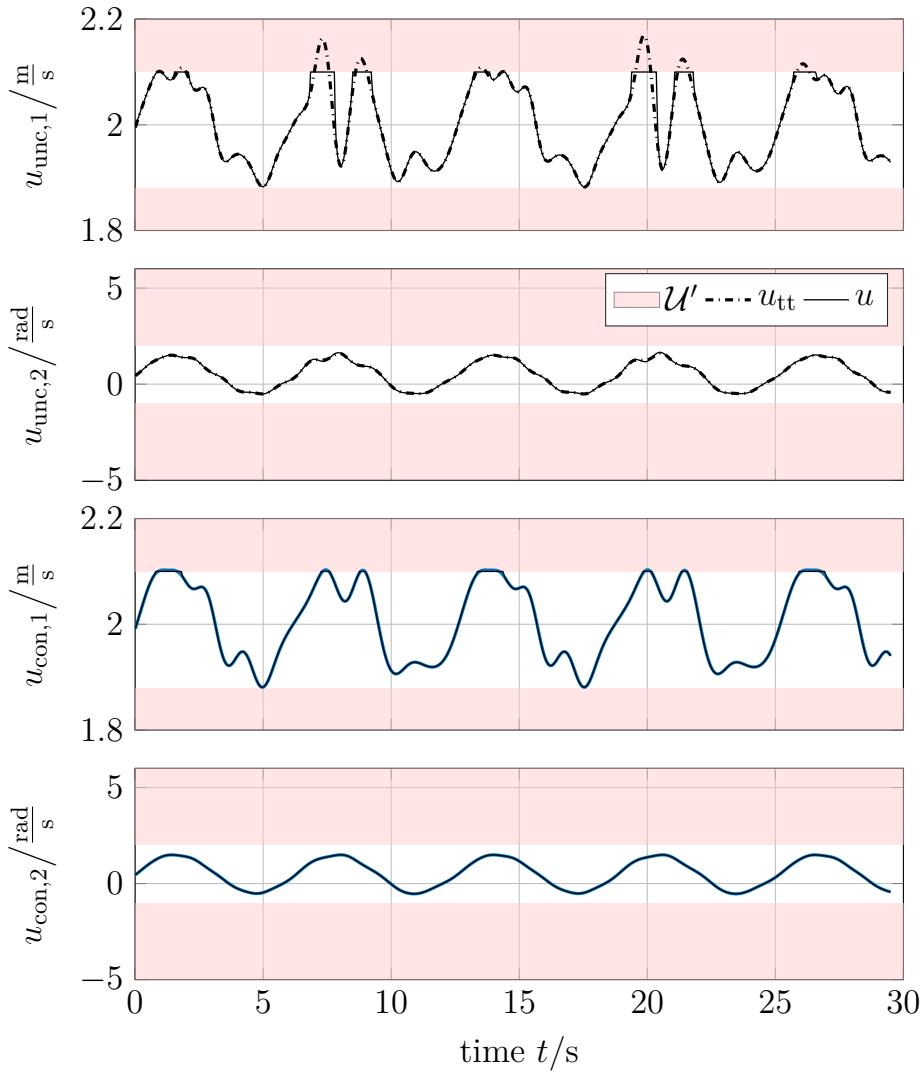


Figure 5.25: The chosen control inputs u (thin black solid) and the input references u_{tt} for the constrained (blue solid) and unconstrained GP (black dashdotted).

the reference in the constrained GP learning case enables precise tracking, the reference obtained via unconstrained learning is not exactly trackable. The latter results in tracking errors which are up to 77 times bigger than in the constrained learning GP reference, cf. Figure 5.24. These errors occur as the MPC saturates the control input to its feasible region, see Figure 5.25. Exact tracking of the unconstrained learned references is impossible despite the satisfaction of the state and output constraints since it does not satisfy the input constraints. In contrast to this, the constrained learning ensures trackability.

Example 3. *We aim to show the benefit of constrained learning via training of multiple GPs for systems with output size larger one. To this end, the planar robot of Section 2.3.4 and Figure 2.1 should perform a continuous circular motion in Cartesian space as required, for instance, in polishing tasks. The planar Cartesian reference is a circle with a radius of* 20 cm *centered at* $\mathrm{x} = \mathrm{y} = 0.4\,\mathrm{m}$. *This reference is not*

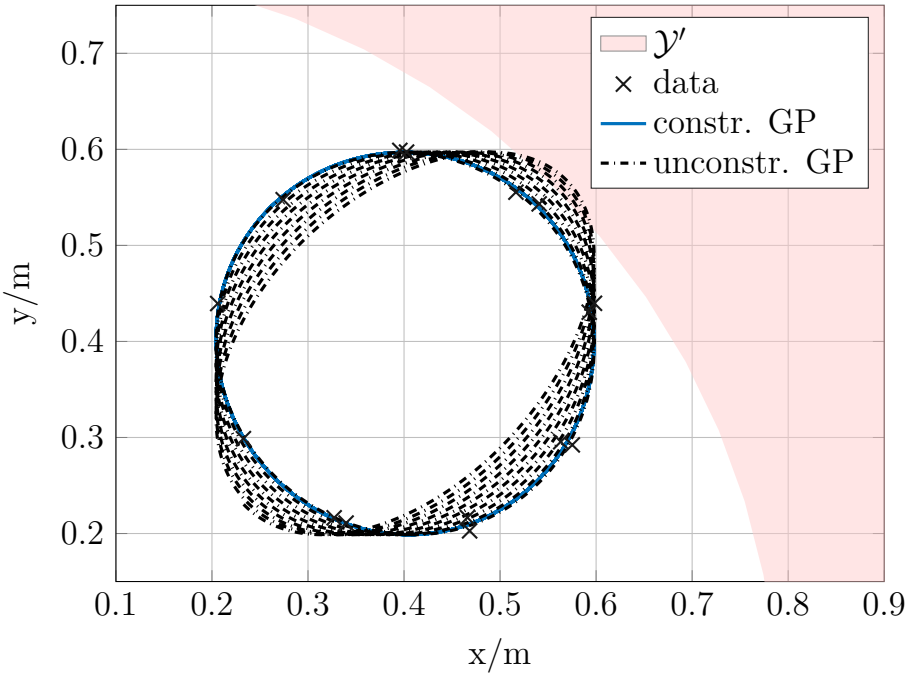


Figure 5.26: Cartesian workspace of the planar two degree of freedom robot with a circular reference, which should be learned.

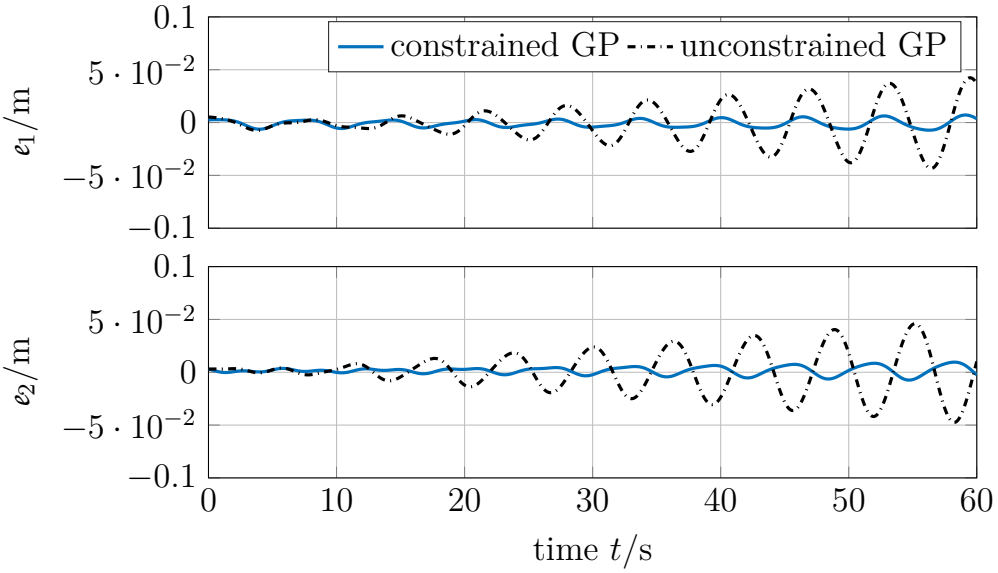


Figure 5.27: GP approximation error for the constrained and the unconstrained case.

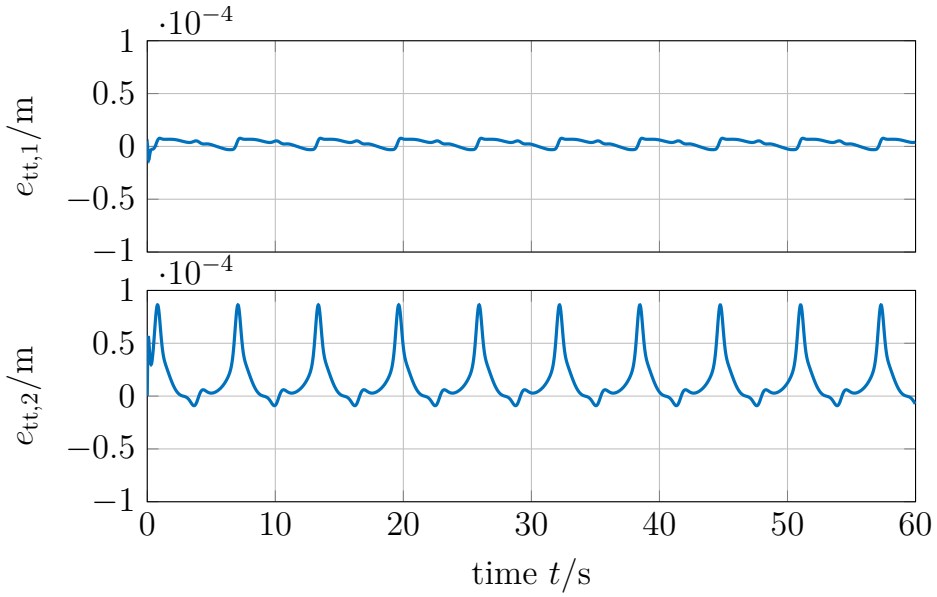


Figure 5.28: Output control errors of the model predictive controller for the learned reference in the constrained GP case.

known analytically and should be learned. It is represented by the data set D_ϕ with a noise standard deviation of $\sigma = 5\,\mathrm{mm}$ depicted as black crosses in Figure 5.26. When two GPs are learned independently from each other to capture the two dimensional motion they might identify slightly different period lengths for each dimension. This case is shown in Figure 5.26 as black dashdotted line, where the estimated hyperparameters are $\hat{\sigma}_{\mathrm{per},1} = 2.11, \hat{\sigma}_{\mathrm{per},2} = 2.09, \hat{l}_{\mathrm{per},1} = 21.32, \hat{l}_{\mathrm{per},2} = 20.79, \hat{T}_{\mathrm{per},1} = 6.26\,\mathrm{s}$, and $\hat{T}_{\mathrm{per},2} = 6.31\,\mathrm{s}$. Even though the approximation quality in Cartesian space in the beginning is good, the learned Cartesian reference positions for the robot end-effector will change over time as $\hat{T}_{\mathrm{per},1} \neq \hat{T}_{\mathrm{per},2}$. In contrast to this, the proposed constrained learning algorithm couples the learned output references via the constraints and the shared hyperparameters $T_{\mathrm{per},1} = T_{\mathrm{per},2} = T_{\mathrm{per}}$. This way, the same Cartesian position reference is predicted for each circular round, see Figure 5.26, blue solid line for which $\hat{\sigma}_{\mathrm{per},1} = 2.23, \hat{\sigma}_{\mathrm{per},2} = 2.16, \hat{l}_{\mathrm{per},1} = 22.48, \hat{l}_{\mathrm{per},2} = 21.59$, and $\hat{T}_{\mathrm{per},1} = 6.28\,\mathrm{s}$. The corresponding approximation errors of the constrained and unconstrained GP are depicted in Figure 5.27. Moreover, the missing synchronisation in the unconstrained case leads to a constraint violation in the output space, cf. Figure 5.26. In this case, the output constraint set $\mathcal{Y}$, indicated via its complement by the red shaded area in Figure 5.26, restricts the reference to the physically reachable workspace of the robot. The mappings from the flat output to the states and inputs do not exist when this constrained is violated. Since the references x_{tt} and u_{tt} cannot be calculated for the unconstrained case, it is impossible to set up an MPC controller as outlined in Section 2.3 for this case. Rather, only the constrained GP learning allows designing an MPC appropriately. The MPC control errors in the output space for the tracking of the constrained GP reference are depicted in Figure 5.28. The magnitude of the control errors lies in

the submillimeter range. This precision affirms the trackability of the reference learned via constrained optimisation.

5.4.3 Learning of Periodic References with Online Data Update

So far, a learning procedure for periodical references is proposed that guarantees constraint satisfaction and trackability for all times. Therein, the hyperparameters are learned offline, and the training data is not updated. These assumptions entail a severe drawback which is outlined below. Assume the hyperparameter optimisation led to an estimated period length of $T_{\mathrm{per}} \approx T_{\mathrm{p}}$ with an estimation error $e_T = T_{\mathrm{per}} - T_{\mathrm{p}}$, where T_{p} is the true period length of the reference. For large times (and consequently large distance to the training data) this error will add up to a significant phase shift. This shift does not affect the trackability of the reference but degrades its approximation quality. An update of the prediction training data D_{pred} during run time (*online*) eliminates this issue. However, increasing the number of training data points $n_{D_{\mathrm{pred}}}$ leads to an increased computational load and is consequently undesirable. Therefore we propose an update strategy which keeps the prediction data set size $n_{D_{\mathrm{pred}}}$ constant. Every time a new data point $(t_{\mathrm{new}}, r_{\mathrm{new}})$ is included in the set, another (old) data point is excluded. A common approach to do so are shifted horizons or windows of past measurements similar to moving horizon estimation [117, 189]. Windowing for periodic references was, for instance, considered in [157] to eliminate a potential phase shift. We extend this idea to include a distance measure $\measuredangle : \mathbb{R}^{n_{\mathrm{y}}} \times \mathbb{R}^{n_{\mathrm{y}}} \to \mathbb{R}_0^+$ in the update strategy to evaluate distances or similarities between the data points. The map $\measuredangle$ can hereby be chosen to encode a desired distance measure, as e.g. Euclidean, Chebychev, or more general Minkowski distances. This distance is calculated between the new reference data r_{new} and all components $\gamma_{\mathrm{pred},i}$ of the prediction data set D_{pred}. To this end the index set

$$\mathcal{I} := \left\{ i \in \{1, 2, \ldots, n_{D_{\mathrm{pred}}}\} \middle| \measuredangle(r_{\mathrm{new}}, \gamma_{\mathrm{pred},i}) < \bar{d} \right\} \tag{5.24}$$

contains all components for which the distance measure fulfils a certain threshold $\bar{d} \in \mathbb{R}_0^+$. The oldest (and potentially most outdated) data point from $\mathcal{I}$ is excluded from the training data set and replaced by the new one. This data update is described in Algorithm 2.

Algorithm 2 Data Update

1: **Input** training data set D_{pred}, new data point $(t_{\mathrm{new}}, r_{\mathrm{new}})$, distance threshold $\bar{d}$
2: find index set $\mathcal{I}$ of training data points close to new data via (5.24)
3: find oldest among the close points $\hat{i} = \arg\min_{i \in \mathcal{I}} \chi_{\mathrm{pred},i}$
4: obtain $D_{\mathrm{pred,up}}$ by replacing $(\chi_{\hat{i}}, \gamma_{\hat{i}})$ by $(t_{\mathrm{new}}, r_{\mathrm{new}})$ in D_{pred}
5: **return** $D_{\mathrm{pred,up}}$

The update criterion reduces data lumping in specific areas compared to windowing, especially if new data is not available at equidistant time instances. Therefore, situations are prevented where data updates reduce the approximation quality due to imbalanced spatial distribution of the training data. A theoretical analysis of the effect of data distribution on closed-loop performances is performed in [127]. Furthermore, an event-triggered update of data with a safe forgetting scheme, as proposed in [224], could be considered in future work.

To guarantee that $n_{D_{\text{pred}}}$ is constant, the following assumption on $\bar{d}$ is used:

Assumption 24. *The threshold $\bar{d}$ is chosen such that the index set $\mathcal{I}$ is non-empty for all times.*

Instead of Assumption 24, one could also use windowing (excluding the oldest data point) as a backup if $\mathcal{I}$ is empty. If a large threshold is chosen, more possible data points for exclusion are compared, and their actuality becomes more important then their location. Consequently, the threshold $\bar{d}$ can be used to find a trade-off between the importance of actuality and location. Note that in the limit of an infinite threshold $\bar{d} \to \infty$ this strategy is equal to windowing. Overall, the combination of location and actuality information for online updates allows for improvements in the prediction accuracy of the GP.

However, such updates of the data set D_{pred} can lead to a loss of trackability guarantees. Thus, an additional updating criterion is included, which prevents from losing these guarantees. Trackability loss can occur due to two circumstances: Either the predicted reference with the updated training data $D_{\text{pred,up}}$ is not trackable (it cannot be followed when starting on it), or the difference between the current reference and the updated one is so large that the transition leads to the trackability loss. To lower the chances for the second case, the transient between the previous prediction and the new one is designed based on a smooth blending. This blending at time t_k is realised via a $\mathcal{C}^\beta$ sigmoid function $\alpha : \mathbb{R} \to [0, 1]$ such that

$$\tilde{r}_{\text{tt}}(t) := m^+\left(t; D_{\text{pred}}, \hat{\phi}\right)(1 - \alpha(\Delta t)) + m^+\left(t; D_{\text{pred,up}}, \hat{\phi}\right)\alpha(\Delta t) \tag{5.25}$$

with $t \geq t_k$ and $\Delta t = t - t_k - \frac{T_\alpha}{2}$. Here, the sigmoid function converges to one and is assumed to be equal to one from time T_α on, see also Figure 5.29. Only if the prediction $\tilde{r}_{\text{tt}}(t)$ with the updated training data $D_{\text{pred,up}}$ and the blending α is trackable for $t \in [t_k, t_k + \bar{\tau}]$ with $\bar{\tau} \geq T_{\text{per}} + T_\alpha$, the prediction is used, i.e. $r_{\text{tt}}(t) = \tilde{r}_{\text{tt}}(t)$ for all $t \geq t_k$. Otherwise, the data D_{pred} is not updated and a prediction based on it is performed as before. This allows for example to exclude outliers or data points which lead to singularities in the covariance matrices. This way, the GP prediction of the learned reference is robustly trackable despite an online prediction training data update.

The proposed update scheme is described in detail in Algorithm 3. To check the trackability, the time derivatives of (5.25) up to degree β are needed which can be

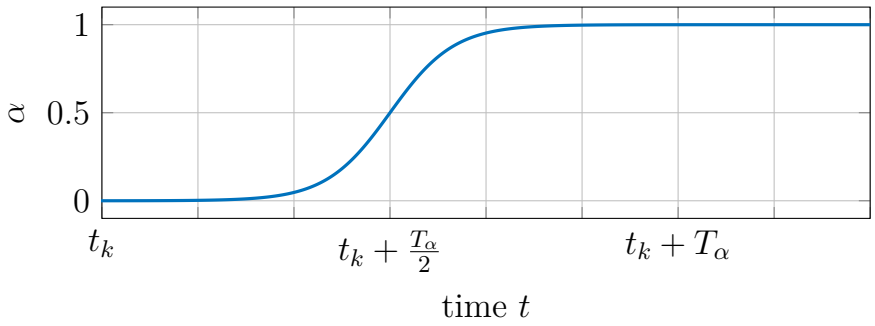


Figure 5.29: Sigmoid function α used for smooth blending.

Algorithm 3 GP online learning and prediction for periodical references

Init $k = k_0, \bar{t}, \mathcal{Y}, \mathcal{X}, \mathcal{U}, D_\phi, T_\alpha, T$
obtain $\hat{\phi}$ via (5.19) using $\left(\bar{t}, \mathcal{Y}, \mathcal{X}, \mathcal{U}, D_\phi\right)$
choose $\bar{\tau}$ such that $\bar{\tau} \geq T_\alpha + H\hat{\phi}$
initialise $D_{\text{pred}} = D_\phi$
while true **do**
    if new data $(t_{\text{new}}, r_{\text{new}})$ available **then**
        obtain $D_{\text{pred,up}}$ by Algorithm 2
        $\tau \leftarrow [t_k, t_k + \bar{\tau}]$
        calculate $\tilde{r}_{\text{tt}}, \dot{\tilde{r}}_{\text{tt}} \ldots, \tilde{r}^{(\beta)}$ via (5.25) and (5.26) for times τ
        if $\tilde{r}_{\text{tt}} \in \mathcal{Y}$ **then**
            determine $\tilde{x}_{\text{tt}} = \Phi(\tilde{r}_{\text{tt}}, \dot{\tilde{r}}_{\text{tt}}, \ldots, \tilde{r}_{\text{tt}}^{(\beta-1)})$ and $\tilde{u}_{\text{tt}} = \Psi(\tilde{r}_{\text{tt}}, \dot{\tilde{r}}_{\text{tt}}, \ldots, \tilde{r}_{\text{tt}}^{(\beta)})$
            if $\tilde{x}_{\text{tt}} \in \mathcal{X}$ and $\tilde{u}_{\text{tt}} \in \mathcal{U}$ **then**
                $t \leftarrow [t_k, t_k + T]$
                calculate $r_{\text{tt}}, \dot{r}_{\text{tt}} \ldots, r^{(\beta)}$ via (5.25) and (5.26) for times t
                compute $x_{\text{tt}} = \Phi(r_{\text{tt}}, \dot{r}_{\text{tt}}, \ldots, r_{\text{tt}}^{(\beta-1)})$ and $u_{\text{tt}} = \Psi(r_{\text{tt}}, \dot{r}_{\text{tt}}, \ldots, r_{\text{tt}}^{(\beta)})$
                $D_{\text{pred}} \leftarrow D_{\text{pred,up}}$
            else
                $t \leftarrow [t_k, t_k + T]$
                calculate $r_{\text{tt}} = m^+(t; D_{\text{pred}}, \hat{\phi}), \ldots, r_{\text{tt}}^{(\beta)} = m^{+,(\beta)}(t; D_{\text{pred}}, \hat{\phi})$
                compute $x_{\text{tt}} = \Phi(r_{\text{tt}}, \dot{r}_{\text{tt}}, \ldots, r_{\text{tt}}^{(\beta-1)})$ and $u_{\text{tt}} = \Psi(r_{\text{tt}}, \dot{r}_{\text{tt}}, \ldots, r_{\text{tt}}^{(\beta)})$
            end if
        else
            $t \leftarrow [t_k, t_k + T]$
            calculate $r_{\text{tt}} = m^+(t; D_{\text{pred}}, \hat{\phi}), \ldots r_{\text{tt}}^{(\beta)} = m^{+,(\beta)}(t; D_{\text{pred}}, \hat{\phi})$
            determine $x_{\text{tt}} = \Phi(r_{\text{tt}}, \dot{r}_{\text{tt}}, \ldots, r_{\text{tt}}^{(\beta-1)})$ and $u_{\text{tt}} = \Psi(r_{\text{tt}}, \dot{r}_{\text{tt}}, \ldots, r_{\text{tt}}^{(\beta)})$
        end if
    end if
    use learned reference $r_{\text{tt}}, x_{\text{tt}}, u_{\text{tt}}$ in a model predictive controller
    $k \leftarrow k + 1$
end while

obtained via the general Leibniz rule

$$\tilde{r}_{\mathrm{tt}}^{(\beta)} = \sum_{j=0}^{\beta} \binom{\beta}{j} m^{+,(j)}\left(t; D_{\mathrm{pred}}, \hat{\phi}\right) (1-\alpha(\Delta t))^{(\beta-j)} + \sum_{j=0}^{\beta} \binom{\beta}{j} m^{+,(j)}\left(t; D_{\mathrm{pred}}, \hat{\phi}\right) \alpha(\Delta t)^{(\beta-j)}, \tag{5.26}$$

where $\binom{\beta}{j}$ denotes binomial coefficients. Note that the optimisation (5.19) is only performed once, offline, whereas the training data $\mathcal{D}_{\mathrm{pred,up}}$ is updated online. This separation is beneficial in terms of calculation times, as the optimisation has higher computational demand then inferring the posterior distribution.

Lemma 3. *Consider a differentially flat system* (2.1) *and a Gaussian process with constant prior mean m and a periodic prior covariance function κ. A reference $r_{\mathrm{tt}}(t)$ provided by the posterior mean* (4.13) *of the GP trained with* (5.19) *under the Assumption 24 and with $\bar{\tau} \geq T_{\mathrm{per}} + T_\alpha$ is trackable for all times $t \geq 0$ in the sense of Definition 1 for system* (2.1) *despite online training data update of the form described in Algorithm 3.*

Proof. Problem (5.19) guarantees for all $t \in \left[0, \bar{t}\right]$ that $r_{\mathrm{tt}}(t) = m^+\left(t|D_\phi, \hat{\phi}\right) \in \mathcal{Y}$, $x_{\mathrm{tt}}(t) = \Phi\left(m^+\left(t; D_\phi, \hat{\phi}\right), \dot{m}^+\left(t; D_\phi, \hat{\phi}\right), \ldots, m^{+(\beta-1)}\left(t; D_\phi, \hat{\phi}\right)\right) \in \mathcal{X}$ and $\tilde{u}_{\mathrm{tt}}(t) = \Psi\left(m^+\left(t; D_\phi, \hat{\phi}\right), \dot{m}^+\left(t; D_\phi, \hat{\phi}\right), \ldots, m^{+(\beta)}\left(t; D_\phi, \hat{\phi}\right)\right) \in \mathcal{U}$. From the constant prior mean $m(t) = \mu$, the periodic prior covariance κ, and due to $\bar{t} \geq T_{\mathrm{per}}$ it follows that $r_{\mathrm{tt}}(t) = r_{\mathrm{tt}}(t + T_{\mathrm{per}}), x_{\mathrm{tt}}(t) = x_{\mathrm{tt}}(t + T_{\mathrm{per}})$, and $u_{\mathrm{tt}}(t) = u_{\mathrm{tt}}(t + T_{\mathrm{per}})$ for all $t \in \mathbb{R}_0^+$. Consequently, the prediction based on the set D_ϕ guarantees $r_{\mathrm{tt}}(t) \in \mathcal{Y}, x_{\mathrm{tt}}(t) \in \mathcal{X}$ and $u_{\mathrm{tt}}(t) \in \mathcal{U}$ for all $t \in \mathbb{R}_0^+$.

The data set D_{pred} is initialised with $D_{\mathrm{pred}} = D_\phi$ and an update of D_{pred} by $D_{\mathrm{pred,up}}$ at time t_k is only performed when $\tilde{r}_{\mathrm{tt}}(t) \in \mathcal{Y}, \tilde{x}_{\mathrm{tt}}(t) \in \mathcal{X}$, and $\tilde{u}_{\mathrm{tt}}(t) \in \mathcal{U}$ for $t \in [t_k, t_k + \bar{\tau}]$. If this is fulfilled, $\tilde{r}_{\mathrm{tt}}, \tilde{x}_{\mathrm{tt}}$, and $\tilde{u}_{\mathrm{tt}}$ are used as $r_{\mathrm{tt}}, x_{\mathrm{tt}}$, and u_{tt} from t_k on. This way, trackability at least up to $t_k + T_{\mathrm{per}} + T_\alpha$ is guaranteed. As $\bar{\tau} \geq T_{\mathrm{per}} + T_\alpha$ it covers both the smooth transient from the old to the new reference as well as at least one period of periodic predictions of the new reference. Since we assume that $\alpha(\Delta t) = 1$ for $t \geq t_k + T_\alpha$ we can conclude that $r_{\mathrm{tt}}(t) = r_{\mathrm{tt}}(t + T_{\mathrm{per}}) \in \mathcal{Y}, x_{\mathrm{tt}}(t) = x_{\mathrm{tt}}(t + T_{\mathrm{per}}) \in \mathcal{X}$, and $u_{\mathrm{tt}}(t) = u_{\mathrm{tt}}(t + T_{\mathrm{per}}) \in \mathcal{U}$ for all $t \geq t_k + T_\alpha$ at each time t_k even if the update is performed. Consequently, trackability despite online training update is guaranteed for all times $t \geq 0$. □

Example 4. *The planar robotic manipulator, cf. Figure 2.1 and Example 3, is used to perform periodic motions while synchronising this motion to a second manipulator. A constrained hyperparameter optimisation is used to learn the Cartesian motion which is depicted in Figure 5.30. The hyperparameter optimisation is performed on the training data depicted in black crosses. The number of data points is $n_{D_\phi} = 50$ and*

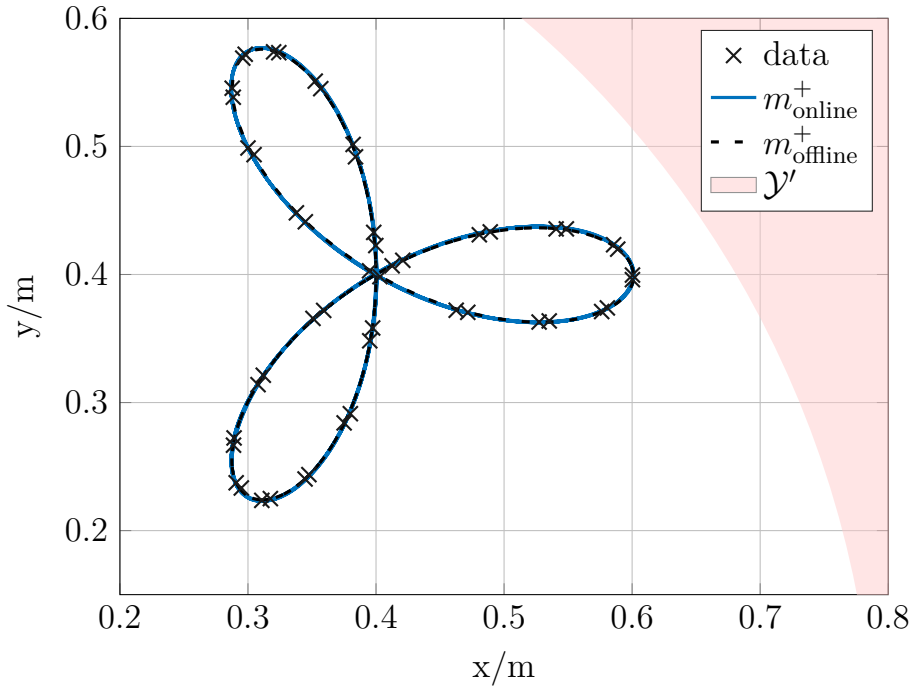


Figure 5.30: Learned reference for the end-effector position in Cartesian space.

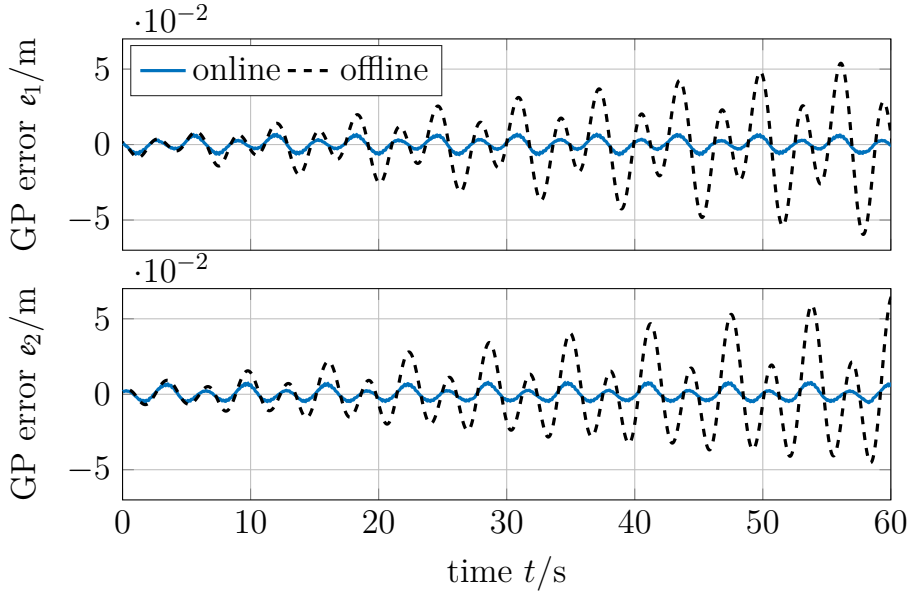


Figure 5.31: GP approximation error with and without online prediction training data update.

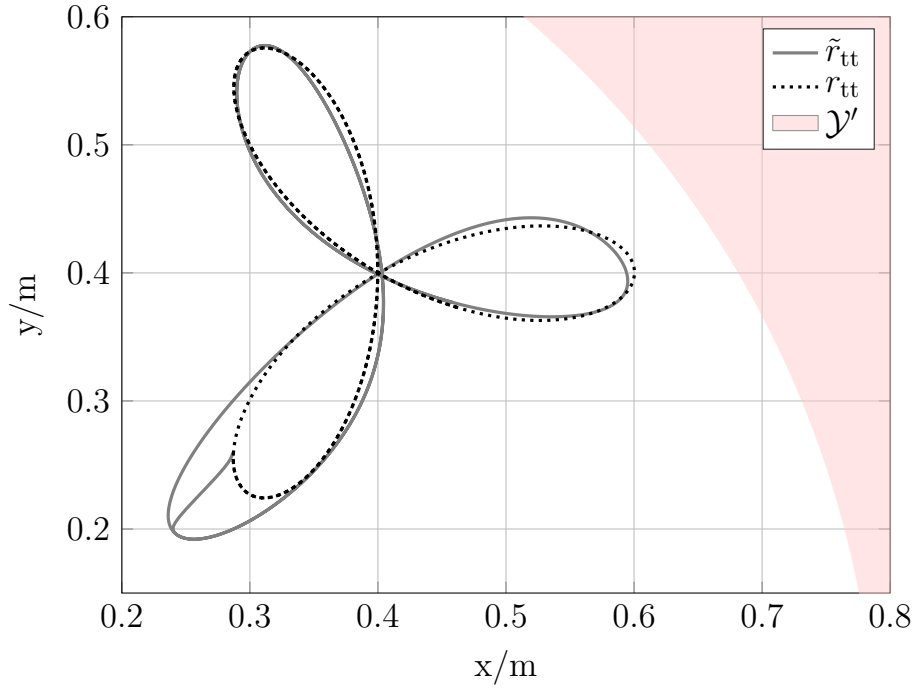


Figure 5.32: Cartesian reference prediction $\tilde{r}_{\mathrm{tt}}$ with an outlier in the data $D_{\mathrm{pred,up}}$ as well as a prediction of r_{tt} based on D_{pred} which excludes the outlier.

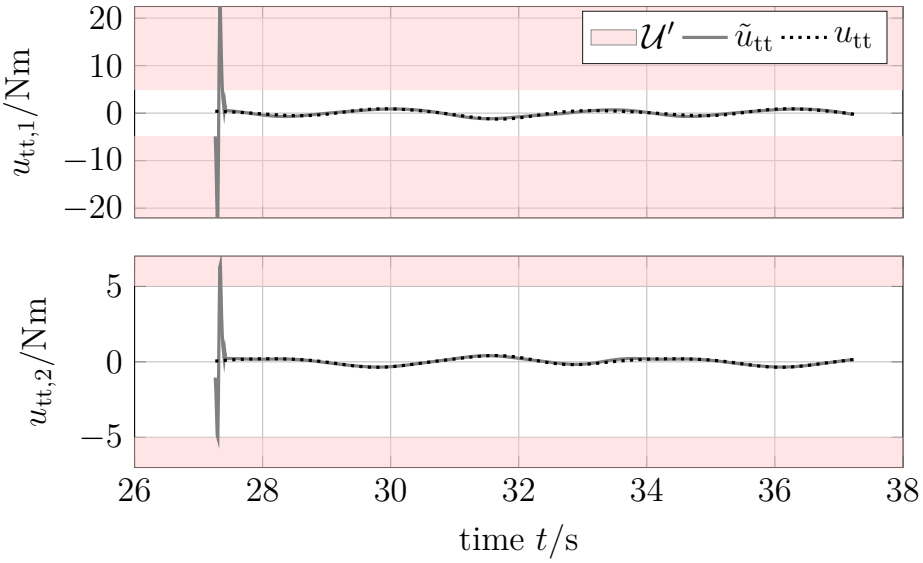


Figure 5.33: Learned reference input $\tilde{u}_{\mathrm{tt}}$ for the references $\tilde{r}_{\mathrm{tt}}$ with outlier in $D_{\mathrm{pred,up}}$ and reference input u_{tt} for r_{tt} without outlier in data D_{pred}.

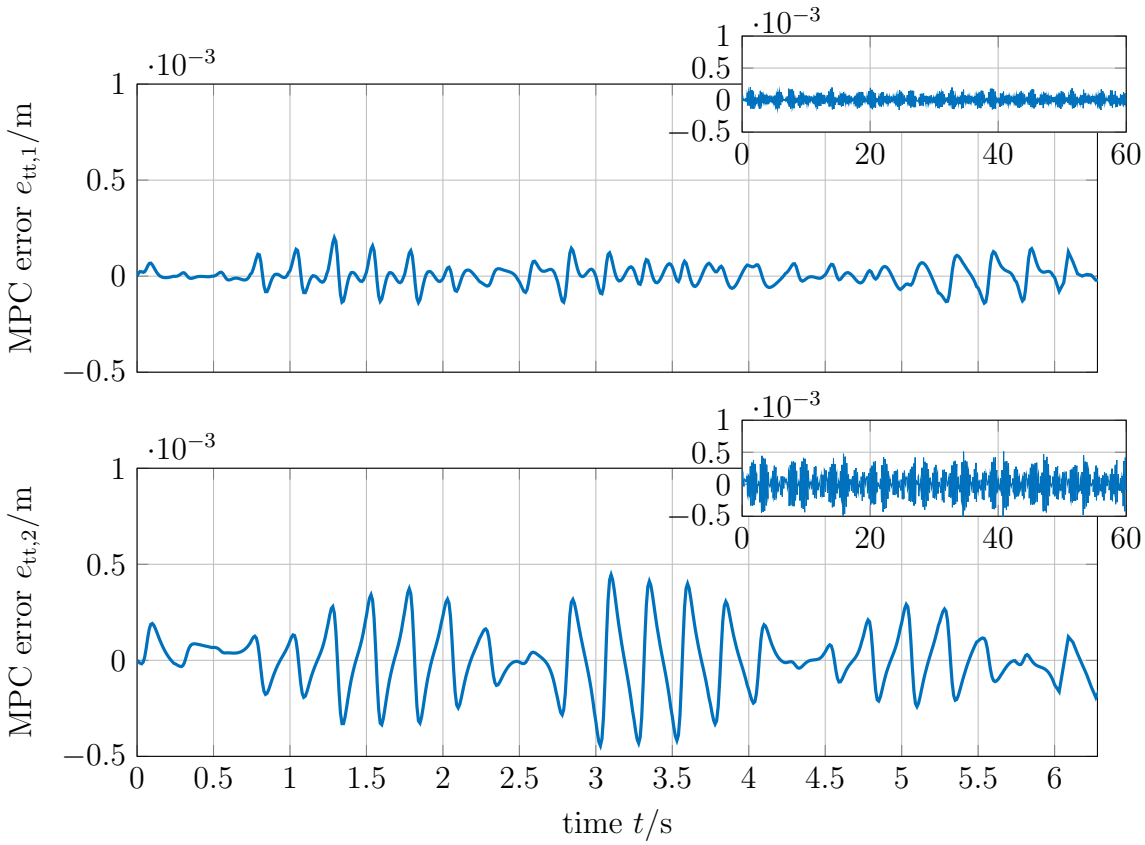


Figure 5.34: MPC output control errors to follow a learned reference with online data update. The small axes show the error for 60 s while the larger figures show the first period of this motion in more detail.

the standard deviation of the noise in the data is $\sigma = 0.5\,\mathrm{mm}$. *For comparison, two GP predictions with the same hyperparameters* $\hat{\phi} = (\hat{\sigma}_{\mathrm{per},1}, \hat{\sigma}_{\mathrm{per},2}, \hat{l}_{\mathrm{per},1}, \hat{l}_{\mathrm{per},2}, \hat{T}_{\mathrm{per}}) = (1.11, 1.77, 4.01, 5.00, 6.30\,\mathrm{s})$ *are depicted in Figure 5.30. One GP prediction uses the online data update described in Algorithm 3 (depicted as a solid blue line), while the other (depicted as a black dashed line) is based purely on the offline data (black crosses). As a distance measure the Euclidean distance* $d = \sqrt{y_1^2 + y_2^2}$ *with the threshold* $\bar{d} = 5\,\mathrm{cm}$ *is used in the online update. Both the offline and online case show good approximation quality in the Cartesian space. However, in both cases, the period length* $T_\mathrm{p} = 2\pi\,\mathrm{s}$ *of the motion was overestimated. The prediction errors of the Cartesian positions over time are shown in Figure 5.31. While the online update allows compensating the estimation error, the prediction error increases over time for the offline learning due to a phase shift, cf. Figure 5.31, black dashed line. As a result, the online update allows for a significant increase in the approximation quality. Still, it allows for guaranteed trackability due to the smooth blending and the included update criteria. At time* $t_k = 27.26\,\mathrm{s}$ *an outlier in the online captured data occurs such that* $t_\mathrm{new} = t_k, r_\mathrm{new} = (0\,\mathrm{m}, 0\,\mathrm{m})^\top$. *This situation can appear, for example, if data loss during communication or a malfunction of a sensor occurs. This outlier would lead to a significant performance loss. In Figure 5.32 the prediction* $\tilde{r}_{tt}$ *starting at* t_k *and*

based on $D_{pred,up}$ including the outlier is shown as gray solid line. Starting from around $y_1 = \mathrm{x} = 0.29\,\mathrm{m}$ and $y_2 = \mathrm{y} = 0.26\,\mathrm{m}$ the blending over to the new prediction can be seen as slightly s-shaped curve. Even though no output constraints are violated, the gray curve is not trackable for the system as the demanded inputs to follow it exceed the input constraints, cf. Figure 5.33. Therefore this data point is not included in D_{pred}. The prediction on this set without the new data point is shown in Figure 5.32 as a black dotted line. This restricted update maintains trackability of the reference as well as prevents from prediction errors due to outliers. Consequently, it can be used in conjunction with a model predictive controller allowing for guaranteed recursive feasibility and potentially small control errors, as shown in Figure 5.34. Larger control errors occur if the change in the reference due to an updated data set is more dominant. Still, all updates in the reference allow for trackability and result in submillimeter control errors.

5.5 Summary

In this chapter, we proposed different ways to support model predictive control with machine learning. Gaussian processes were used to learn the output model of an MPC while guaranteeing nominal stability as well as including chance constraints. This learning-supported model predictive control scheme has been applied to a robotic example in simulation and experiment, where the interaction of the robot with its environment is learned and controlled. The hybrid GP model reduces the modelling error by more than 58 % compared to the first principles models. The closed-loop simulations show the superior average performance of the learning-supported MPC over the first principle model predictive controllers. In the adaptive case, where the influence of the model-plant mismatch is reduced as much as possible, the learning-supported force controller still shows 50 % smaller errors than the controller with linear force model. At the same time, it requires less insight and manual design decisions than nonlinear first-principle models and performs at least equally well. We accomplished a real-time feasible experimental validation that underlined these simulation results.

Furthermore, Gaussian processes have been used to learn external reference signals that take the system dynamics and constraints into account. The developed concept allows for direct use of the GP predictions inside a model predictive controller. Algorithms for constrained GP hyperparameter learning have been proposed to ensure the trackability of the learned references. As a consequence, we guarantee the recursive feasibility of the controller, which is a fundamental property in MPC from the theoretical and practical perspective. Various simulation examples were conducted to illustrate the applicability as well as the benefits of the proposed algorithms. It was shown, that the proposed constrained learning algorithms increase the reliability and performance of the GP prediction models compared to classical unconstrained learning.

All in all, this chapter outlines different possibilities on how to merge machine learning and model predictive control safely and reliably. Benefits from both disciplines such as easy adaptation, handling of noisy measurements, explicit consideration of constraints and closed-loop guarantees have been merged in an interactive way. Consequently, the proposed algorithms and achieved results build an important step toward the autonomy of dynamical systems.

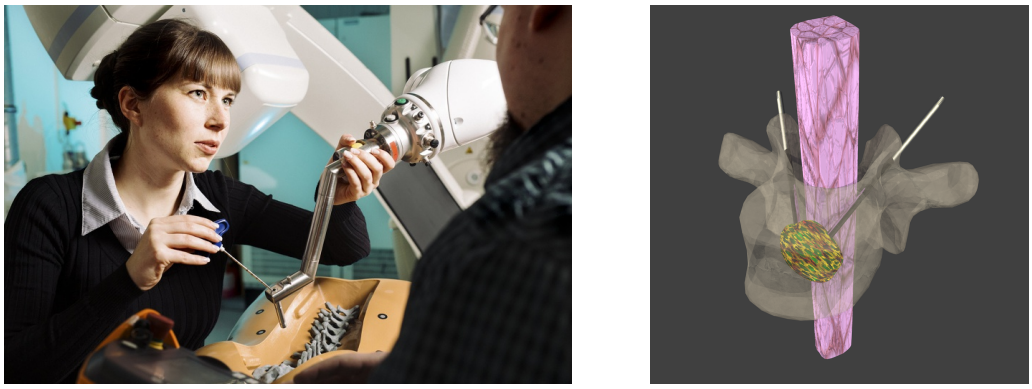
6 Respiratory Motion Compensation

As an application example for the presented approaches, robot-supported interventions in the spine, such as radio frequency ablation, are investigated, cf. Figure 6.1. The robot should assist the physician in the precise positioning of surgical instruments while compensating for the patient's movements. We propose to use Gaussian processes to learn and predict these motions. The predicted motion signals serve as references for a model predictive controller that proactively controls the robotic system to track the motions.

6.1 Radio Frequency Ablation

We consider radiofrequency ablation in the spine [43, 74, 239]. The goal is to destroy a tumour or metastasis, which afflicts the vertebral bodies of the spine, via heating of the malignant cells, cf. Figure 6.1. A high-frequency alternating current induces heat between electrodes, which must be inserted in the target area. For a successful treatment, the control of the electrical energy and the positioning of the needle-shaped electrodes is of significant importance. The former can be achieved via preoperative modelling of the expected temperature distributions over time in the area of interest for varying electrode activations and placements [5, 191, 207]. This therapy planning involves the modelling of the tissue temperature distribution via partial differential equations. In spinal applications, spatially varying tissue properties such as electrical and thermal conductivity of the involved tissue types such as bone, muscular or soft tissue must be incorporated [155, 159, 242]. This planning phase results in an optimal operating mode and needle position, where the necrosis zone covers the metastasis while minimising harm for surrounding healthy tissue. Besides the sensitivity of the temperature profile and distribution to mis-estimated tissue properties, it is also sensitive to the needle position. Therefore, the electrodes must be precisely positioned for successful treatment. Moreover, precision in needle placement is essential to minimise harm to surrounding structures by unwanted mechanical penetration. Especially in the spine, precise insertion is crucial to avoid paralysis or other harm to the nervous system in the spinal cord. For this chapter, we consider the transpedicular insertion, as illustrated in Figure 6.1.

To meet the high positioning accuracy demands, one can support the physicians in the electrode placement with a robotic system. Various robotic assistant systems have been proposed for spinal interventions and are already in clinical use [17, 37, 47, 129, 195]. Most of these systems require a rigid attachment to the spine or represent



(a) Robot-assisted insertion of a trocar.

(b) Isometric view of inserted electrodes in vertebral body.

Figure 6.1: Illustration of transpedicular electrode placement for a radio frequency ablation in the spine.

highly specified and not very generic solutions. Instead of considering a bone mounted setup or a parallel manipulator, we rely on a versatile robotic arm registered to the patient, similar to the setup used in [87, 157], cf. Figure 6.1a. This setup requires the synchronisation of the robot motion to the movements of the patient to avoid harming the patient. These movements can, for instance, occur due to respiration [132, 144].

6.2 Motion Capturing, Modelling, and Prediction

The movements of the patient can be measured during an intervention via optical tracking systems [129, 216]. If this tracking data is used directly as a reference for the robot, precise tracking of the actual motion is often not achieved. The deviation originates from an inherent delay in the acquired tracking data due to data processing. Moreover, the tracking data can be corrupted with noise and might not even be available for some times, for example when the line of sight is interrupted by the surgeon. These issues are not only related to percutaneous interventions of the spine but also laparoscopic procedures, cardiac interventions or radiotherapy [194, 204]. Hence, various prediction filters are proposed in the literature to compensate for the motion, which often focus on delay compensation in a feed-forward manner. For example, Kalman filters estimate amplitudes, phases and frequencies of multiple augmented sinusoidal signals representing the motion [188]. Other model-based approaches use autoregressive models with moving average (ARMA) and adjust the involved weights based on the data via recursive least squares [50]. A weighted frequency linear combiner was used in [132] and [193] for motion prediction in spinal and urological procedures, respectively. Learning based approaches span from neural networks [142] over support vector regression [51] to Gaussian processes [44–46]. Several review studies have been reported to compare the different approaches [50, 52, 107]. Among other findings, it was shown that the motion compensation algorithm included in available robotic

systems in radiotherapy (an ARMA model with weight adjustment via normalised least mean squares) shows significant space for improvement when compared to the aforementioned algorithms [50]. In [52] it was observed that an extended Kalman filter was not outperforming model-free prediction algorithm, but led to a smoother prediction. Hence, a model-free, stochastic learning algorithm such as Gaussian processes appears promising since it combines the benefits from stochastic Kalman filters and model-free approaches. In [107] it has been concluded that noise in the data has a significant effect on the quality of the predictions. Therefore, Gaussian processes are an appealing strategy for motion prediction as they can directly handle the noise in the captured data.

In [24] a combination of a Kalman filter and a Gaussian process is used for motion prediction and extended in [25] for higher dimensions. The Kalman filter is used for prediction, while the GP corrects the prediction. Furthermore, the GP evaluates the model quality via the posterior variance. Instead of using a squared exponential covariance function and outsourcing the prediction into a Kalman filter as done in [24, 25], we directly tackle the overall task with a Gaussian process capable of prediction, modelling, and filtering of the motions. In [54] a GP is used in combination with a relevance vector machine. The GP corrects the predictions of the relevance vector machine and finds correlations between multiple output dimensions. In our setup, the GP is used not only for correction purposes but covers the whole prediction. In our case, multiple GP outputs are coupled via the constrained hyperparameter optimisation, instead of taking the full cross-correlation matrix into account. This coupling is beneficial in terms of calculation times, especially for online prediction. In [44–46] Gaussian processes are used for motion prediction in radiotherapy, while using the GP to merge information from multiple optical marker positions. In our setup, spine motions for patients in the prone position are predicted rather than organ motions in the supine pose. Also, we do not use multiple optical markers and merge their information to a one-dimensional position but use one Moiré phase marker that provides three Cartesian positions and three orientations at once [222]. Again, we do not use all correlations between the GP outputs but encode their dependency in the constrained hyperparameter optimisation.

In contrast to all these existing approaches, our proposed learning algorithm includes constraints on the learned posterior. Furthermore, our approach updates data online in a safe manner, for example, excluding outliers. This update leads to a smoother prediction of the reference that is provably trackable by the robotic manipulator. Hence, only safe and plausible predictions are made.

6.3 Model Predictive Motion Compensation Control

In this example, model predictive control tracks the predicted reference. Instead of controlling the surgical couch as done in [89, 90, 181], we use MPC for control of the

robotic arm to eliminate relative motion between the goal structure and the surgical device. In contrast to [40], optical instead of haptic feedback is used. Moreover we tackle the reference prediction with the aformentioned machine learning algortihms instead of keeping the reference constant over the prediction horizon as in [40]. In [6], different controllers including repetitive, adaptive, feed-forward and model predictive controllers were compared. The feed forward and the model predictive controller showed superior performance. As we directly want to consider constraints, the choice of MPC as motion compensation control is justified. While in [71, 77] unconstrained MPC for motion compensation in teleoperated robots is considered, we include constraints in our setup to incorporate additional safety feature which are also linked to the GP reference predictor. Since safety is a dominant feature of our coupled motion prediction and compensation algorithm, it is suited for delicate control tasks such as the considered robot-supported spine interventions.

General Control Setup for Motion Compensation

The overall setup of the proposed motion compensation algorithm is shown in Figure 6.2. The preoperative intervention planning (Figure 6.2, top) provides a desired goal position and orientation of the needle $p_{\text{goal}}(t) \in \mathbb{R}^6$ relative to the patient which allows for optimal treatment. Based on this goal position, an entry path is planned for the robot joint coordinates. This path consists of the angular positions $q_{\text{path}}(t) \in \mathbb{R}^{n_{\text{q}}}$ and velocities $\dot{q}_{\text{path}}(t) \in \mathbb{R}^{n_{\text{q}}}$. The path planner allows obtaining the needle insertion path while taking into account access path limitations. These limitations should be considered in the planning as well as in the motion compensation control since the needle can not be inserted arbitrarily to reach the goal posture. Such a path planning in joint coordinates allows obtaining a collision-free motion in the operating suite. At the same time, optimal robot configurations in terms of robustness to external deflection can be determined. The third component of the offline phase is the hyperparameter optimisation for the Gaussian process motion model, cf. Figure 6.2. This optimisation is executed on captured motion data directly before the robot-assisted needle insertion is performed. This way, patient-specific respiratory motion models can be learned. The optimal hyperparameters are then used online in a GP motion prediction to obtain smooth values and predictions of the goal structure pose $p_{\text{GP}}(t) \in \mathbb{R}^6$ and its derivative $\dot{p}_{\text{GP}}(t) \in \mathbb{R}^6$. An inverse kinematics algorithm merges the information from the online GP motion prediction and the preoperatively planned insertion path. As a result, a trajectory of predicted angular positions $q_{\text{r}}(t) \in \mathbb{R}^{n_{\text{q}}}$ and velocities $\dot{q}_{\text{r}}(t) \in \mathbb{R}^{n_{\text{q}}}$ is obtained which reflects the superimposed Cartesian path. These predictions serve as a reference for the model predictive controller. The MPC, which is formulated in joint space, steers the robot to follow the references q_{r} and $\dot{q}_{\text{r}}$ in an optimal way while including constraints to ensure safety.

In the following, the used Gaussian process, the inverse kinematics algorithm, and

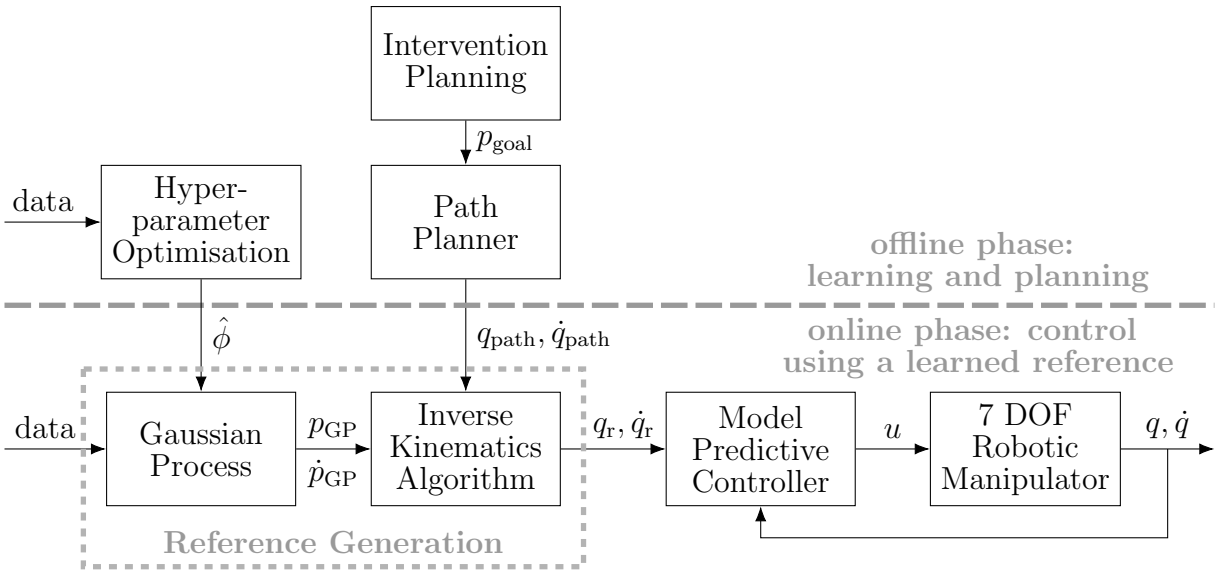


Figure 6.2: System setup including the predictive controller which uses a reference prediction provided by the GP.

the controller will be presented in more detail.

Gaussian Process

The Gaussian process predicts the motion of the goal structure in the spine based on interventional data. This data is obtained via a camera system and a marker placed on the skin of the patient. In our setup, we use a single camera and a Moiré phase marker, which provides six-dimensional data, reflecting the Cartesian position and orientation*. This data is shown in Figure 6.3. Here, x, y, and z denote the Cartesian position in the base frame, while A, B, and C denote the Tait-Bryan Euler angles around y, x, and z (intrinsic) representing the orientation. The camera does not provide the data in an equidistant grid, but whenever data processing is finished. The average sampling time of the camera is 15.6 ms, which is 3.9 times larger than the sampling time of the considered robot controller of 4 ms. The hyperparameter optimisation is performed on data of the first 15.57 seconds, which cover nearly four periods. This time is indicated by a grey vertical line in Figure 6.3. Every 15^{th} data point is used for the optimisation leading to $n_{D_\phi} = 67$. The training data is depicted in Figure 6.3 as black crosses. For each dimension, a single output GP is used, while the six Gaussian processes are coupled via the hyperparameters, such as a shared period length. The periodic covariance κ_{per} is used. The optimal hyperparameters obtained via (5.19) are listed in Table 6.1. The trained GP is evaluated on the data shown in Figure 6.3. This evaluation includes interpolation for the first 15.57 seconds and extrapolation thereafter. The performance is shown in Figure 6.3 in blue, while the

*The data was provided by our project partner the Fraunhofer Institute for Factory Operation and Automation IFF in the framework of the Forschungscampus *STIMULATE*.

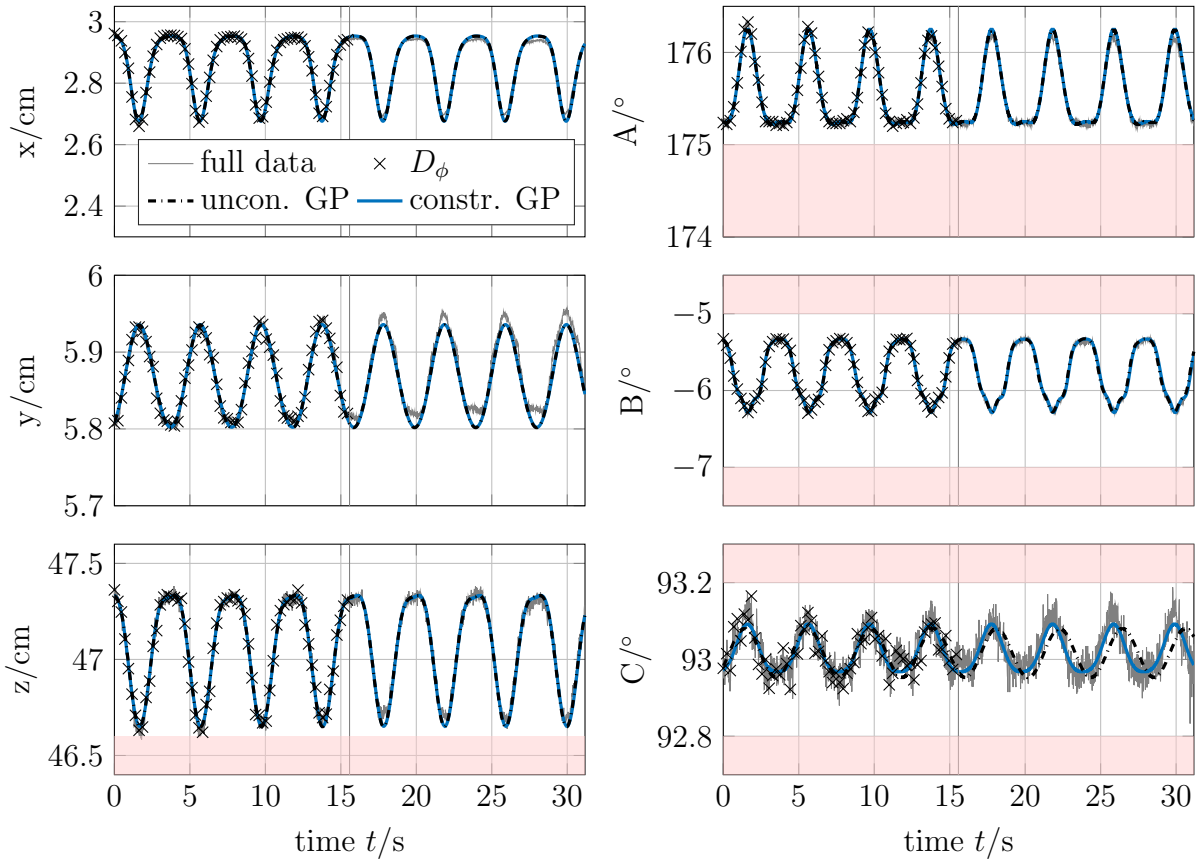


Figure 6.3: The training data D_ϕ (black crosses) is a subset of the captured motion data (grey solid). The GP predictions are based on a constrained (blue solid) and unconstrained (black dashdotted) hyperparameter optimisation on this data.

considered constraints are depicted as red shaded areas. These include constraints on the maximum insertion depth in the Cartesian z-direction as well as orientational constraints to cover the limitations in the access path through the pedicle of the vertebral body. These constraints are fulfilled and non-active during the optimisation, cf. Figure 6.3. As a comparison, six completely decoupled GPs are learned in an unconstrained manner. Their hyperparameters are also given in Table 6.1. In Figure 6.3 the unconstrained prediction is shown as a black dash-dotted line. Since the constraints are not active during learning of the constrained GP, both GPs perform similarly in most of the directions. However, the shared period length in the constrained case (see also Table 6.1 column 6) prevents from asynchronous prediction as in the unconstrained case. The unconstrained GP misestimated the period length for angle C, see Table 6.1 column 3. This leads to growing errors for larger time spans, cf. Figure 6.4 bottom right. The coupled and constrained GP can compensate for that by taking measurements from all directions into account. All in all, the constrained learning of GPs allows us to obtain reliable and realistic estimates for filtering and prediction of respiratory motion signals.

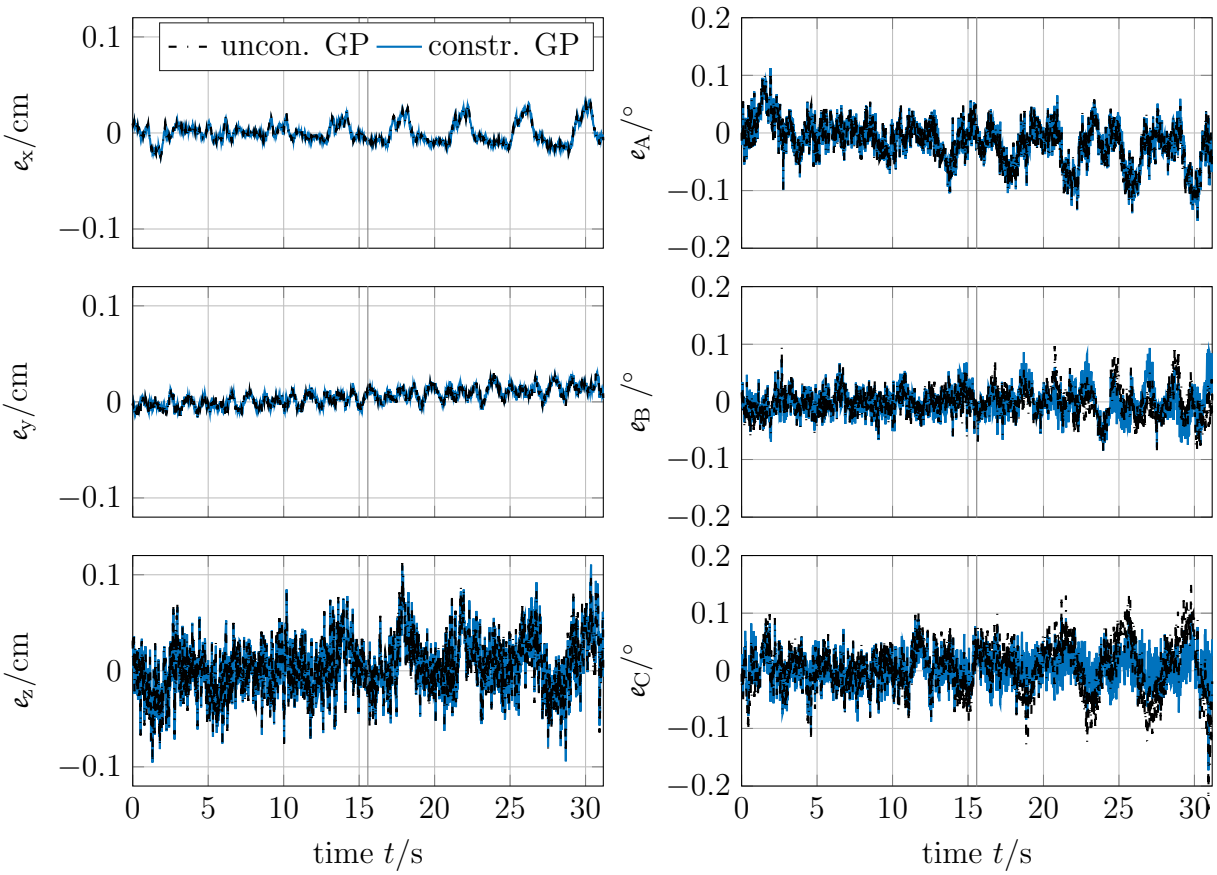


Figure 6.4: Prediction error for unconstrained and constrained GP learning.

Inverse Kinematics Algorithm

The inverse kinematics algorithm merges the predicted pose of the GP with the pre-operatively planned insertion path. Moreover, it maps the superimposed path to a desired joint angles motion of the robot. To do so, the inverse kinematic algorithm

$$\dot{q}_\mathrm{r} = J_\mathrm{A}^\dagger(q_\mathrm{r})\big(\dot{p}_\mathrm{GP} + K_\mathrm{P}(p_\mathrm{GP} - h_\mathrm{fk}(q_\mathrm{r}))\big) + \big(I - J_\mathrm{A}^\dagger(q_\mathrm{r})J_\mathrm{A}(q_\mathrm{r})\big)\,\dot{q}_\mathrm{path}$$

from [210] is used. Here, $J_A^\dagger(q_\mathrm{r}) \in \mathbb{R}^{n_\mathrm{q}\times 6}$ is the pseudo inverse of the analytic Jacobian of the manipulator and I is an identity matrix of size $n_\mathrm{q} \times n_\mathrm{q}$. The term $(p_\mathrm{GP} - h_\mathrm{fk}(q_\mathrm{r}))$ defines the error between the predicted Cartesian pose p_GP and the desired pose $h_\mathrm{fk}(q_\mathrm{r})$. So to say, it can be viewed as a control error. The matrix $K_\mathrm{P} \in \mathbb{R}^{n_\mathrm{q}\times 6}$ can be seen as a proportional controller matrix where q_r is the feedback. The controller matrix K_P should be chosen such that the inverse kinematic algorithm is asymptotically stable. Besides this feedback, two feed forward terms are included in the algorithm to track the change $\dot{p}_\mathrm{GP}$ and to assign a desired elbow motion via $\dot{q}_\mathrm{path}$. This algorithm uses the derivative of the patient motion $\dot{p}_\mathrm{GP}$. We explicitly exploit the fact that the GP allows us to predict the derivative of its output, as outlined in Section 4.6. Without a GP, the noisy data needs to be differentiated numerically to obtain the velocity information.

Table 6.1: Optimal hyperparameters for the constrained and unconstrained case.

	unconstrained GP training			constrained GP training		
	$\hat{l}_{\text{per}}$	$\hat{T}_{\text{per}}$	$\hat{\sigma}_{\text{per}}$	$\hat{l}_{\text{per}}$	$\hat{T}_{\text{per}}$	$\hat{\sigma}_{\text{per}}$
x	6.346	4.044 s	0.049	6.002	4.044 s	0.010
y	61.035	4.042 s	0.102	49.409	4.044 s	0.100
z	7.803	4.041 s	0.065	6.686	4.044 s	0.100
A	5.928	4.043 s	0.198	4.953	4.044 s	0.301
B	1.464	4.035 s	0.099	1.391	4.044 s	0.202
C	23.082	4.171 s	0.017	19.886	4.044 s	0.100

Figure 6.5 shows the result of a numerical differentiation via finite differences with and without a low pass filter. Moreover, it compares these velocities to the GP derivative prediction. Even though the low pass filter drastically reduces the noise, the GP allows for even smoother predictions. The GP outperforms standard filtering and the use of unfiltered data such that the inverse kinematics algorithm can work efficiently. Supplementary material on the performance of the inverse kinematics algorithm can be found in Appendix A.7. All in all, the proposed reference generator provides suitable references for the model predictive controller steering the robotic manipulator.

Model Predictive Control

In the simulations and experiments, we use the model predictive controller to manipulate a Franka Emika Panda robot, which holds the medical instrument, see Figure 6.6. The Franka Emika Panda is a seven degree of freedom lightweight robot [68]. It is designed for human-robot interactions and allows control on torque level, similar to the KUKA lightweight robot. In [73] the dynamic model (3.1) of the Franka Emika Panda has been parametrised. Based on this dynamical model, a feedback linearisation is performed [210]. The MPC including the feedback linearisation is illustrated in Figure 6.7. The state space formulation of the linearised model used in the MPC is given by

$$\dot{x} = \begin{pmatrix} \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} \end{pmatrix} x + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ I & \mathbf{0} \end{pmatrix} u \tag{6.1}$$

$$y = x, \tag{6.2}$$

where $\mathbf{0} \in \mathbb{R}^{n_\text{q} \times n_\text{q}}$ denotes a matrix of zeros, $I \in \mathbb{R}^{n_\text{q} \times n_\text{q}}$ is the identity matrix, and the states are $x = (q_1, q_2, \ldots, q_{n_\text{q}}, \dot{q}_1, \ldots, \dot{q}_{n_\text{q}})^\top$. The input $u \in \mathbb{R}^{n_\text{q}}$ is the input to the linearised system and refers to the acceleration of the joints. The surgical needle is

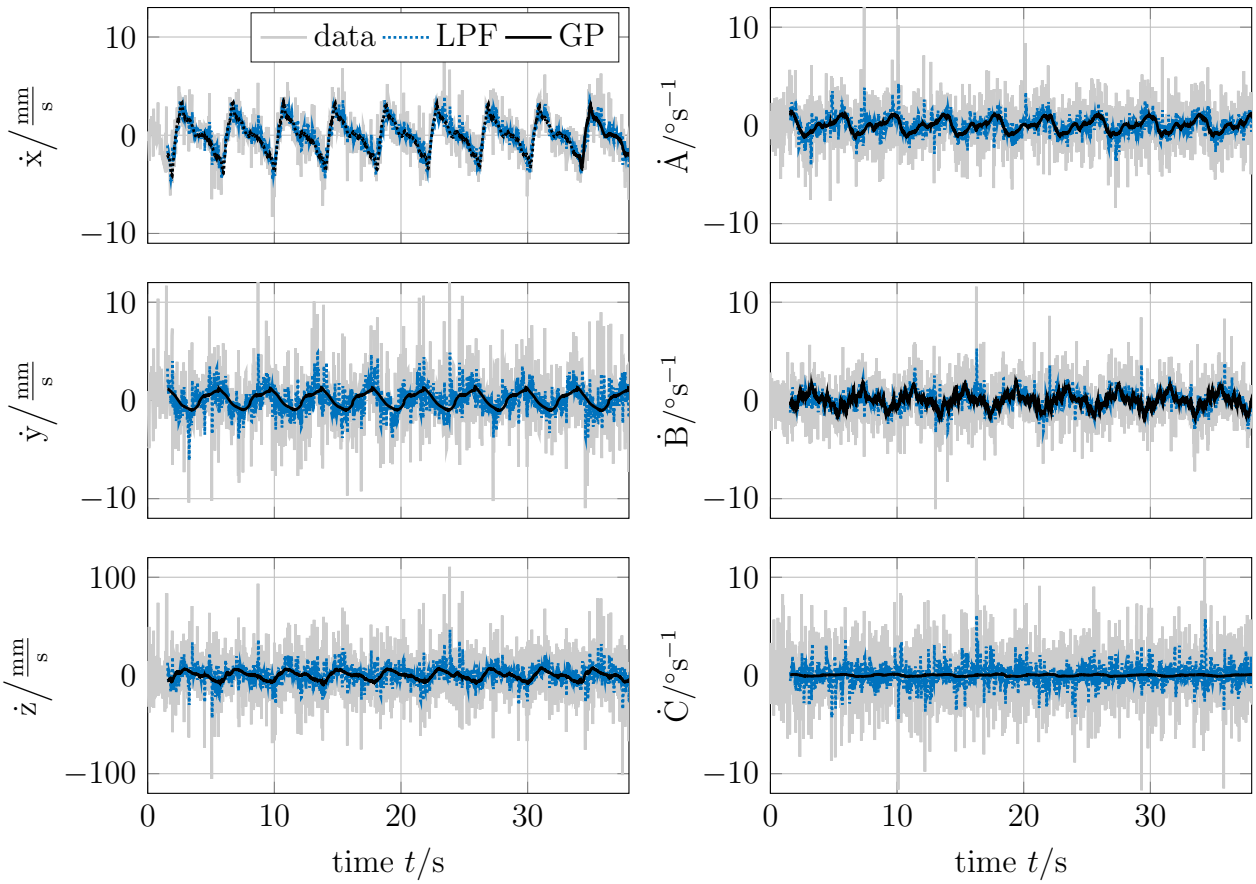


Figure 6.5: Comparison of velocity predictions based on finite difference, finite differences with low pass filter and GP.

aligned with the rotational axis of the sevens joint of the Panda. Since the needle is rotation symmetric, rotations around the last joint are redundant. Hence, we consider $n_q = 6$ in this example, while the last joint is kept constant. The MPC controller is designed according to Section 2.3 as a trajectory tracking controller using optimisation problem (2.11). In contrast to the robotic force control examples from Chapter 3 and Section 5.3.4, the reference speed cannot be used as an additional degree of freedom. This is because the motion of the robot must be synchronised to the patient respiratory motion to compensate it. Hence, this example is a typical trajectory tracking case, where we encounter time-dependent references.

The implementation of the overall control setup as shown in Figure 6.2 was done in MATLAB and acados [227]. Specifications on the objective function and the control parameters can be found in Appendix A.7. The optimal control problem was implemented in acados using a MATLAB interface. For the experiments, the communication between the controller and the robot was designed and setup in [96]. It uses an ethernet connection based on user datagram protocol (UDP). A C++ gateway function enables a real-time communication between MATLAB and the robot with user defined sampling times. In this work, 4 ms have been chosen as sampling time.

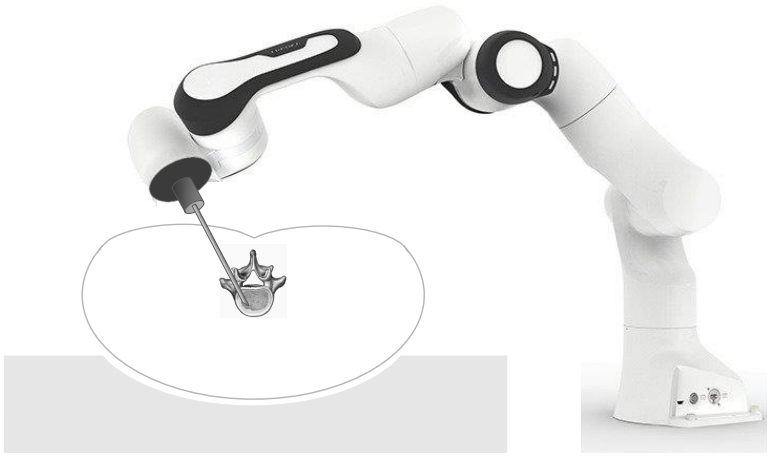


Figure 6.6: Illustration of Franka Emika Panda robot [69] in a minimally invasive treatment on the spine.

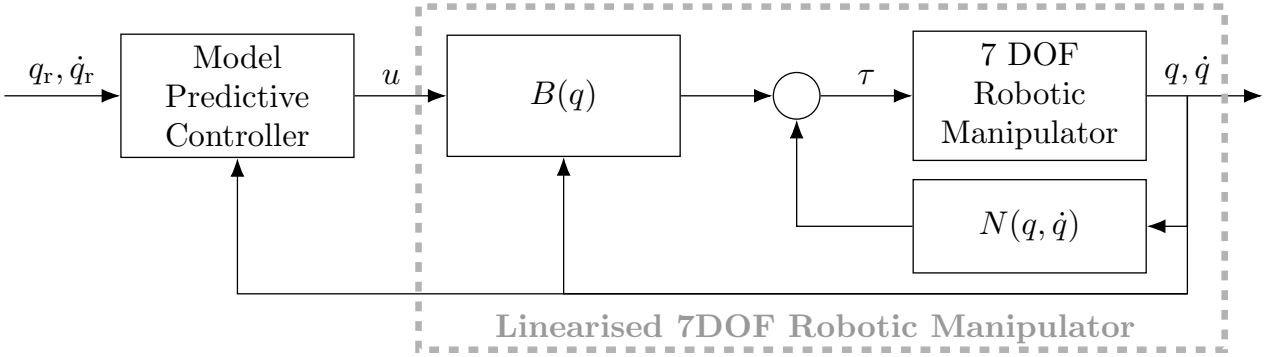


Figure 6.7: Model predictive controller with feedback linearisation.

6.4 Simulation Results

The simulation results for the model predictive controller for motion compensation are shown in Figure 6.8. The reference which is based on the Gaussian process and the inverse kinematics algorithm is depicted in dashed black. It represents a straight insertion path in Cartesian space along the needle with additional respiratory motion compensation. The insertion is finished after 14 s. Afterwards, the robot should holds its position while compensating respiratory motion of the patient. The controlled robot position is shown as a blue solid line in Figure 6.8. As can be seen, each joint angle follows its respective reference. The corresponding optimal inputs as well as the angular velocities and joint torques can be found in Appendix A.7. All control errors are below 0.06 °, see also Figure A.19 in Appendix A.7. Since the errors in joint space are often hard to interpret, we also discuss the control errors in the Cartesian space. Figure 6.9 shows the Cartesian pose errors in black. A comparison to the achievable performance of the same controller with different reference definitions is also shown in Figure 6.9. The solid light grey line describes the control error if the reference is using the data directly, and the dotted dark grey line uses low pass filtered data. In these

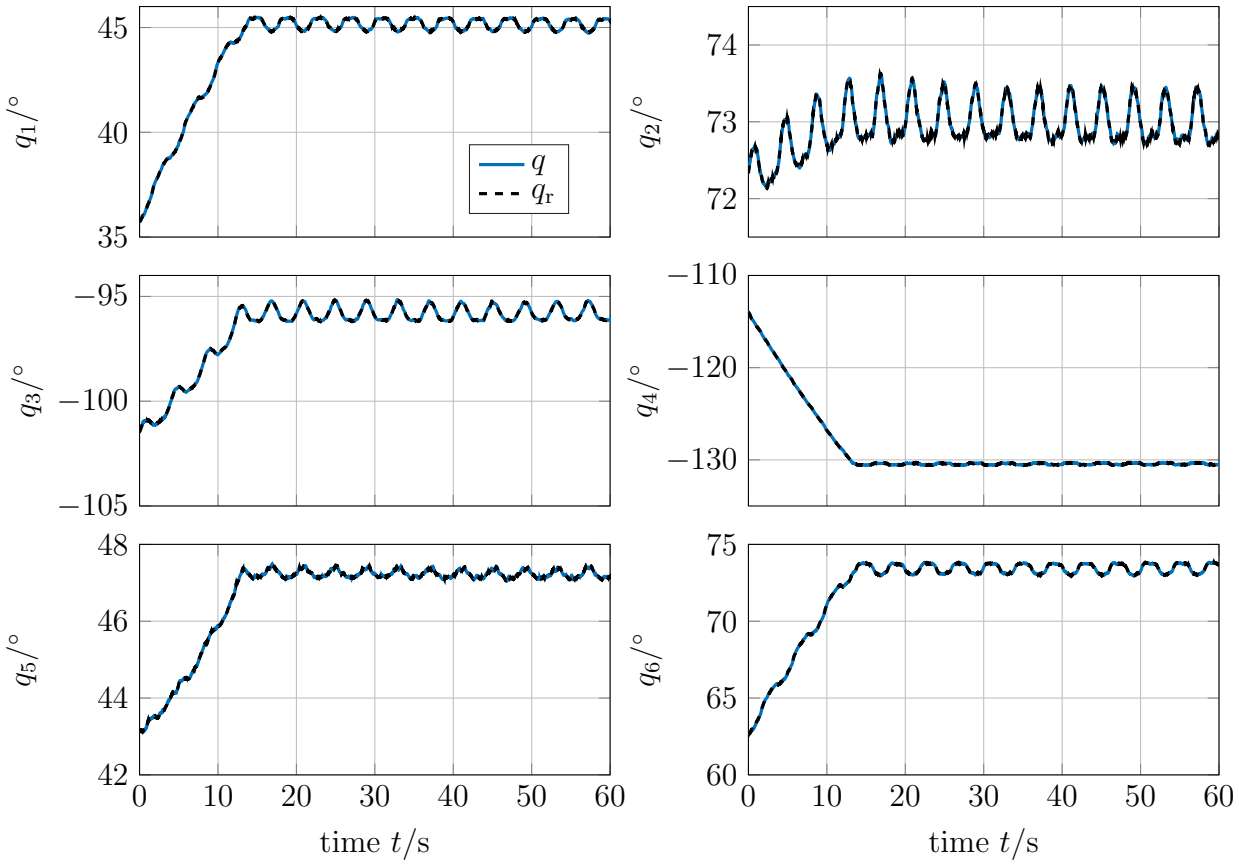


Figure 6.8: Angular positions for motion compensation in simulation.

two cases, no prediction of the upcoming references is performed. Hence, the current reference value is held constant over the prediction horizon. As we can see, the control error of the same controller (identical tuning, sampling and prediction horizon) varies significantly for different references. The noisy reference MPC produces the largest control errors of up to 1.1 mm and 0.15 °. The low pass filter exhibits smaller control errors up to 0.6 mm and 0.08 °. The best control performance is achieved by the GP-supported model predictive controller with maximum errors of 0.3 mm and 0.05 °. Overall, we were able to show that the GP based MPC outperforms non-learning based approaches. Besides the satisfactory performance, the proposed learning based MPC provides additional safety by incorporating constraints in the learning and control. Based on this successful evaluation in simulation of the Gaussian process based motion compensation via MPC, experiments on the real robot can be performed.

6.5 Experimental Validation

An experimental validation of the learning-supported motion compensation was performed. The same controller parameters as in the simulation are used, cf. Appendix A.7. The joint positions encountered in the experiments are shown in Figure 6.10 in blue. The robot follows its reference (black dashed) despite a model plant

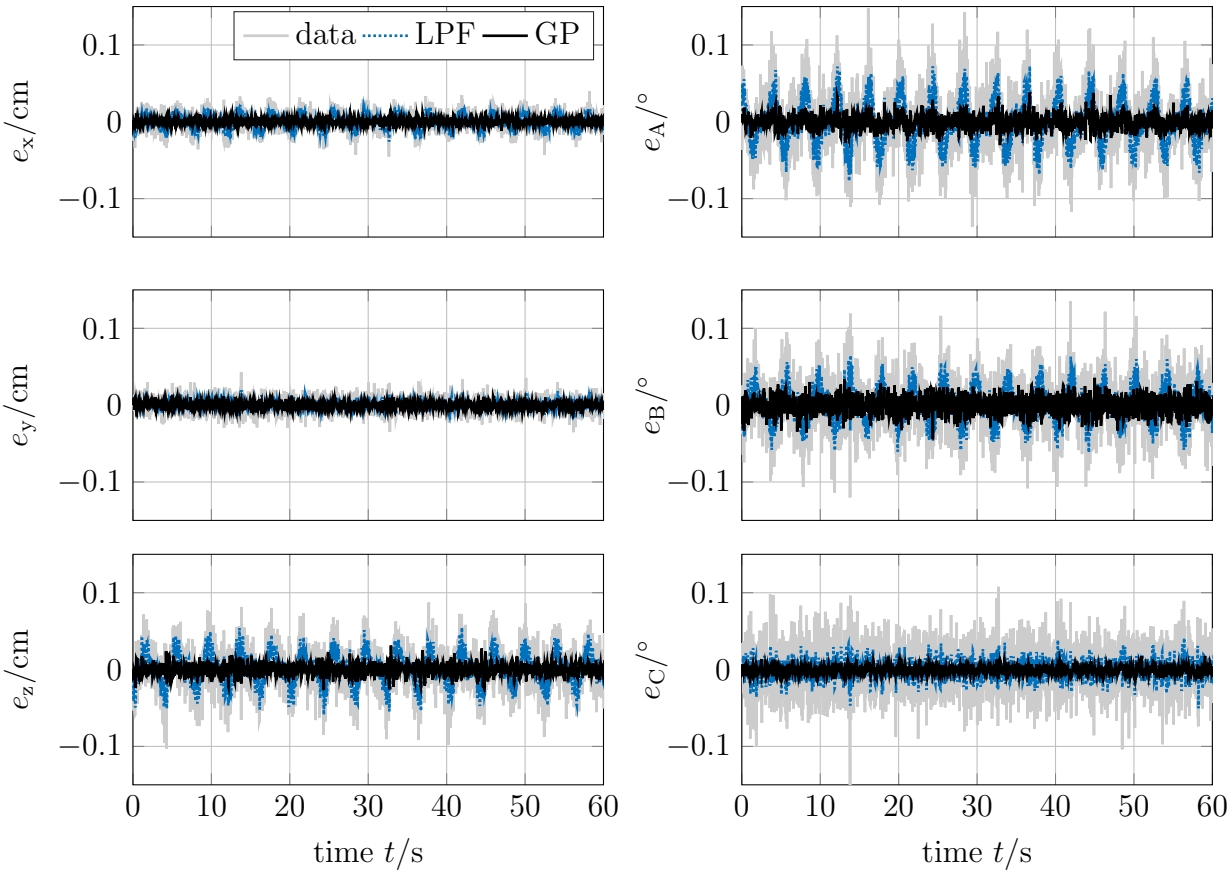


Figure 6.9: Cartesian errors for motion compensation in simulation.

mismatch. In Figure 6.11, a close-up of angle three is depicted to show the effect of the model-plant mismatch. The predictions of the optimal controller for every 50^{th} iteration are shown in gray. These predictions start at the current measurement and move towards the reference (black). However, the actual robot (blue) moves not exactly along the predictions resulting in a small offset from the reference. This model-plant mismatch leads to higher control errors compared to the simulations, which assume perfect cancellation of the non-linearities via the model-based feedback. The control errors of the joint angles are shown in Figure 6.12. Figure 6.12 shows that the control error is larger during the insertion phase than during pure motion compensation in most of the joints. This can originate from non-compensated nonlinearities which have a higher effect for larger or faster motions. During the compensation phase, an oscilation in the control error can be seen which synchronously moves with the desired reference. Since the motion repeatedly changes its direction for each of the joints, friction has a relatively high impact on the robot. Unfortunately, the friction model used for compensations from [73] does not represent all occurring effects in our robotic system. Especially joints four to six require much higher inputs than the model suggests to overcome the friction. The corresponding control inputs and joint torques are depicted in Appendix A.7. Nevertheless, all control errors are smaller than 0.75 °. The corresponding pose errors in Cartesian space are depicted in Figure 6.13. The absolute

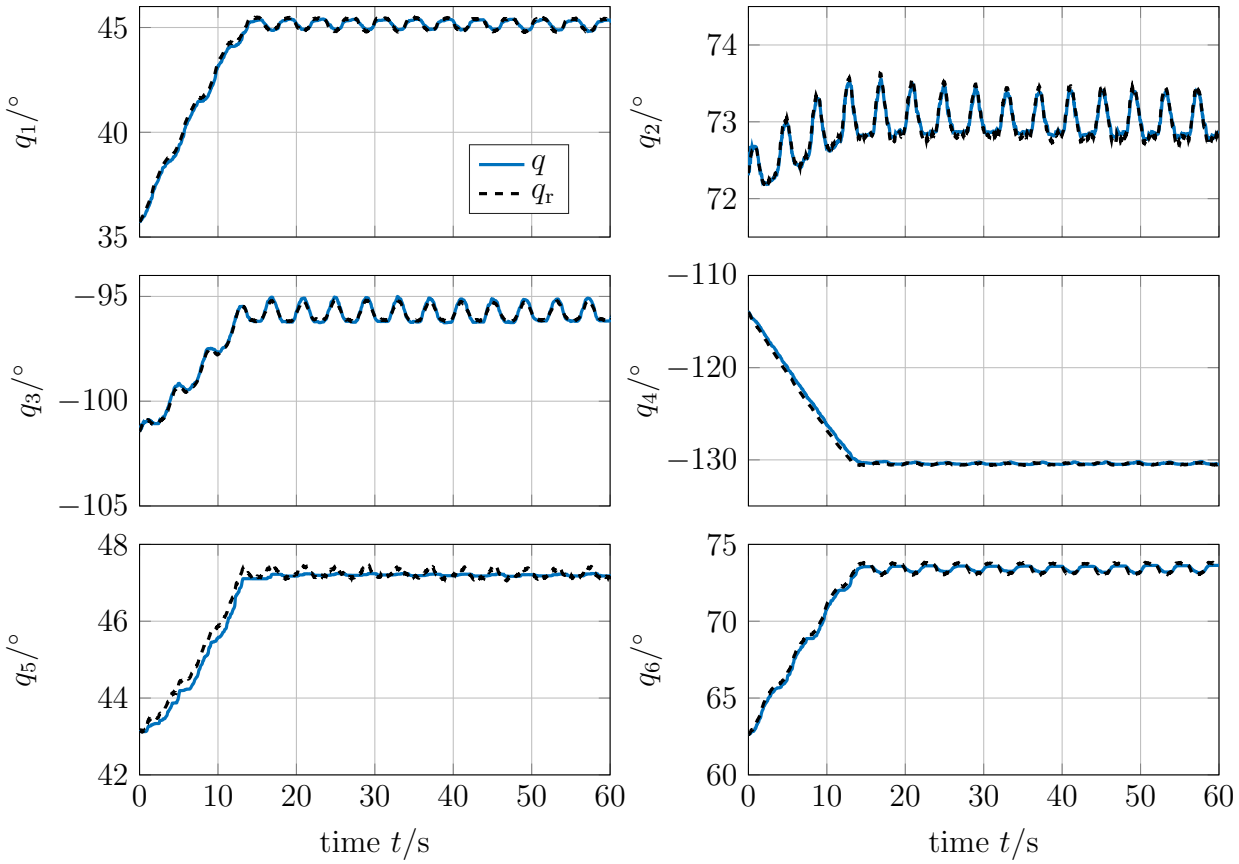


Figure 6.10: Angular positions for motion compensation in experiment.

Table 6.2: Cartesian pose errors for the experiments on learning-supported motion compensation.

	e_x	e_y	e_z	e_A	e_B	e_C
maximum error	2.76 mm	0.90 mm	2.10 mm	0.54 °	0.25 °	0.45 °
RMSE	0.97 mm	0.20 mm	0.78 mm	0.23 °	0.10 °	0.19 °

maximum errors of 2.76 mm and 0.54 ° occur in the Cartesian x-direction and by the rotation around the x-axis during the insertion, cf. Figure 6.13 and Table 6.2. The root mean square error in all directions lies below 1 mm and 0.3 °, respectively. Hence we obtained a submillimeter precision for the robot assisted needle placement. These high accuracies show the capabilities to improve treatments like the radio frequency ablation or pedicle screw placement in the spine by robotic assistance.

6.6 Summary

Experimental validation of the proposed learning-supported MPC for periodic references in robot-assisted surgery was conducted. Human respiratory motions were learned and compensated. Comparisons to standard filtering as well as non-predictive

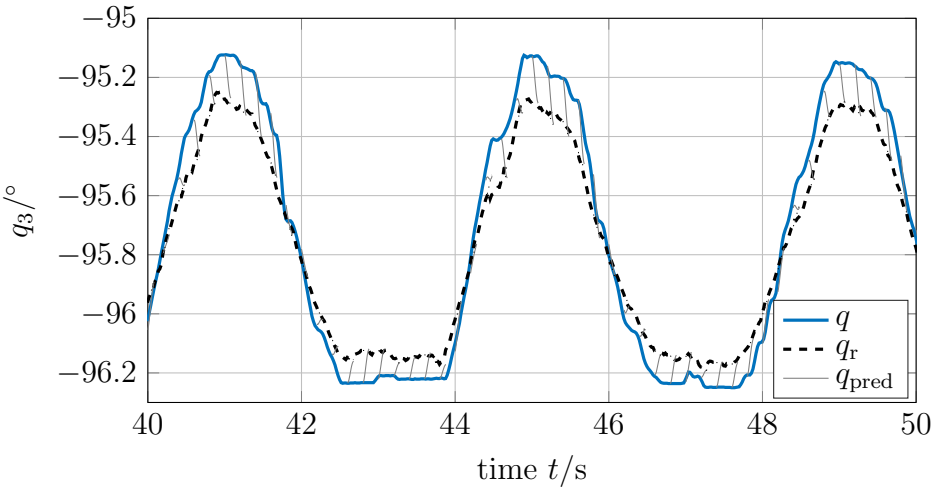


Figure 6.11: Close-up for angular position of joint three in experiment.

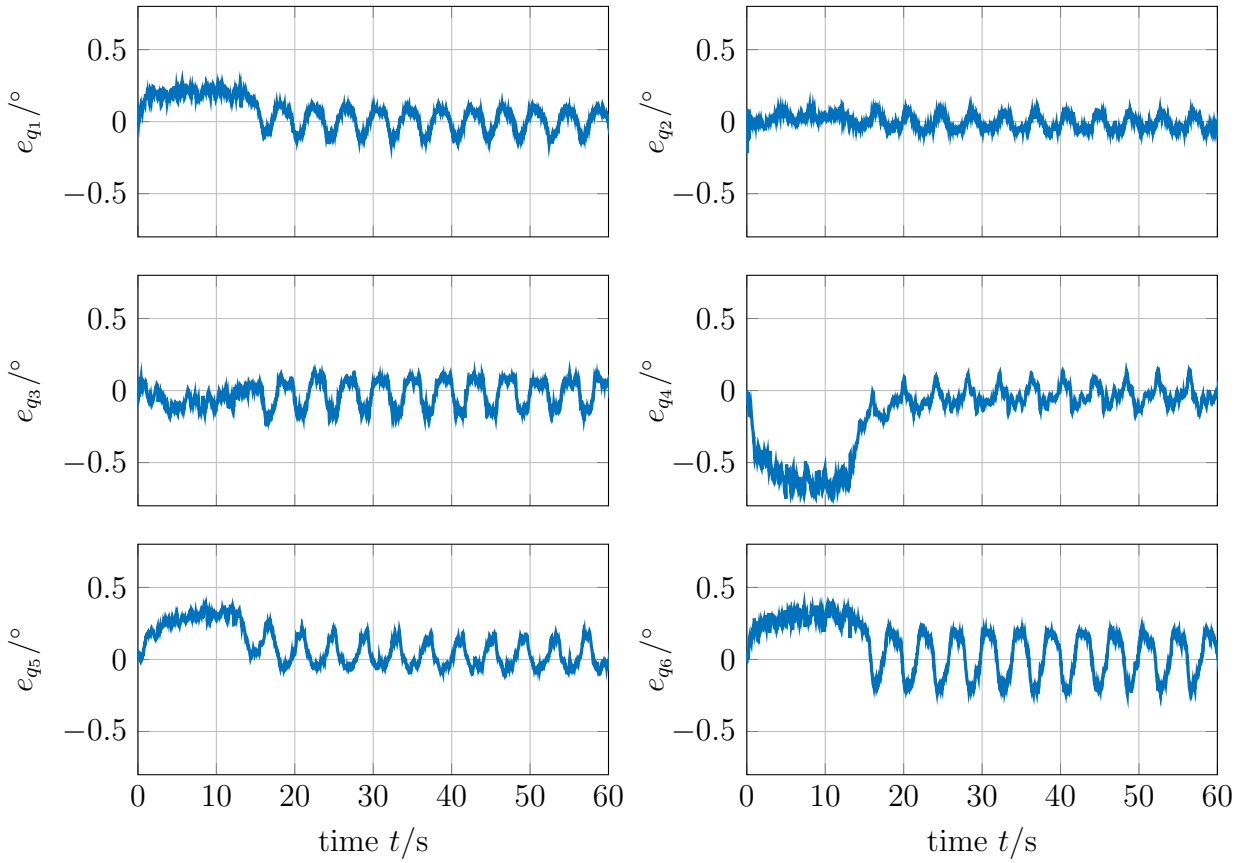


Figure 6.12: Angular position errors for motion compensation in experiment.

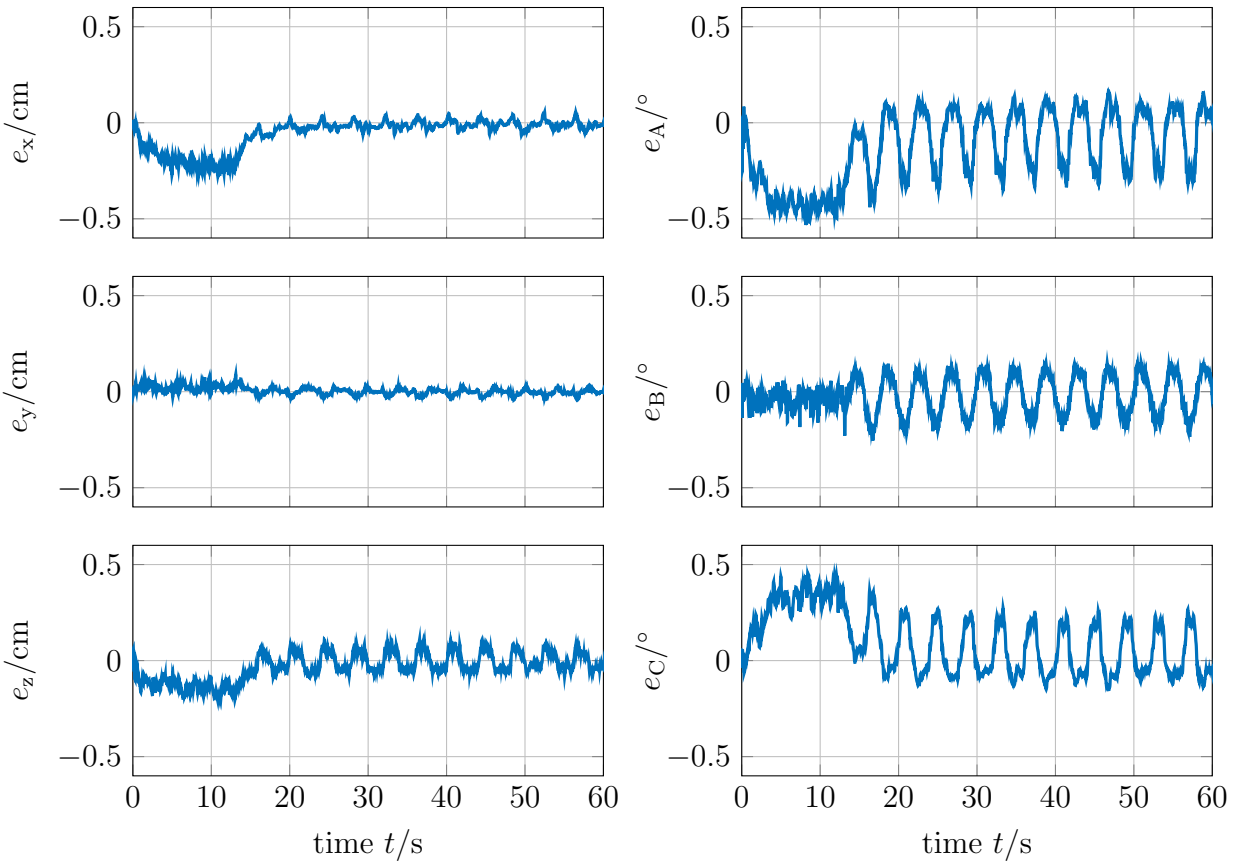


Figure 6.13: Cartesian position errors for motion compensation in experiment.

approaches for reference definitions showed the superior performance of the GP reference generator. The introduced constraints in the GP reference learning allowed for an implicit coupling of multiple GPs for each pose and orientational direction. We obtained improved prediction performance compared to unconstrained and fully decoupled GPs while also outperforming standard filtering techniques. Using the predicted respiratory motions in a model predictive controller enabled the precise positioning of surgical instruments relative to the anatomical goal structures while satisfying constraints on the actuators and the insertion path. We achieved a sub-millimetre precision of the needle tip in terms of the root mean square control error in Cartesian space. Hence, our concept allows precise positioning of medical equipment relatively to a moving goal position to support physicians in demanding medical treatments.

7 Conclusion and Perspective

This thesis aims at the development of new approaches to increase the autonomy of dynamical systems safely and reliably. To do so, we proposed model predictive control strategies for systems that interact with their environment and learn from this interaction. In particular, we considered robotic systems that are supposed to exert various forces on their surroundings or should synchronise their motions to them.

In the first part, we developed interaction force control schemes based on model predictive control. The proposed model predictive controller directly controls the interaction force. This allows the system to follow desired force profiles while ensuring safety by taking constraints on the contact forces into account. Loss of contact or damage by too high forces is prevented. In comparison to existing predictive force controllers, no artificial dynamics for the interaction need to be added. This lowers the computational complexity, while the controller tuning and prediction achieve the reduction of oscillations. To increase the performance even in the case of disturbances, path-following formulations were utilised. Consequently, the robot is behaving not only compliant in the force-controlled directions but also along the geometric reference path. For all approaches, we can establish stability. An experimental validation next to several simulation studies has underlined these advantages. It illustrates the increased autonomy for systems that perform motions with desired contact forces while the developed controller guarantees safe interaction. However, we have not leveraged all possible benefits from the proposed control setup. In the experiments, the surrounding materials have been homogeneous in space. Since model predictive control provides foresight, it would outperform non-predictive controllers all the more in environments with distributed material properties. Hence, an experimental validation with a more diverse environment should be considered in future work.

While the model-based force controller design allows for convergence, constraint satisfaction, and closed-loop stability, the performance and transferability of the controller can benefit from machine learning. Thus, we proposed Gaussian processes to learn static output maps from noisy data. With the devised setup, adaptation to varying environments is achieved without manual effort by experienced control engineers. Instead of including the machine learning scheme into the whole system model, we specialised the proposed approach on the learning of output maps. This perspective eliminated the necessity of uncertainty propagation through the dynamics, and hence, the approximation of non-Gaussian distributions. As shown, we can guarantee stability under reasonable conditions. The performance of the approach was compared to linear and nonlinear first-principle based controllers in simulations

and experiments. We showed that the learned contact force controller outperforms non-adaptive first-principle controllers by over 47 % in terms of the root mean square error for a given reference task. Moreover, the learning-supported controller showed superior performance over linear and adaptive first-principle models in both simulations and experiments. We validated the reliability of the learning-supported model predictive controller via an a posteriori constraint tightening based on the variance information of the Gaussian process. Overall, we accomplished an increased autonomy of systems that operate in varying environments, accompanied by additional safety features. However, online constraint tightening based on recently proposed approximation error bounds [122, 128] should be addressed in future research.

Besides the learning of outputs, we also addressed the task of a priori unknown references. In many corporative tasks, systems should synchronise their motions to other systems, humans, or follow a desired reference. In such cases, references are not known a priori but described via data. Future references might be unknown while past and current information can be corrupted with noise. We proposed the framework of Gaussian processes reference generators for model predictive control to tackle these issues. The Gaussian processes are trained on noisy and past data to predict references over the horizon of the model predictive controller. These references should be smooth and trackable, such that closed-loop stability of the control system under the learned references is maintained. To learn trackable references, we proposed a new constrained hyperparameter optimisation for the learning of Gaussian processes, which encodes the system properties. We outlined how this setup can be used for asymptotically constant and periodic references resulting in trackability and stability guarantees. Online data update schemes were proposed to include data on the fly while preventing trackability loss. The derived approach closely interconnects machine learning and control theory. Hence, we accomplished autonomous behaviour of dynamical systems that receive external signals as references while providing feasibility guarantees that enable reliable control. Besides several simulation examples, an experimental validation in robot-supported medical tasks was achieved. In robot-assisted surgery, precise placement of surgical instruments is needed, while the respiratory motions of patients must be compensated. The proposed learning-supported model predictive controller allows for such precise placement and compensation. It achieved a submillimeter precision in terms of the root mean square error in experiments.

While we tackled trajectory tracking formulations for the reference learning model predictive control setup, an extension to path following formulations is interesting as well. Future research could focus on the utilisation of the variance provided by the Gaussian process to extend a reference path to a corridor [56]. Moreover, a fusion of the proposed learning-supported force control with the reference generator would allow for more sensitive robot control. As Gaussian processes easily allow for sensor fusion, haptic and optical data could be merged in future work to equip robots with tact, sense and sensibility in delicate control tasks.

A Appendix

A.1 Parameters of Planar Robot Example

The inertia matrix B used in the examples of the Sections 2.2.4, 2.3.4, 2.4.4, and in Example 3, and 4 for the planar two degree of freedom robot is

$$B = \begin{pmatrix} b_5 \sin(q_2) + b_1 \cos(q_2) + b_2 & -b_6 \sin(q_2) + b_3 \cos(q_2) + b_4 \\ -b_6 \sin(q_2) + b_3 \cos(q_2) + b_4 & -b_4 \end{pmatrix}, \tag{A.1}$$

with the parameters

$$\begin{aligned} b_1 &= 1.0645822136, & b_4 &= -0.443227999912, \\ b_2 &= 1.747789839512, & b_5 &= 0.0137409152, \\ b_3 &= -0.5322911068, & b_6 &= 0.0068704576. \end{aligned}$$

These values were identified in [11]. The output $y = (y_1, y_2)^\top$ of the system is the Cartesian end-effector position, which can be obtained from the joint angles q_1 and q_2 via the forward kinematics

$$\begin{aligned} y_1 &= l_1 \sin(q_1) + l_2 \sin(q_1 - q_2) \\ y_2 &= l_1 \sin(q_1) + l_2 \cos(q_1 - q_2). \end{aligned}$$

Here, $l_1 = 0.4\,\mathrm{m}$ and $l_2 = 0.39\,\mathrm{m}$ are the link lengths, see also Figure 2.1b. The weighting matrices of the model predictive controller in Sections 2.2.4, 2.3.4, Example 3, and Example 4 are

$$\begin{aligned} Q_i &= \mathrm{diag}(1 \cdot 10^4, 1 \cdot 10^4) \\ R_i &= \mathrm{diag}(0.01, 0.01) \\ Q_{\mathrm{E},i} &= \mathrm{diag}(0.1, 0.1, 0.01, 0.01), \end{aligned}$$

with $i \in \{\mathrm{s,tt}\}$. For the path following case (Section 2.4.4), extended matrices are used to cope for the extended number of states, inputs, and outputs due to the virtual system. They are chosen to be

$$\begin{aligned} Q_{\mathrm{pf}} &= \mathrm{diag}(1 \cdot 10^4, 1 \cdot 10^4, 1000) \\ R_{\mathrm{pf}} &= \mathrm{diag}(0.01, 0.01, 0.01) \\ Q_{\mathrm{E,pf}} &= \mathrm{diag}(0.1, 0.1, 0.1, 0.1, 0.01, 0.01,). \end{aligned}$$

A.2 Direct Kinematics

The goal of direct kinematics is to express the end-effector coordinates in a base frame in dependence of the joint coordinates. A short overview about the involved transformations is given in the following based on [210]. Since the KUKA Lightweight robot and the Franka Emika Panda robot consist of seven revolute joints, the joint coordinates are the angular positions $q \in \mathbb{R}^7$. Following the Denavit-Hartenberg convention, a coordinate frame in each link of the manipulator is attached. The transformation from coordinate frame i to frame $i-1$ can be expressed by a homogeneous transformation matrix

$$T_i^{i-1}(q_i) = \begin{pmatrix} \cos(q_i) & -\sin(q_i)\cos(\alpha_i) & \sin(q_i)\sin(\alpha_i) & a_i\cos(q_i) \\ \sin(q_i) & \cos(q_i)\cos(\alpha_i) & -\cos(q_i)\sin(\alpha_i) & a_i\sin(q_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.2}$$

This transformation matrix contains the translation by d_i along the z-axis of Frame $i-1$, rotation by q_i around this axis, translation by a_i along the x-axis of the rotated frame, and rotation by α_i around this axis. The Denavit-Hartenberg parameter for the KUKA Lightweight robot with frames assigned as in Figure 3.1b are given in Table A.1. The Denavit-Hartenberg parameter for the Franka Emika Panda robot are listed in Table A.2. Based on this transformations, the overall mapping for the position and orientation of Frame 7 to Frame 0 is given by

$$T_7^0(q) = T_1^0(q_1) \cdot T_2^1(q_2) \cdot \ldots T_7^6(q_7). \tag{A.3}$$

Using this direct kinematics the transformation between the end-effector and a base frame can be obtained via

$$T_{\mathrm{e}}^{\mathrm{b}}(q) = T_0^{\mathrm{b}} \cdot T_7^0(q) \cdot T_{\mathrm{e}}^7, \tag{A.4}$$

where T_0^{b} and T_{e}^7 denote configuration independent transformations, which denote the position and orientation of frame 0 with respect to the base frame and the end-effector pose and orientation with respect to the seventh frame of the robot. For more details, see for instance [210].

In all examples in this work the transformation matrix T_0^{b} is the identity. The transformation from the last joint of the KUKA robot to the end-effector, i.e. the pen tip, is

$$T_{\mathrm{e}}^7 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.232\,\mathrm{m} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.5}$$

for the implementations in Chapter 3 while it is

$$T_{\mathrm{e}}^{7} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.183\,\mathrm{m} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.6}$$

in Section 5.3.4. This difference originates from the different mountings of the wrist sensor and the way of holding the pen. While in Chapter 3 the pen is held by the Barrett hand, in Section 5.3.4 the pen is placed in a small metal cylinder. The transformation from the last joint of the Panda robot to the end-effector, i.e. the needle-shaped electrode for the radio frequency ablation, is

$$T_{\mathrm{e}}^{7} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.1\,\mathrm{m} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.7}$$

Table A.1: Denavit-Hartenberg parameter for KUKA Lightweight robot for coordinate frames depicted in Figure 3.1b.

Joint	a_i [rad]	d_i[m]	α_i [rad]
1	0	0.31	$\pi/2$
2	0	0	-$\pi/2$
3	0	0.4	-$\pi/2$
4	0	0	$\pi/2$
5	0	0.39	$\pi/2$
6	0	0	-$\pi/2$
7	0	0.078	0

Table A.2: Denavit-Hartenberg parameter for Franka Emika Panda.

Joint	a_i [rad]	d_i[m]	α_i [rad]
1	0	0.33	-$\pi/2$
2	0	0	$\pi/2$
3	0.0825	0.316	$\pi/2$
4	-0.0825	0	-$\pi/2$
5	0	0.384	$\pi/2$
6	0.088	0	$\pi/2$
7	0	0.107	0

A.3 Minkowski Sum and Pontryagin Set Difference

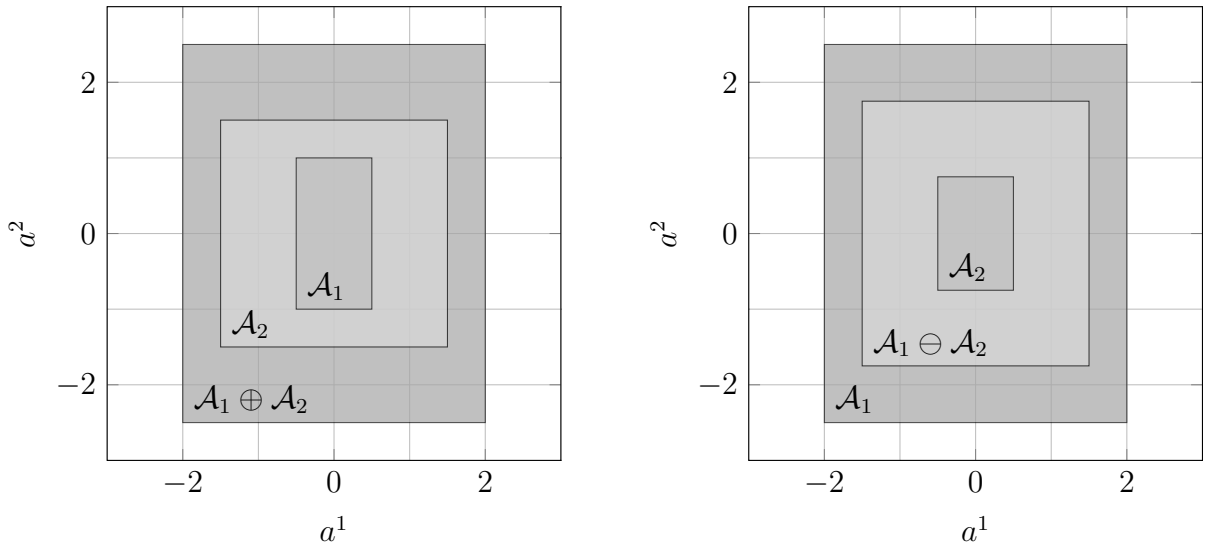
According to [20] the Minkowski sum of two polytopes $\mathcal{A}_1 \in \mathbb{R}^n, \mathcal{A}_2 \in \mathbb{R}^n$ is defined as

$$\mathcal{A}_1 \oplus \mathcal{A}_2 := \{a_1 + a_2 | a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2\}. \tag{A.8}$$

The Pontryagin set difference, which is denoted by $\ominus$, is given by

$$\mathcal{A}_1 \ominus \mathcal{A}_2 := \{a_1 \in \mathcal{A}_1 | a_1 + a_2 \in \mathcal{A}_1, \forall a_2 \in \mathcal{A}_2\}. \tag{A.9}$$

A visualization of the these operators is given in Figure A.1. Please note, that the Pontryagin difference is not the complement of the Minkowski sum, since $(\mathcal{A}_1 \ominus \mathcal{A}_2) \oplus \mathcal{A}_2 \subseteq \mathcal{A}_1$, see [20].



(a) Minkowski sum. **(b)** Pontryagin difference.

Figure A.1: Set addition and subtraction.

A.4 Supplementary Material for Direct Force Control

In the robotic writing example from Chapter 3, the reference for the path following case is chosen to be

$$r_{\mathrm{pf}}(\theta) = \begin{pmatrix} r_{\mathrm{pf,y}}(\theta) \\ r_{\mathrm{pf,z}}(\theta) \\ r_{\mathrm{pf},F}(\theta) \\ r_{\mathrm{pf},\theta}(\theta) \end{pmatrix} = \begin{pmatrix} a_{\mathrm{y},0} + a_{\mathrm{y},1}\theta \\ a_{\mathrm{z},0} + a_{\mathrm{z},1}\sin(2\pi\theta) \\ a_{F,0}\sum_{i=1}^{3} \dfrac{a_{F,1,i}}{\exp(a_{F,2}\theta + a_{F,3,i}) + 1} \\ \theta_{\mathrm{end}} \end{pmatrix} \tag{A.10}$$

with $a_{\mathrm{y},0} = -0.4761\,\mathrm{m}, a_{\mathrm{y},1} = 0.2\,\mathrm{m}, a_{\mathrm{z},0} = 0.6015\,\mathrm{m}, a_{\mathrm{z},1} = 0.04\,\mathrm{m}, a_{F,0} = 2\,\mathrm{N}, a_{F,1,1} = -1\,\mathrm{N}, a_{F,1,2} = -2\,\mathrm{N}, , a_{F,1,3} = 3\,\mathrm{N}, a_{F,2} = 120, a_{F,3,1} = 100, a_{F,3,2} = 62, a_{F,3,3} = 12$ and

$\theta_{\text{end}} = 0$. The weightings in the cost function for tracking and path following are

$$\begin{aligned} Q_{\text{pf}} &= \text{diag}(2 \cdot 10^5, 4 \cdot 10^5, 0.4, 40), \\ R_{\text{pf}} &= \text{diag}(0.4, 0.3, 0.3, 0.06), \\ Q_{\text{E,pf}} &= \text{diag}(0, 0, 0, 0, 0, 0, 40, 0), \\ Q_{\text{tt}} &= \text{diag}(2 \cdot 10^5, 4 \cdot 10^5, 0.4), \\ R_{\text{tt}} &= \text{diag}(0.4, 0.3, 0.3), \\ Q_{\text{E,tt}} &= \text{diag}(0, 0, 0, 0, 0, 0). \end{aligned}$$

In this example, a prediction horizon of $T = 150\,\text{ms}$ is used. The input is assumed to be piecewise constant between sampling times $T_\text{s} = 10\,\text{ms}$. The robot joint and velocity limits define $\mathcal{X} = [-170°, 170°] \times [-120°, 120°] \times [-120°, 120°] \times [-100°\text{s}^{-1}, 100°\text{s}^{-1}] \times [-110°\text{s}^{-1}, 110°\text{s}^{-1}] \times [-130°\text{s}^{-1}, 130°\text{s}^{-1}]$. The virtual states in path following are constrained by $\mathcal{Z} = [-1, 0] \times [0\,\text{s}^{-1}, 0.07\,\text{s}^{-1}]$. The following box constraints have been added for the inputs $\mathcal{U} = [-13\,\text{Nm}, 10\,\text{Nm}] \times [-5\,\text{Nm}, 5\,\text{Nm}] \times [-5\,\text{Nm}, 5\,\text{Nm}]$ and for the virtual input $\mathcal{V} = [-10\,\text{s}^{-2}, 0.1\,\text{s}^{-2}]$. Please note that these input torque constraints refer to the control of the robot where the gravity torques τ_g and the external torque τ_ext are compensated additionally. The former is compensated by inbuild controllers of the KUKA Lightweight robot. The latter is compensated via an additional feedback controller designed in [18] based on the six-dimensional measured forces and torques at the wrist. Hence, the input constraints do not reflect the actual physical limitations of the robot but a more narrow area of the desired operation which covers the required inputs for the considered motions.

A.5 Supplementary Material for Learning-supported Force Control

Additionally to the discussions provided in Section 5.3.4, this appendix gives details on the modelling errors and approximation qualities of the derived force models. Moreover, the used parameters for the controller design as well as complementary figures of the control signals are given.

Controller Parameters

For the controller simulation and experiments in Section 5.3.4 the same cost function and control parameters are used for all MPC independent of their respective force models. The quadratic cost functions of the involved predictive controllers are given by $L_\text{pf} = e_\text{pf}^\top Q_\text{pf} e_\text{pf} + u_\text{pf}^\top R_\text{pf} u_\text{pf}$, $E_\text{pf} = \varepsilon_\text{pf}^\top Q_\text{E,pf} \varepsilon_\text{pf}$ as in Chapter 3. The weightings are $Q_\text{pf} = \text{diag}(9 \cdot 10^6, 9 \cdot 10^6, 6, 4 \cdot 10^3)$, $R_\text{pf} = \text{diag}(6, 6, 6, 6)$, $Q_\text{E,pf} = \text{diag}(0, 0, 0, 0, 0, 0, 4 \cdot 10^3, 0)$. A prediction horizon of $T = 150\,\text{ms}$ and a sampling time of $T_\text{s} = 10\,\text{ms}$ are

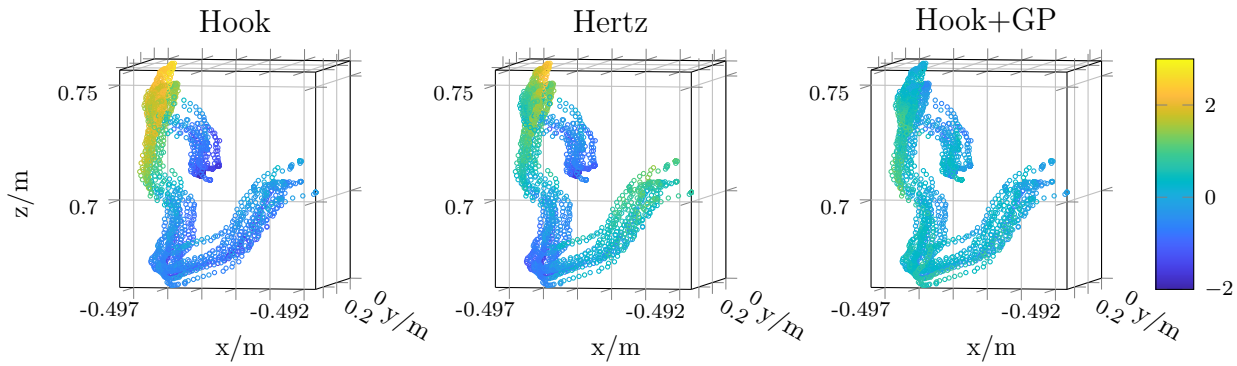


Figure A.2: Error between measured and modelled forces ranging from around −2 N to 3 N.

used. The state constraints are $\mathcal{X} = [-170°, 170°] \times [-120°, 120°] \times [-120°, 120°] \times [-100°\text{s}^{-1}, 100°\text{s}^{-1}] \times [-110°\text{s}^{-1}, 110°\text{s}^{-1}] \times [-130°\text{s}^{-1}, 130°\text{s}^{-1}]$. The virtual states in path following are constrained by $\mathcal{Z} = [-1, 0] \times [0\,\text{s}^{-1}, 0.07\,\text{s}^{-1}]$. The following box constraints have been added for the inputs $\mathcal{U} = [-13\,\text{Nm}, 10\,\text{Nm}] \times [-5\,\text{Nm}, 5\,\text{Nm}] \times [-5\,\text{Nm}, 5\,\text{Nm}]$ and for the virtual input $\mathcal{V} = [-10\,\text{s}^{-2}, 0.1\,\text{s}^{-2}]$. Furthermore, the contact force is constrained by $\text{proj}_3\mathcal{Y} = [0\,\text{N}, 7.5\,\text{N}]$ while these constraints could also build the basis for a constraint tightening.

Modelling

Figure A.2 shows the model errors, i.e. the deviation of the force model output from the measured values, for each of the considered models. As can be seen from the colour code, a significant reduction of the model error is achieved by the hybrid model (Figure A.2, right) compared to the first-principles models (Figure A.2, left and middle).

In Figure A.3 the training data set for the large GP used in the simulations outside of the optimal control problem is shown. Its approximation performance can be seen in Figures A.4-A.6. As a comparison the hybrid model with $n_{D_{\text{pred}}} = 60$ is shown in blue in Figures A.4-A.6. The root-mean-square error between the data and the large GP is 0.3148 N, while the maximum absolute error is 1.1968 N. It approximates the data slightly better than the hybrid model to the price of using a nearly seven times larger data set, cf. Table 5.1. Taking the computational burden into account which grows cubically with the number of data points for every iteration in the optimal control problem, the hybrid model finds the desired trade-off between model complexity and accuracy, while the large GP builds a more precise representation of the underlying true relation useful for simulations.

Simulation and Experimental Results

As supplementary material, the states and inputs of the controllers shown in Figure 5.8 are depicted in Figure A.7 for the robot and in Figure A.8 for the virtual system. Small differences between the state evolutions of the true system for each controller

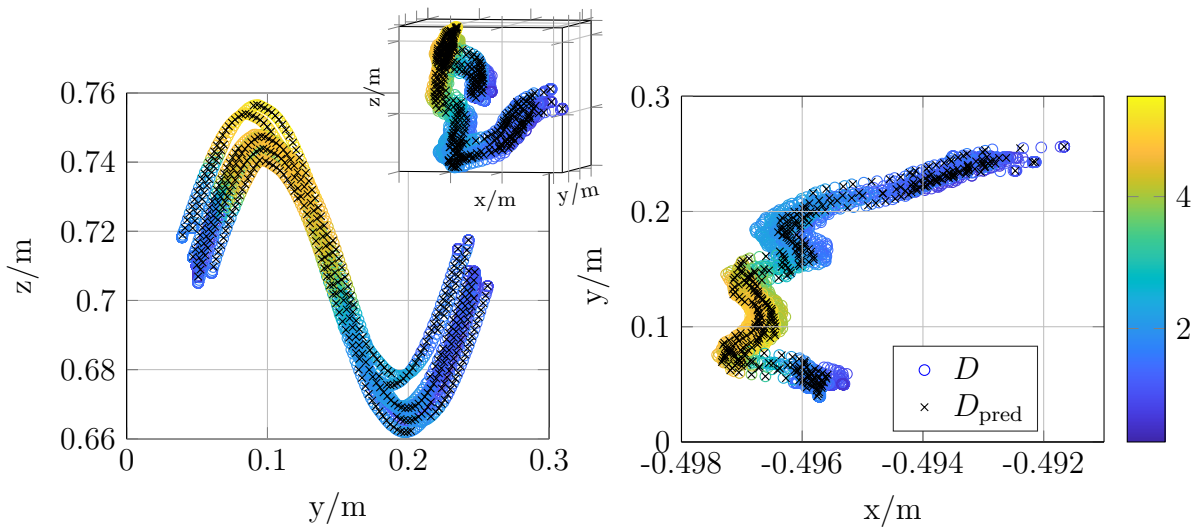


Figure A.3: Black crosses show the prediction data set ($n_{D_{\text{pred}}} = 419$) of the GP that is used in simulations outside of the optimal control problem to replace the real system.

can be seen leading to the different control performances discussed in Section 5.3.4, cf. Figure A.7. Nevertheless, constraints on the virtual system are satisfied for all controllers for all times, cf. Figure A.8 (left). Figure A.8 (right) shows the simulated closed-loop forces based on the large GP (blue) for the MPC with linear spring model (top), nonlinear spring (middle), and hybrid model (bottom). The predictions of the optimal control problems are shown as dashed black line. While the predictions of the controllers satisfy all constraints, the closed-loop system violates them in some cases. In the hybrid case, constraint tightening based on the posterior variances is shown in Figure A.8 (bottom right) for validation purposes. As can be seen, the OCP predictions (black, dashed) satisfy the shrunken constraints (red area), such that the closed loop response (blue) satisfies the original constraints ($F_{\text{n}} > 0$).

The joint angles and torques for the experiments from Section 5.3.4 are shown in Figure A.9. Clearly the noise in the sensor data including position, velocity, force, and torque measurements leads to larger variations in the control inputs as in the simulations. The corresponding virtual system evolution in the experiments is shown in Figure A.10, left. All input constraints, both for the torques as well as for the virtual input, are satisfied. In Figure A.10 right, the contact force with the hybrid model MPC is depicted. The top plot shows the full path with the simulation time from 0 s to 20 s. The middle plot shows a zoomed area in the first four seconds, while the lower plot shows the last six seconds. For validation purposes, the tightened constraints $\tilde{Y}$ are shown in red. As can be seen, these constraints are never active or violated.

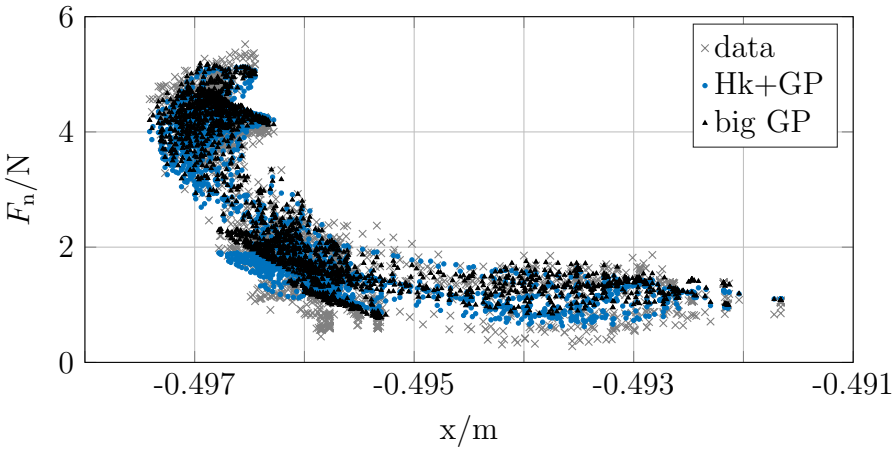


Figure A.4: Comparison of a large GP with $n_{D_{\text{pred}}} = 419$ (black triangles), the hybrid model used in the controller (blue dots) and the data (gray crosses) in x-direction.

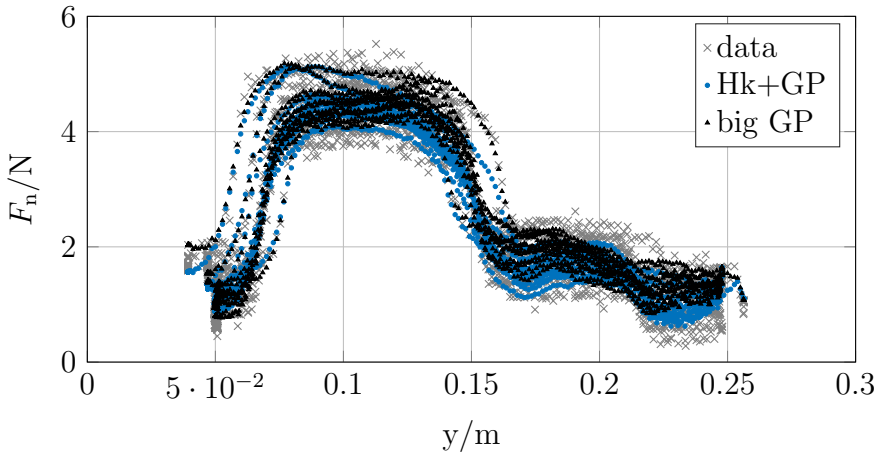


Figure A.5: Comparison of a large GP with $n_{D_{\text{pred}}} = 419$ (black triangles), the hybrid model used in the controller (blue dots) and the data (gray crosses) in y-direction.

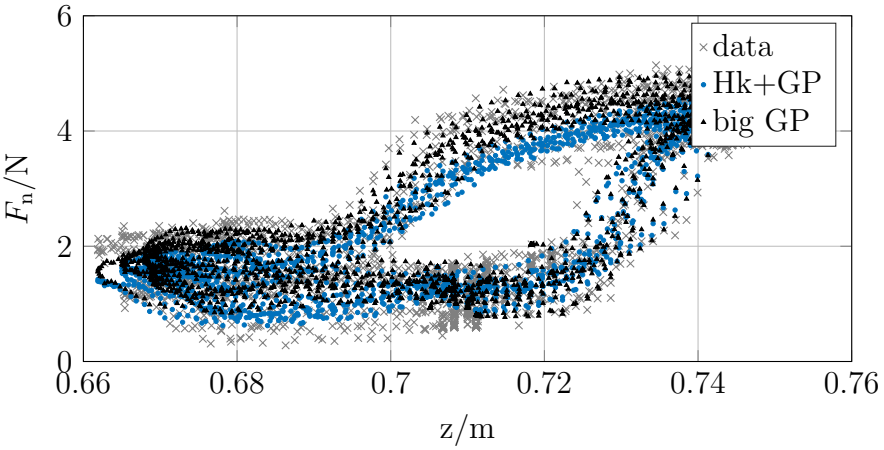


Figure A.6: Comparison of a large GP with $n_{D_{\text{pred}}} = 419$ (black triangles), the hybrid model used in the controller (blue dots) and the data (gray crosses) in z-direction.

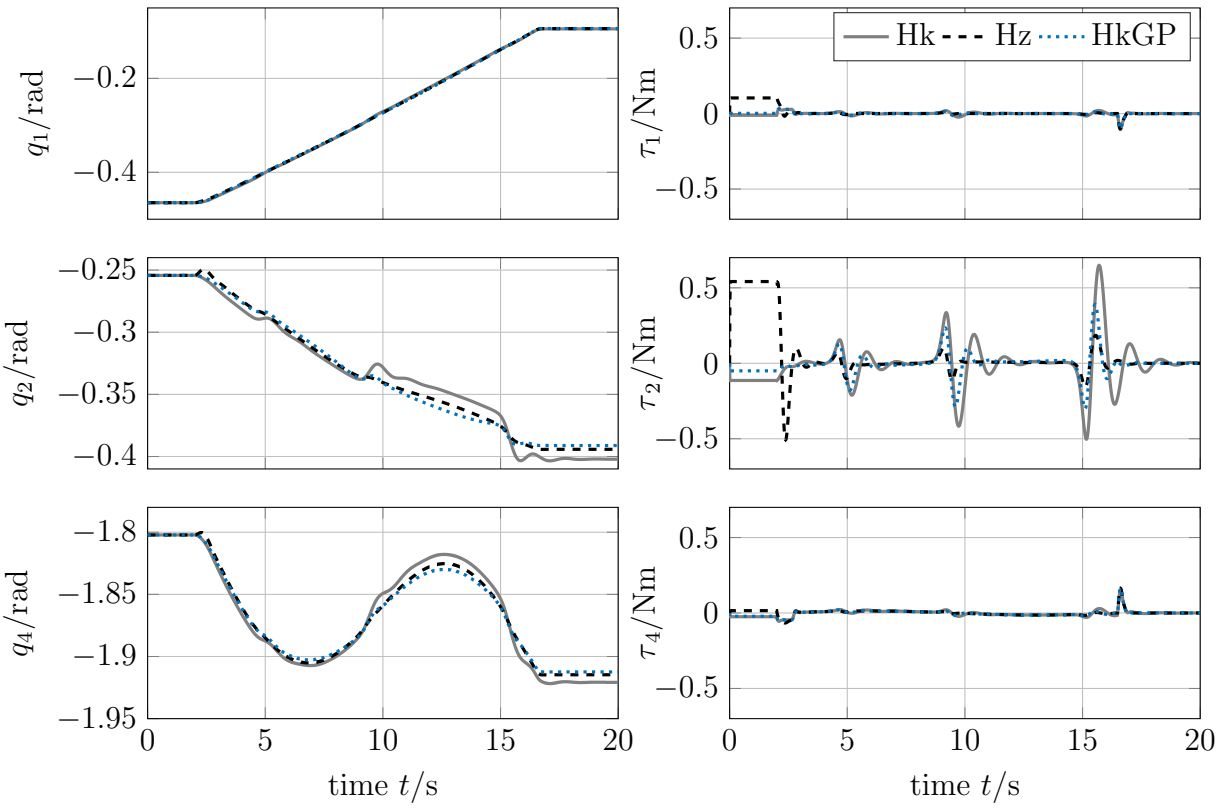


Figure A.7: States and inputs for the simulations shown in Figure 5.8.

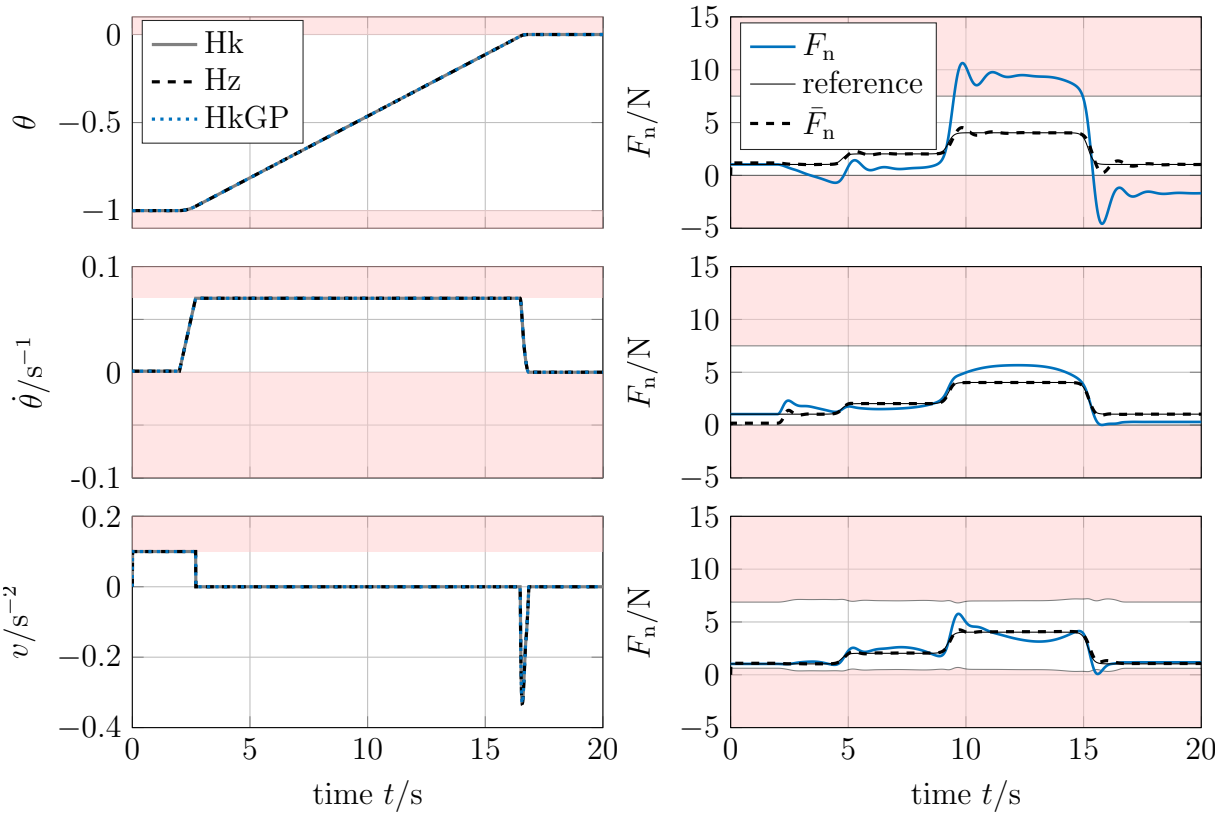


Figure A.8: Virtual system (left) and force prediction (right) corresponding to Figure 5.8.

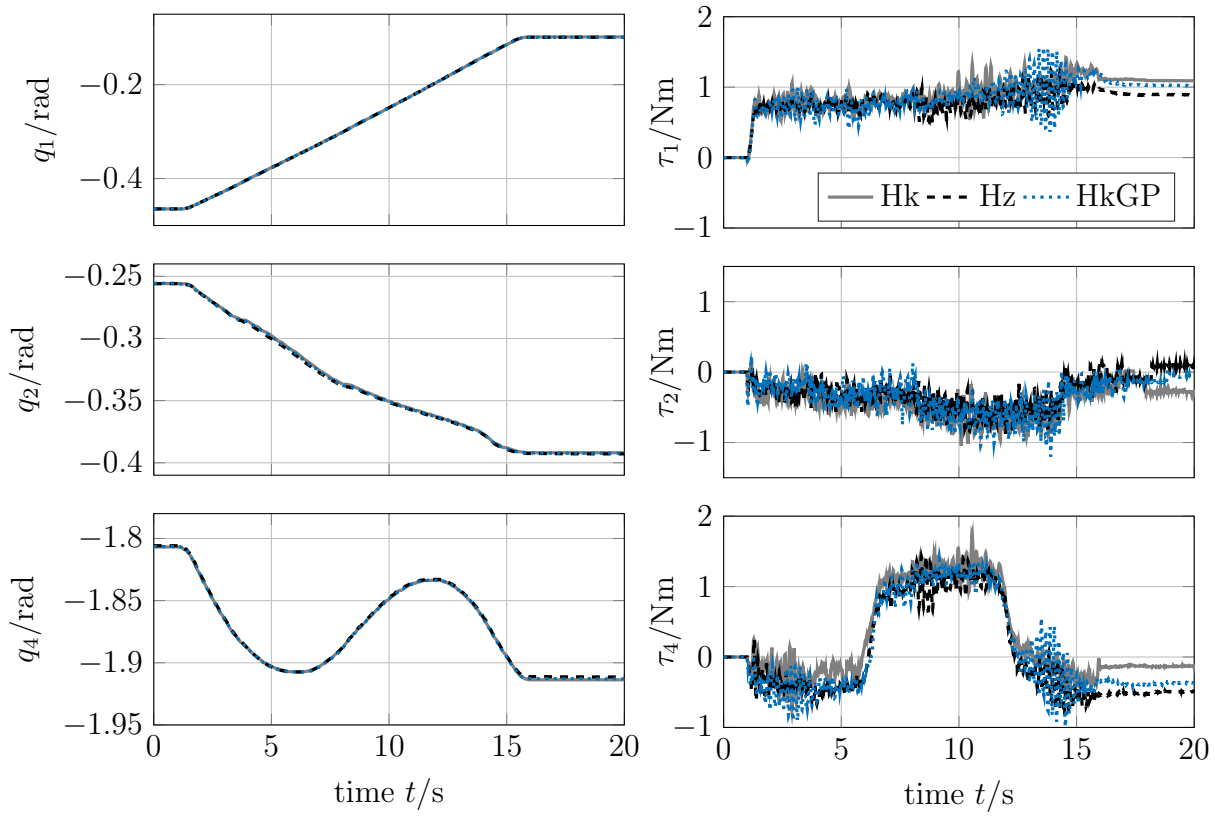


Figure A.9: States and inputs for the experiments shown in Figure 5.10.

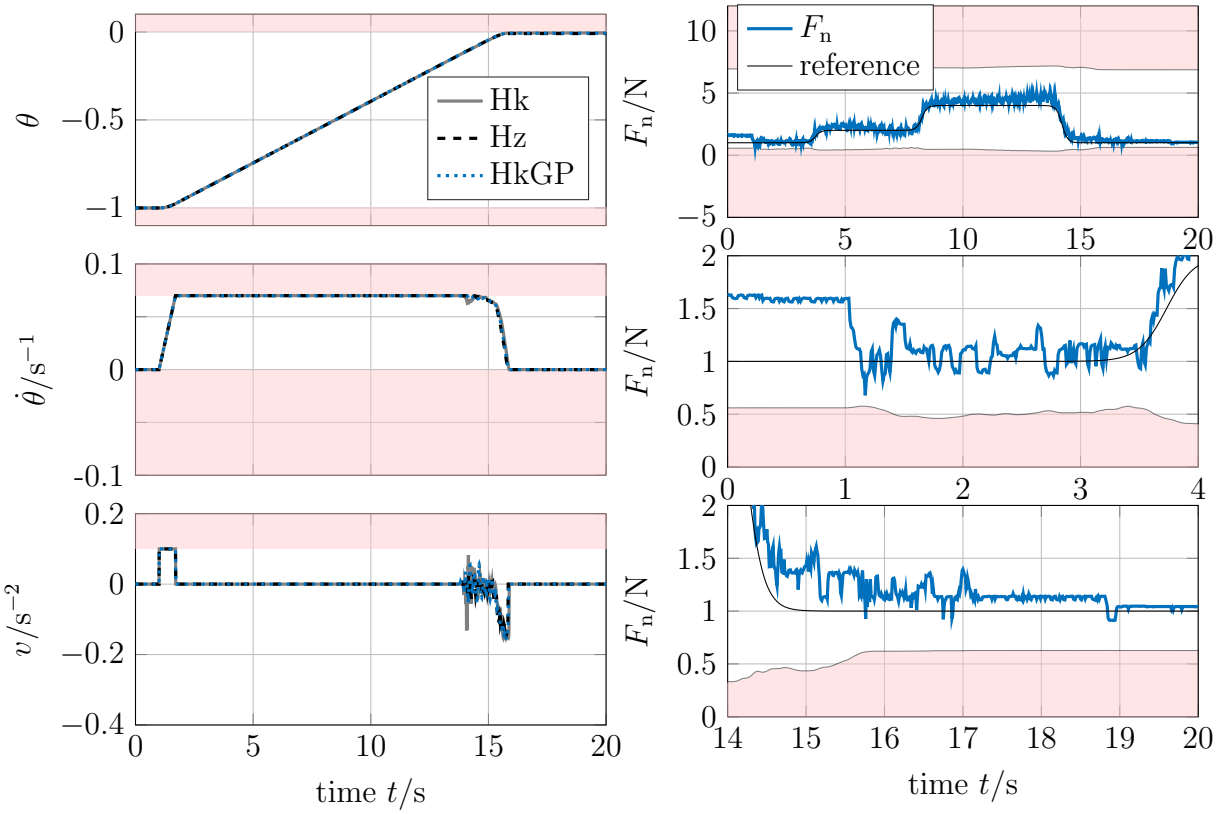


Figure A.10: Virtual system and contact force for the experiments shown in Figure 5.10.

A.6 Comparison of Adaptive Force Model and State Force Model

The model predictive controllers for the robotic manipulator in Chapter 3 and Chapter 5 rely on an output control error formulation. Hence, both the dynamic state and the output maps influence the controller performance. In these formulations the outputs are not nessesarily defined via the availablitiy of the measurements but are selected to represent the variables of interest we want to control. For example, the force model maps from joint angles to the contact force, which both can be measured. Hence, state feedback is possible without the nessecity of estimating the joint angles from the force measurements. Due to the relatively precise joint angle sensors compared to the force sensor, we use the joint angle sensor reeding directly. Therefore, in principle no online force measurments are nessesary from the theoretical perspective. However, the control performance will then entirely depend on the quality of the used contact force model. The resulting control performance for this setup is shown in Figure 5.8. Clearly, without force feedback, the influence of the model-plant mismatch is quite significant. In MPC, feedback is incorporated via the initial conditions of the states. If the contact force is considered as a dynamical state instead of an output of the system, force feedback can easily included via initial force values. To do so, the derivative of the output equation h or $\tilde{h}$ and the contact force F_{n} can be included in an extended system model. This was done for the linear and nonlinear spring model from Section 5.3 as well as for the hybrid model including the derivative of the posterior mean, cf. (4.14). This reformulation bears the drawback of including an additional differentiation in the design phase and consequently an additional integration in the numerical solution and optimisation. As an alternative, in Sections 3.3 and 5.3.4 an update of the output model is proposed which includes the same feedback information as an initial condition in a dynamic state equation. This update was implemented in the ACADO toolkit via the use of online data [7].

The control performance in Cartesian space of the derivative formulation is shown in Figure A.11. A comparison with the performance of the adaptive case shown in Figure 5.9 reveals no difference in the performances. The comparison of the contact force and the corresponding control errors for the derivative and the adaptive setup is provided in Figure A.12. As can be seen, both controller setups behave identical for the linear (Figure A.12, top), the nonlinear (Figure A.12, middle) and the hybrid force model (Figure A.12, bottom). However, the required computation times differ as depicted in Figure A.13. For the first-principles models no significant increase in the computation times for the extended dynamical system including the artificial force dynamics is visible, cf. Figure A.13 left and middle. In other word, the additional computational complexity of the added dynamical force equations is negligible for the first-principles models. However, the Gaussian-process-based hybrid model is considerably more complex. Hence it shows higher computational time in the dynamic force

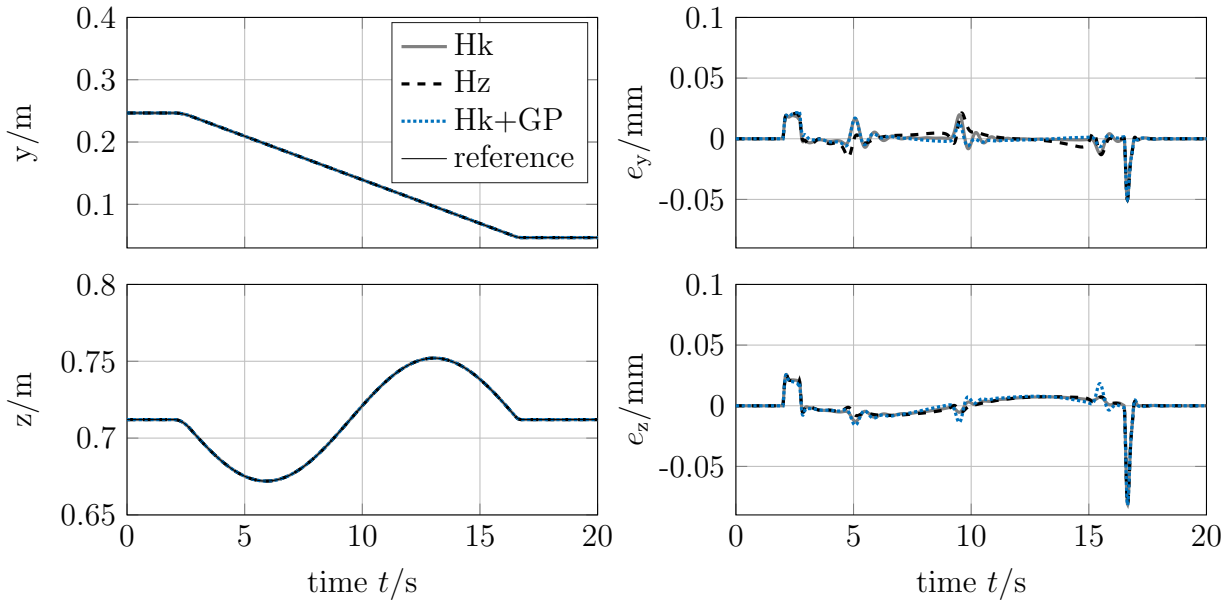


Figure A.11: Control performance for an extended dynamical system model including the derivative of the force model.

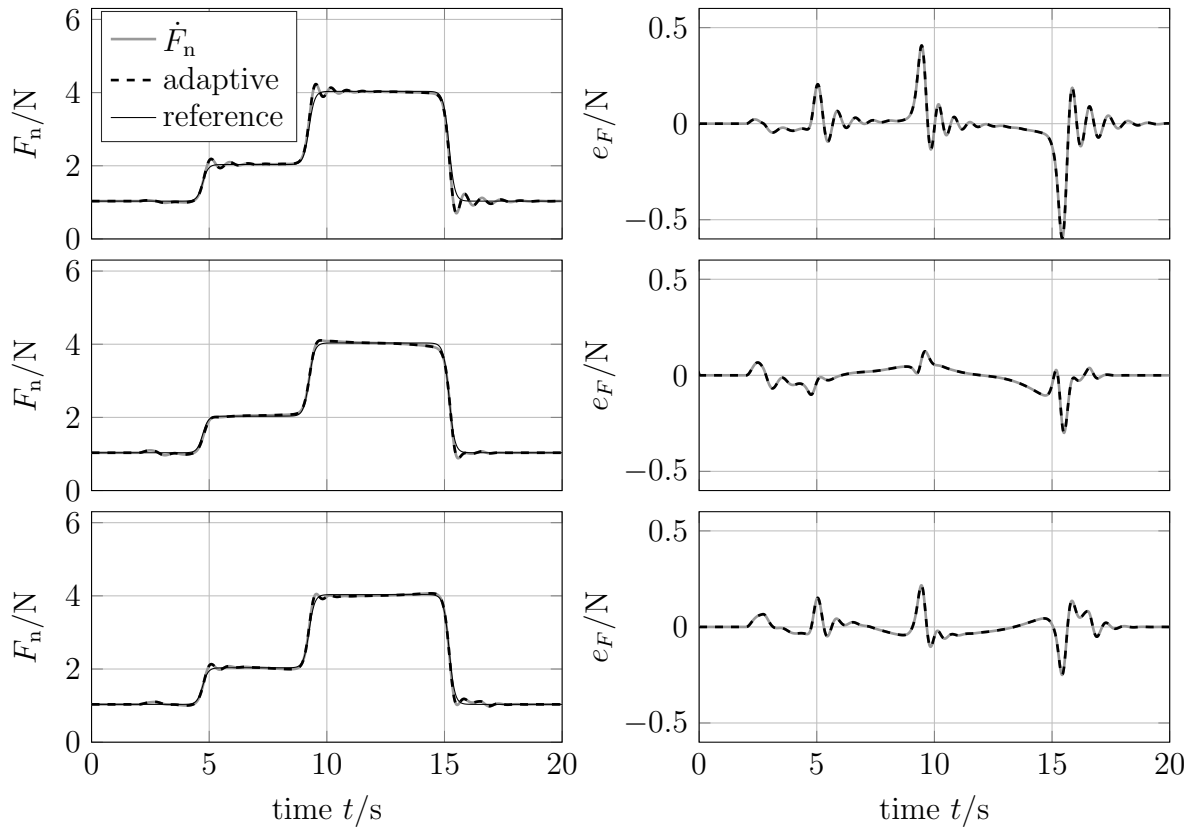


Figure A.12: Comparison between MPC with adaptive force model (black dashed) and dynamic force model (gray solid) for linear spring model (top), nonlinear spring model (middle), and hybrid GP model (bottom).

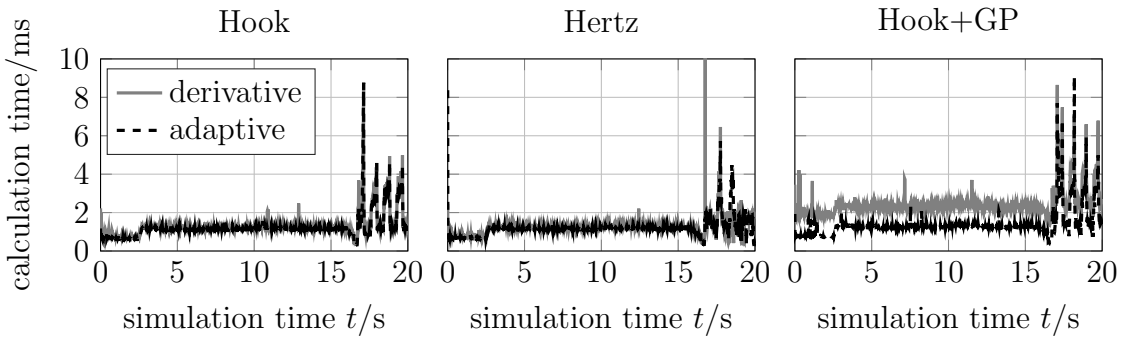


Figure A.13: Comparison of calculation times for the optimal control problem with adaptive force model (black dashed) and the derivative of the force model included in the dynamic system equations (gray solid).

case compared to the adaptive case, cf. Figure A.13 right. The average time for the differential formulation is 2.43 ms while the adaptive MPC only needs 1.37 ms in average to solve the optimal control problem. Consequently, the same control performance with lower computational time can be achieved via the proposed adaptation scheme. Moreover, the adaptive output model does not suffer from recursive evaluations of the GP at uncertain inputs as in a dynamical system. Hence, variance information does not need to be approximated but can be determined via the posterior covariance function as usual.

A.7 Supplementary Material for the Motion Compensation Example

The inverse kinematics algorithm merges the motions captured in Cartesian space and the planned insertion path in joint space. It involves the gain K_P which was chosen to be $K_\mathrm{P} = \mathrm{diag}(160, 160, 160, 40, 40, 40)$. To evaluate the inverse kinematics algorithm, three different inputs to the algorithm are tested. In the first scenario, the inverse kinematics algorithm uses the captured motion data directly. A low pass filter is used in a second scenario. In the third scenario, the inverse kinematics algorithm uses the proposed Gaussian process for prediction and filtering. A comparison of the resulting joint space references is shown in Figures A.14 and A.15. The grey line denotes the solution of the inverse kinematic algorithm based on the unfiltered data. As can be seen, the noise from the Cartesian space is propagated through the inverse kinematics mapping. Especially the joint velocity references are heavily affected by the noise in the data. The low pass filter is lowering this effect, cf. Figures A.14 and A.15 blue dashed lines. Even more smooth references are obtained when the inverse kinematics use the GP filtered predictions, cf. Figures A.14 and A.15 black lines. Moreover, the low pass filter introduces a noticeable delay, cf. Figure A.14 (left middle). The basic idea of the GP prediction is to eliminate the issues via a smooth prediction of the

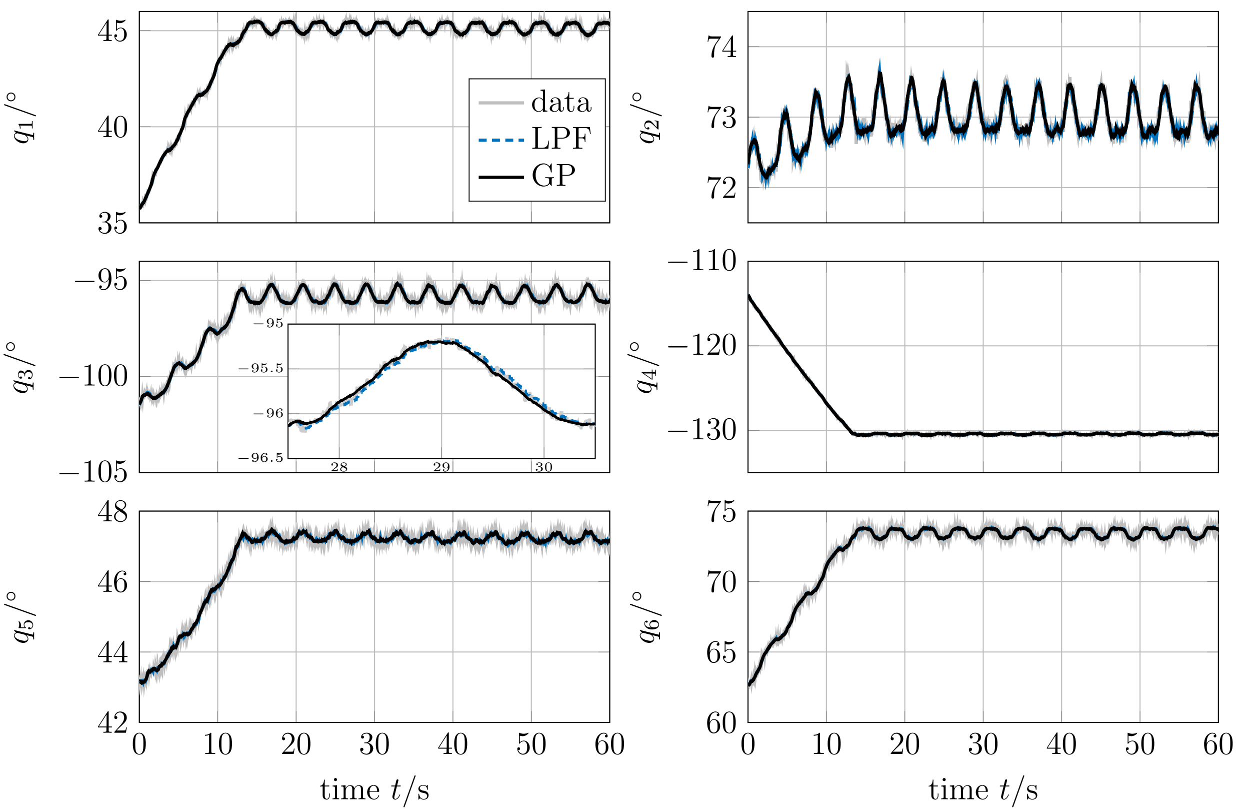


Figure A.14: Angular position references from inverse kinematics.

motions. This goal has been accomplished.

For the simulation and experiments in Chapter 6 the cost functions $L_{\mathrm{tt}} = e_{\mathrm{tt}}^{\top} Q_{\mathrm{tt}} e_{\mathrm{tt}} + u_{\mathrm{tt}}^{\top} R_{\mathrm{tt}} u_{\mathrm{tt}}$, and $E_{\mathrm{tt}} = \varepsilon_{\mathrm{tt}}^{\top} Q_{\mathrm{E,tt}} \varepsilon_{\mathrm{tt}}$ are used. The involved weighting matrices used for simulations and experiments are

$$\begin{aligned} Q_{\mathrm{tt}} &= \mathrm{diag}(10^6, 10^6, 10^6, 1.2 \cdot 10^6, 1.6 \cdot 10^6, 1.4 \cdot 10^6, 100, 100, 100, 1, 0.1, 0.1) \\ R_{\mathrm{tt}} &= \mathrm{diag}(1, 0.1, 1, 0.5, 0.001, 0.01) \\ Q_{\mathrm{E,tt}} &= 10^7 \cdot I, \end{aligned}$$

with $I \in \mathbb{R}^{2n_{\mathrm{q}} \times 2n_{\mathrm{q}}}$.

The state constraints according to the Panda data sheet [68] including a reduction of the allowed maximum velocities are $\mathcal{X} = [-166°, 166°] \times [-101°, 101°] \times [-166°, 166°] \times [-176°, -4°] \times [-166°, 166°] \times [-1°, 215°] \times [-115° \mathrm{s}^{-1}, 115° \mathrm{s}^{-1}] \times [-115° \mathrm{s}^{-1}, 115° \mathrm{s}^{-1}] \times [-115° \mathrm{s}^{-1}, 115° \mathrm{s}^{-1}] \times [-115° \mathrm{s}^{-1}, 115° \mathrm{s}^{-1}] \times [-143° \mathrm{s}^{-1}, 143° \mathrm{s}^{-1}] \times [-143° \mathrm{s}^{-1}, 143° \mathrm{s}^{-1}]$.

Constraints on the optimal inputs are $\mathcal{U} = [-8.5 \cdot 10^3 \,° \mathrm{s}^{-2}, -8.5 \cdot 10^3 \,° \mathrm{s}^{-2}] \times [-8.5 \cdot 10^3 \,° \mathrm{s}^{-2}, -8.5 \cdot 10^3 \,° \mathrm{s}^{-2}] \times [-8.5 \cdot 10^3 \,° \mathrm{s}^{-2}, -8.5 \cdot 10^3 \,° \mathrm{s}^{-2}] \times [-8.5 \cdot 10^3 \,° \mathrm{s}^{-2}, -8.5 \cdot 10^3 \,° \mathrm{s}^{-2}] \times [-8.5 \cdot 10^3 \,° \mathrm{s}^{-2}, -8.5 \cdot 10^3 \,° \mathrm{s}^{-2}] \times [-8.5 \cdot 10^3 \,° \mathrm{s}^{-2}, -8.5 \cdot 10^3 \,° \mathrm{s}^{-2}]$. A prediction horizon of $T = 80\,\mathrm{ms}$ and a sampling time of $T_{\mathrm{s}} = 4\,\mathrm{ms}$ are used.

Supplementary information for the simulation results are given in Figures A.16-A.20. Figure A.16 shows the optimal inputs, while Figure A.17 displays the resulting

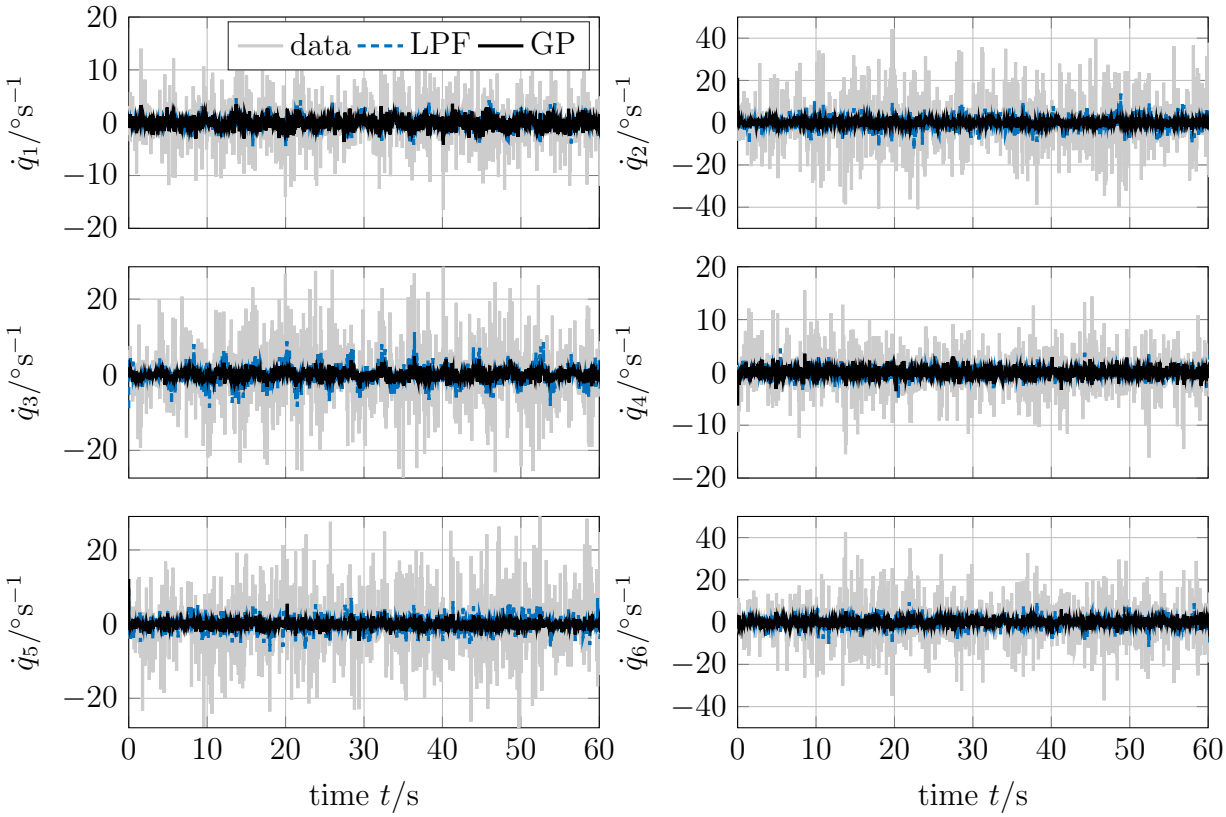


Figure A.15: Angular velocity references from inverse kinematics.

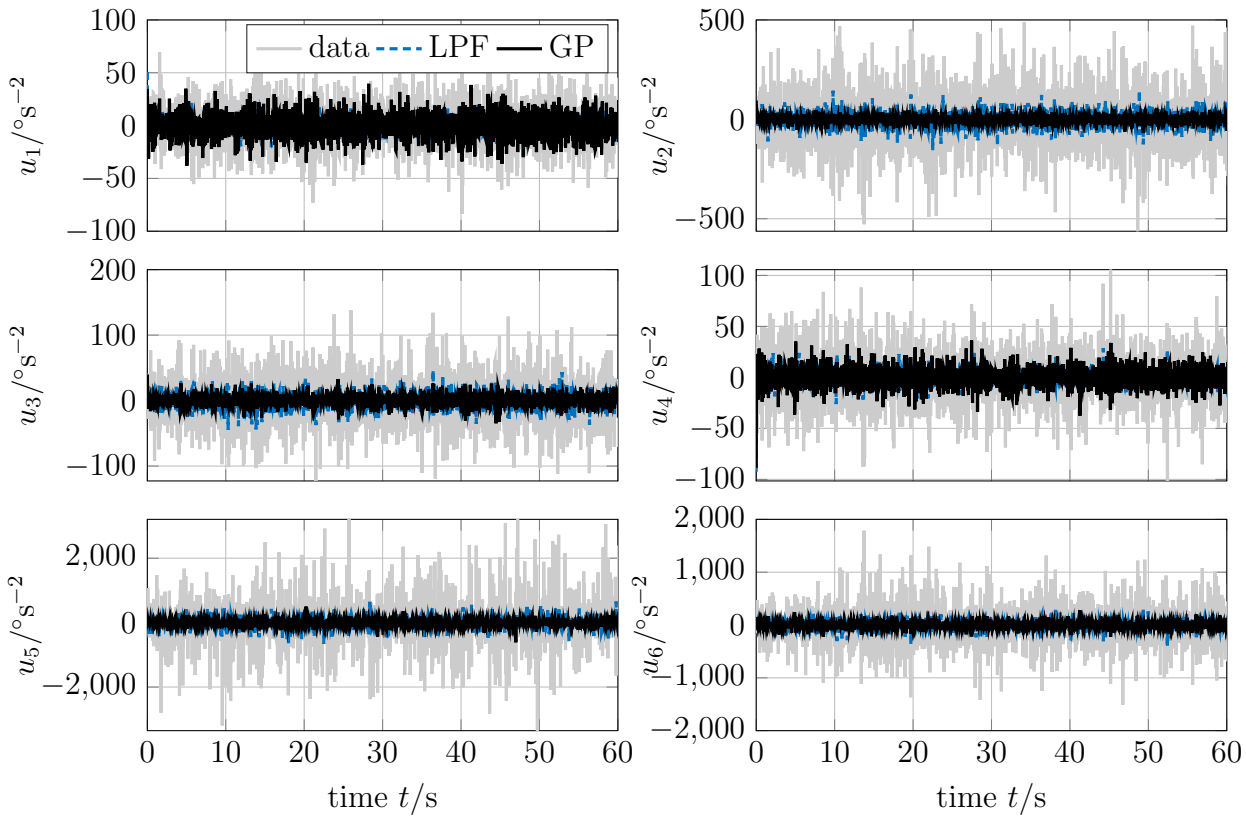


Figure A.16: Angular accelerations as optimal inputs for linearised system in simulation.

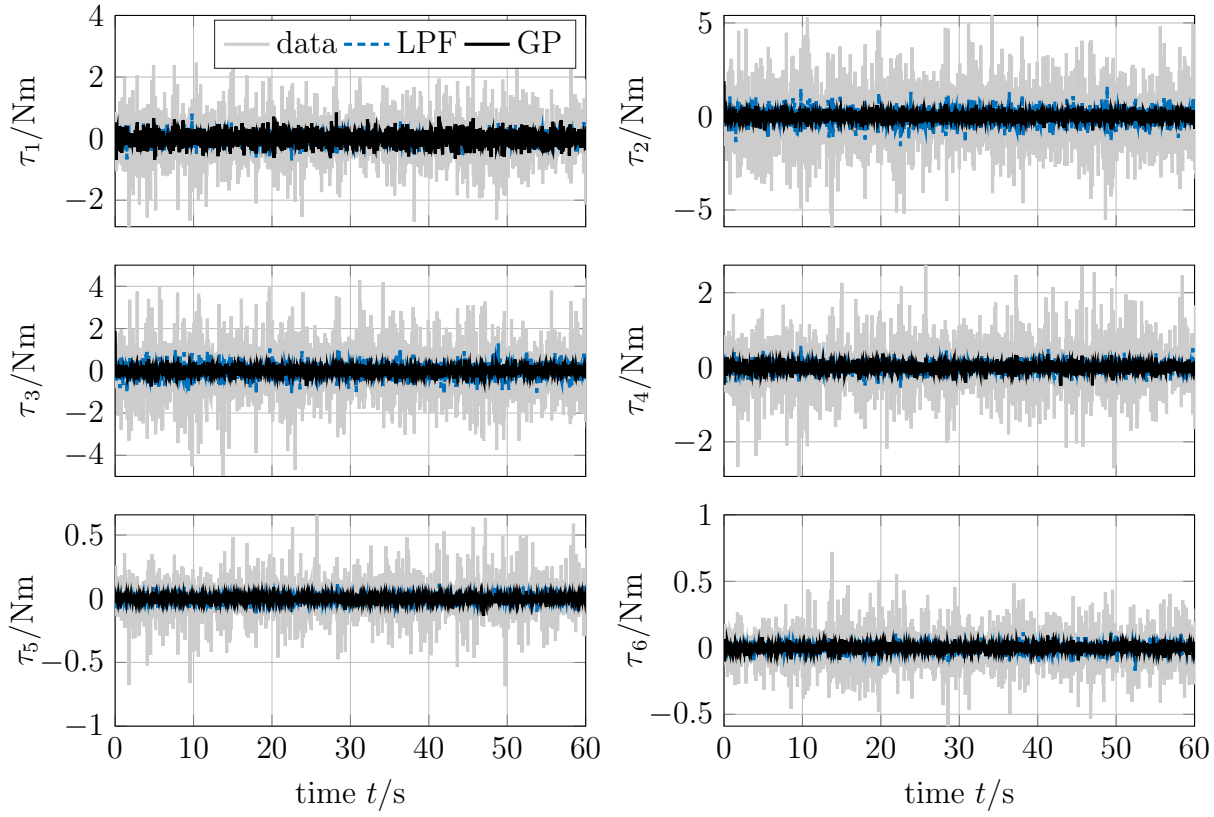


Figure A.17: Joint torques in simulation.

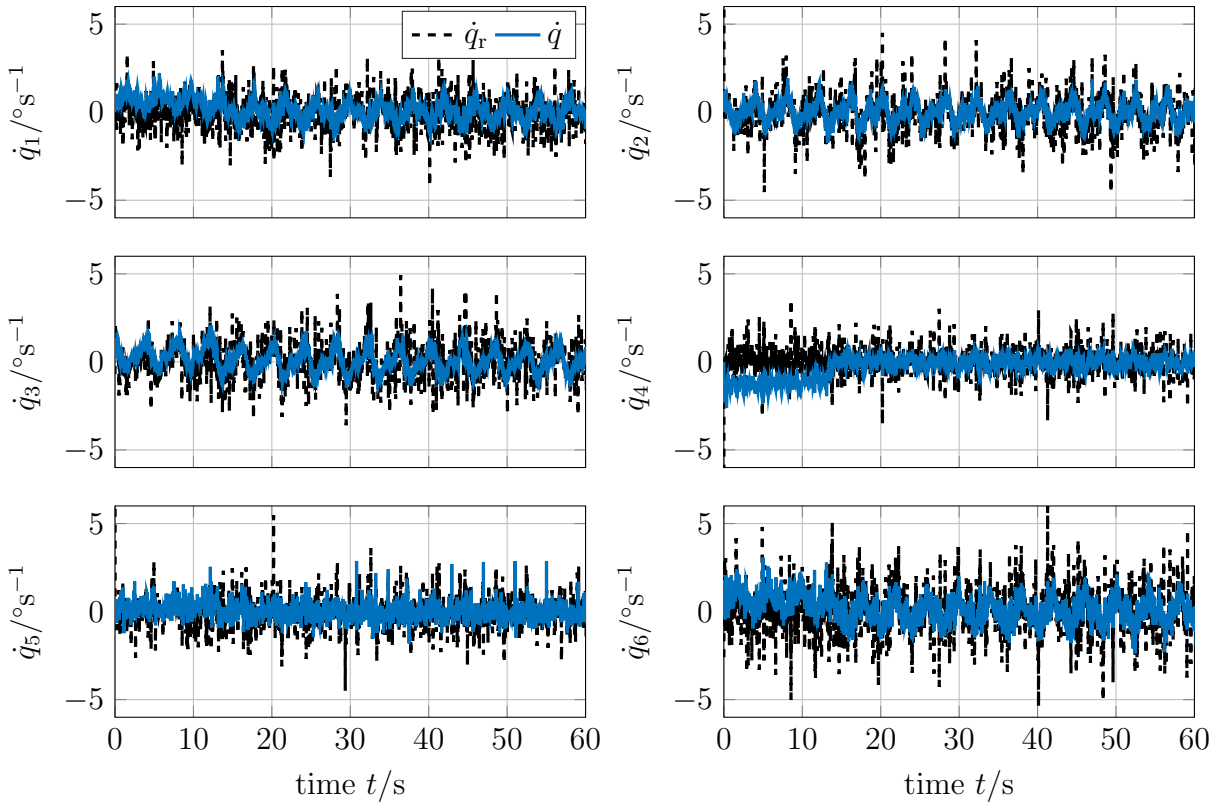


Figure A.18: Angular velocities in simulation.

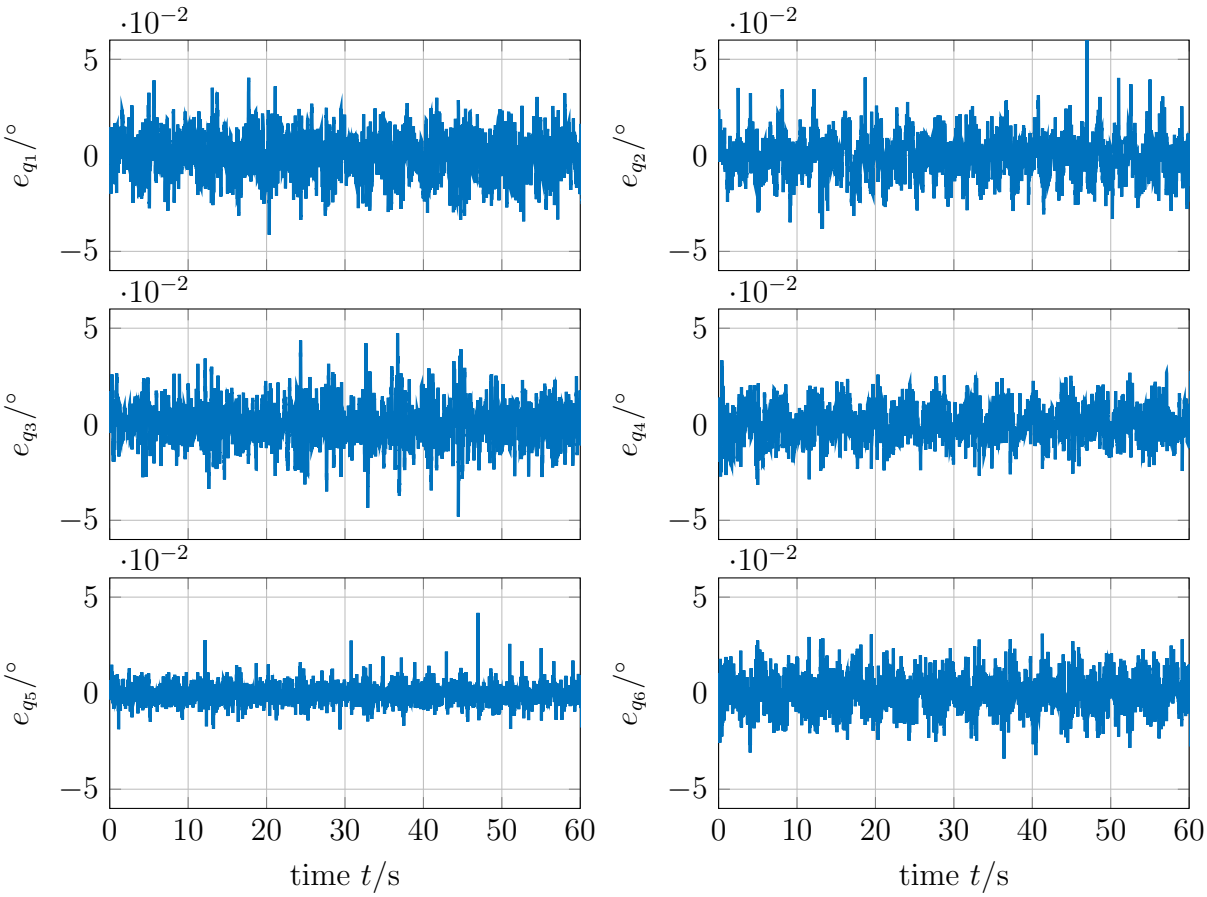


Figure A.19: Angular position errors for motion compensation in simulation.

joint torques according to the diagram in Figure 6.7. The angular velocities for the GP supported MPC are depicted in Figure A.18. The simulation control errors are depicted in Figure A.19. The scattering in the error origins from the online update of the reference with noisy data. The prediction data set D_{pred} is updated whenever new optical data is available and if this update allows for trackability according to Algorithm 3. Additionally to the control error comparison in Figure 6.9 we also wanted to show the error between the robot pose and the original unfiltered data. In general, the errors made by the GP, the inverse kinematics, the controller, and the linearisation might add up. Hence, the errors $\tilde{e}_i$ with $i \in \{\text{x,y,z,A,B,C}\}$ of the robot pose with respect to the data are of additional interest. Figure A.20 displays the errors $\tilde{e}$ between $h_{\text{fk}}(q)$ and the data instead of the control errors e between $h_{\text{fk}}(q)$ and $h_{\text{fk}}(q_{\text{r}})$. Most of the curves in Figure A.20 only show the noise of the data instead of an actual error. However, the MPC based on a low pass filter is performing significantly worse than the GP-based MPC. This originates from the delay the low pass filter introduces. Hence, even if the control error in the MPC is low, the error of the filter adds on top leading to larger deviations from the actual patient position. The GP supported MPC does not suffer from this issue, due to the smooth predictions provided by the GP as a reference to the MPC.

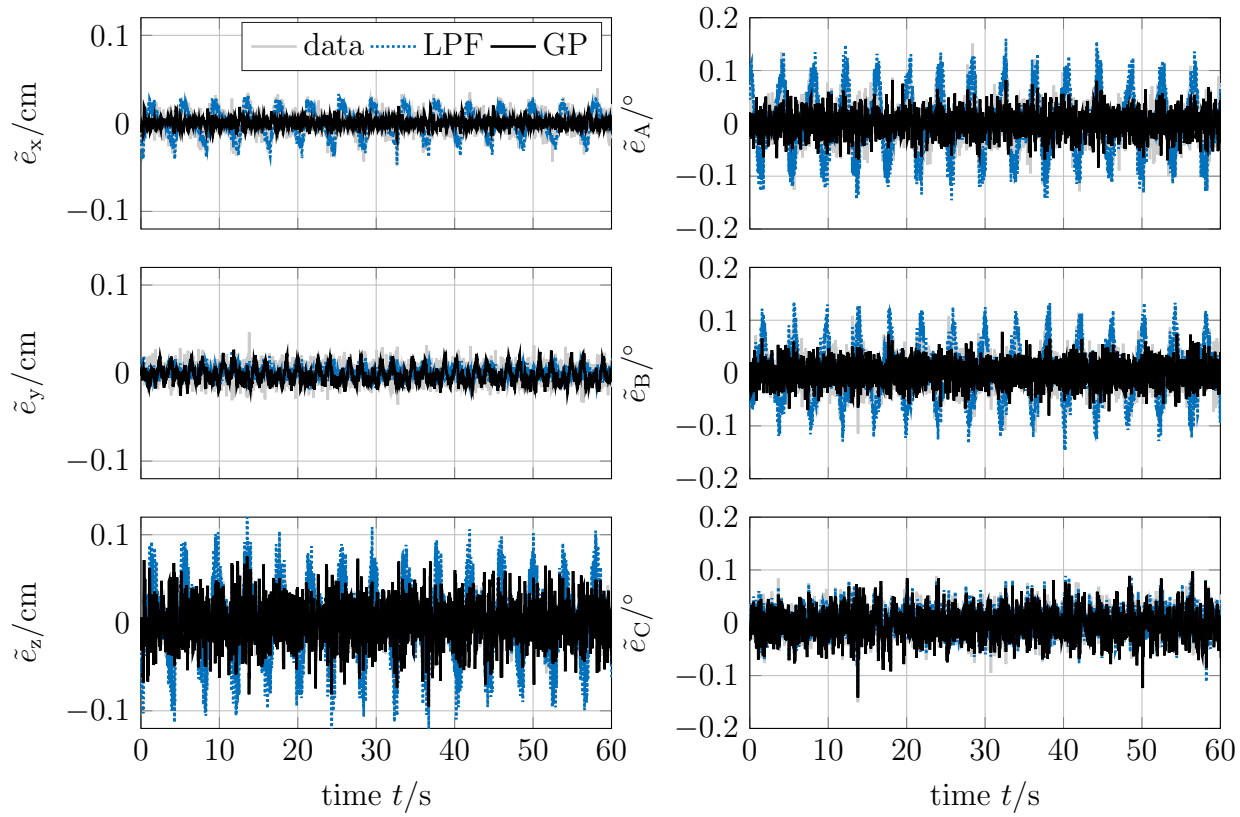


Figure A.20: Errors of robot position compared to original data.

Supplementary plots for the experimental evaluation of the learning supported MPC scheme are provided in Figures A.21-A.23. In Figure A.21, the angular velocities (blue) as well as the corresponding references (black) are shown. Even though the effect of the noise on the references is about one magnitude smaller compared to the unfiltered data, it is still noisy. Nevertheless, the controller tuning emphasises the minimisation of the joint angle error more than the joint velocity error by the chosen weights. Hence, the actual robot motions shows slightly smoother curves than its reference. The optimal inputs are shown in Figure A.22. They are magnitudes larger than the simulated inputs in Figure A.16. This origins from the model-plant mismatch. Since the robot at each sampling is not exactly on the reference, the controller chooses large inputs to drive the system to its reference. Nevertheless, all constraints for the control inputs are satisfied. The corresponding joint torques send to the robot are depicted in Figure A.23. These are larger than the torques in the simulations to overcome the uncompensated friction, cf. Figure A.17. Hence, the model predictive controller shows a certain inherent robustness despite the existing model-plant mismatch.

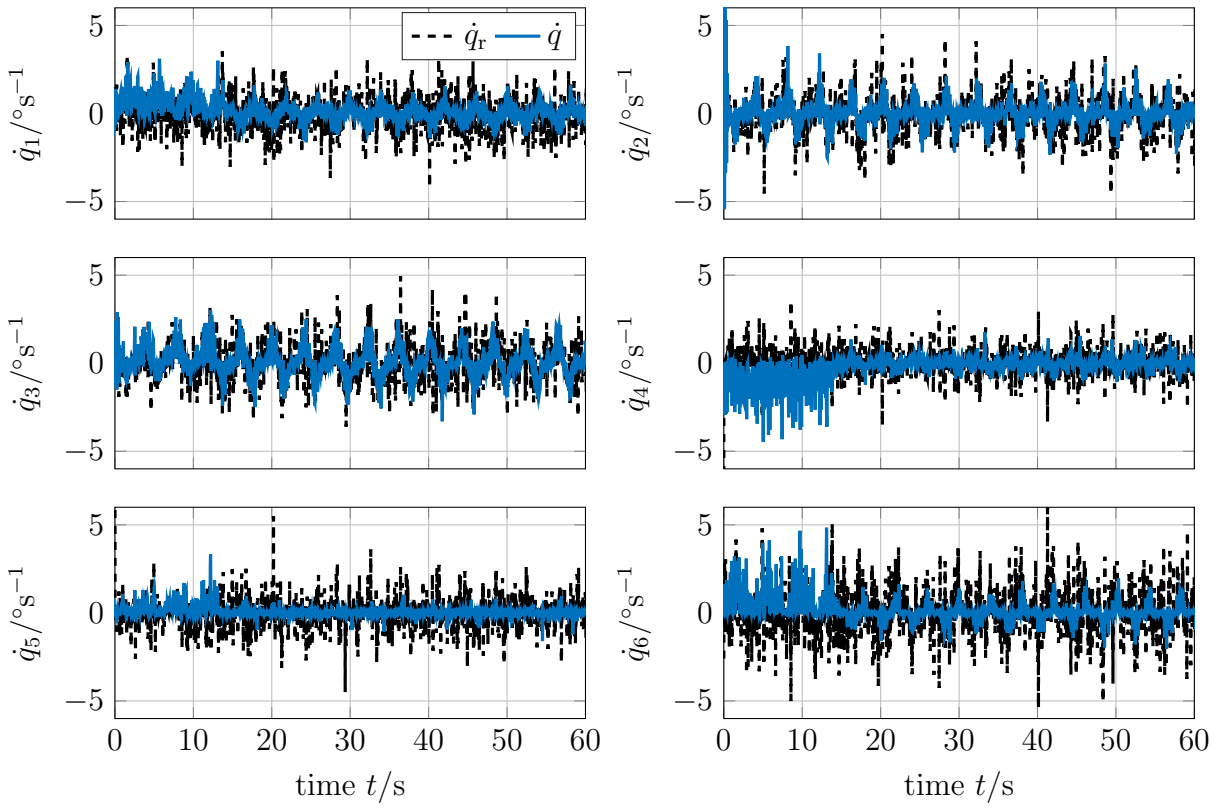


Figure A.21: Angular velocities in experiment.

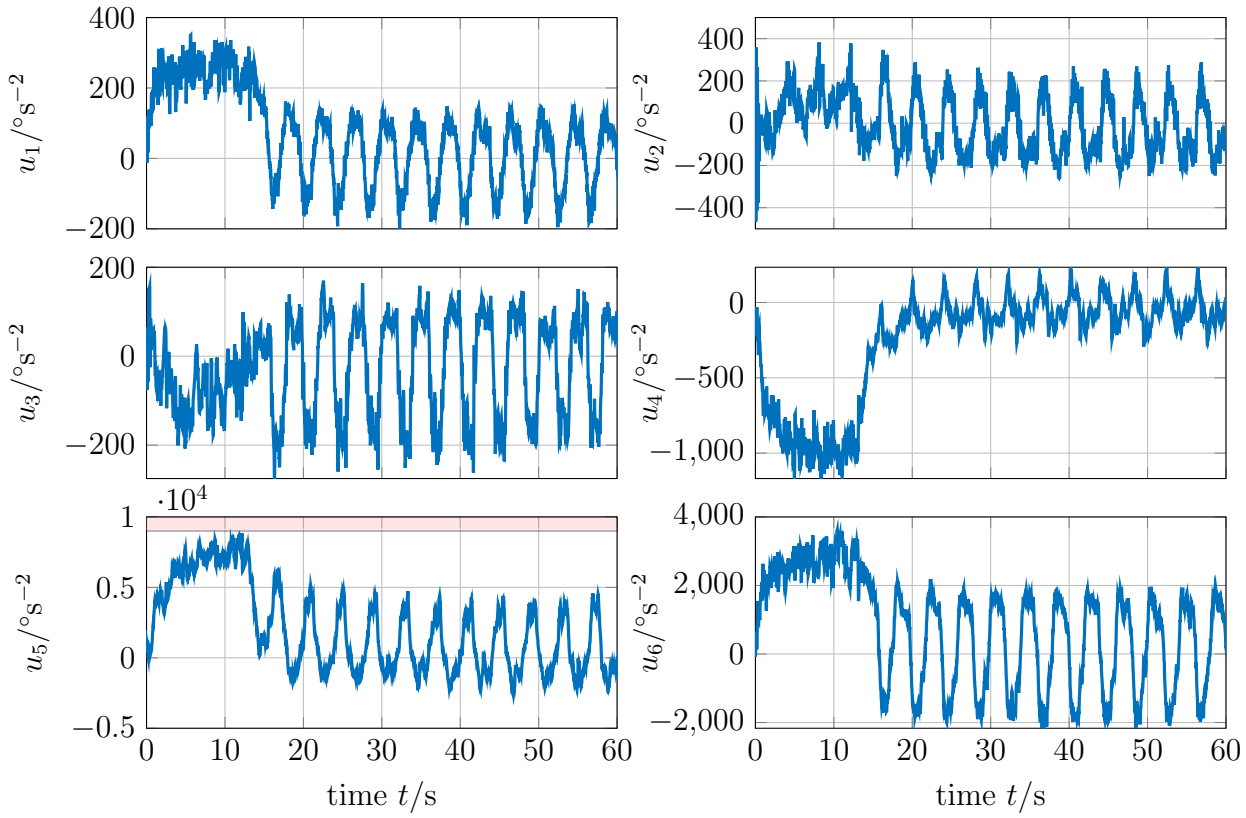


Figure A.22: Angular accelerations as optimal inputs for linearised system in experiment.

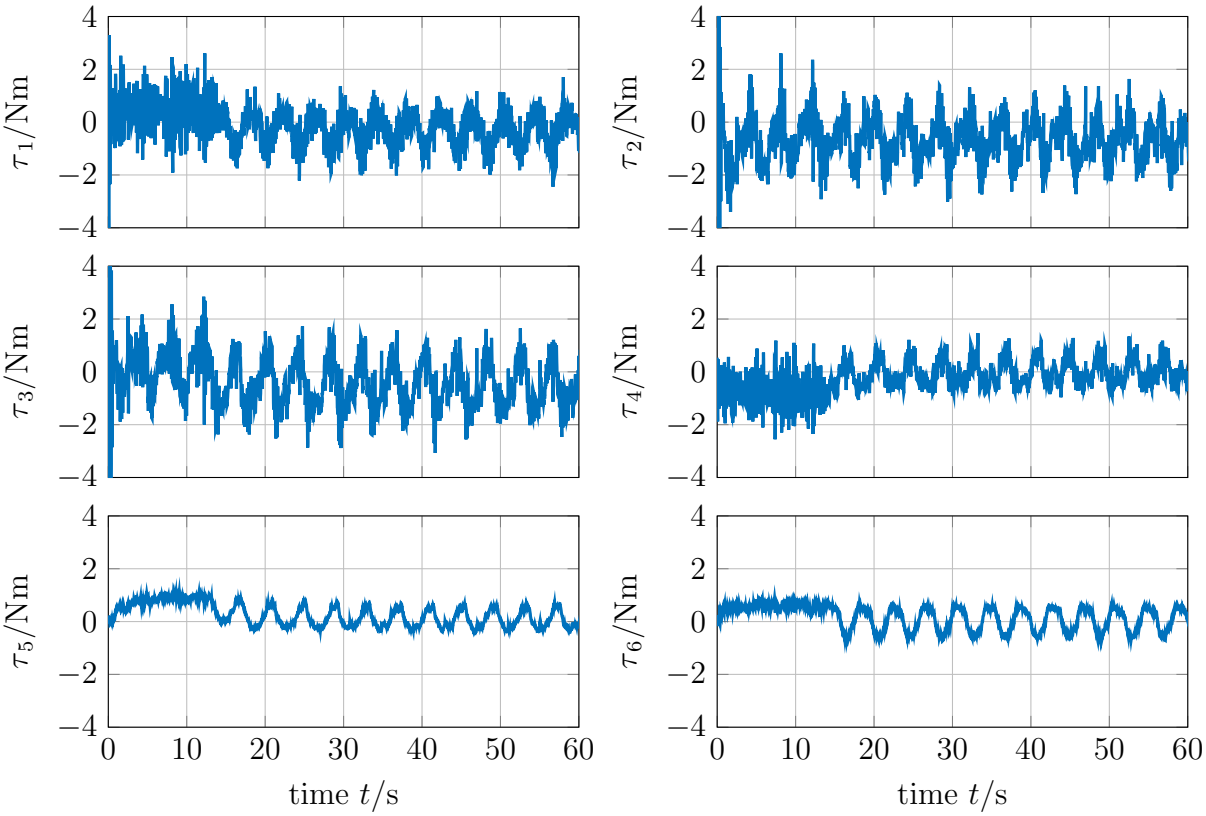


Figure A.23: Joint torques in experiment.

Bibliography

[1] C. Agrell. Gaussian processes with linear operator inequality constraints. *arXiv preprint arXiv:1901.03134*, 2019.

[2] A. Aguiar and J. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, 52(8):1362–1379, 2007.

[3] A. Akametalu, J. Fisac, J. Gillula, S. Kaynama, M. Zeilinger, and C. Tomlin. Reachability-based safe learning with Gaussian processes. In *Conference on Decision and Control (CDC)*, pages 1424–1431. IEEE, 2014.

[4] F. Allgöwer, R. Findeisen, and C. Ebenbauer. *Nonlinear model predictive control*. Springer Basel, 2000.

[5] I. Altrogge, T. Preusser, T. Kröger, S. Haase, T. Patz, and M. Kirby. Sensitivity analysis for the optimization of radiofrequency ablation in the presence of material parameter uncertainty. *International Journal for Uncertainty Quantification*, 2(3):295–321, 2012.

[6] H. Arenbeck, L. Wittschier, D. Kügler, and D. Abel. Control methods for robot-based predictive compensation of respiratory motion. *Biomedical Signal Processing and Control*, 34:16–24, 2017.

[7] D. Ariens, H. J. Ferreau, B. Houska, and F. Logist. ACADO for Matlab User's Manual. `http://acado.sourceforge.net/doc/pdf/acado_matlab_manual.pdf`, September 2020.

[8] A. Aswani, H. Gonzalez, S. Sastry, and C. Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[9] A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin. Reducing transient and steady state electricity consumption in HVAC using learning-based model-predictive control. *Proceedings of the IEEE*, 100(1):240–253, 2011.

[10] K. Ažman and J. Kocijan. Non-linear model predictive control for models with local information and uncertainties. *Transactions of the Institute of Measurement and Control*, 30(5):371–396, 2008.

[11] V. Bargsten, P. Zometa, and R. Findeisen. Modeling, parameter identification and model-based control of a lightweight robotic manipulator. In *International Conference on Control Applications (CCA)*, pages 134–139. IEEE, 2013.

[12] Barrett Technology Inc. Barrett hand and force/torque sensor. `https://advanced.barrett.com/barretthand`, June 2021.

[13] T. Beckers, D. Kulić, and S. Hirche. Stable Gaussian process based tracking control of Euler–Lagrange systems. *Automatica*, 103:390–397, 2019.

[14] M. Bednarczyk, H. Omran, and B. Bayle. Model predictive impedance control. In *International Conference on Robotics and Automation (ICRA)*, pages 4702–4708, 2020.

[15] A. Bemporad, D. Bernardini, R. Long, and J. Verdejo. Model predictive control of turbocharged gasoline engines for mass production. Technical report, SAE Technical Paper, 2018.

[16] F. Berkenkamp and A. Schöllig. Safe and robust learning control with Gaussian processes. In *European Control Conference (ECC)*, pages 2496–2501, 2015.

[17] A. Bertelsen, J. Melo, E. Sánchez, and D. Borro. A review of surgical robots for spinal interventions. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 9(4):407–422, 2013.

[18] J. Bethge. Entwurf und Implementierung einer modellprädiktiven Kraft- und Pfadverfolgungsregelung für Kontaktszenarien in der Robotik. Master's thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2016.

[19] J. Bethge, B. Morabito, J. Matschek, and R. Findeisen. Multi-mode learning supported model predictive control with guarantees. In *Nonlinear Model Predictive Control Conference (NMPC)*, pages 517–522. Elsevier, 2018.

[20] F. Blanchini and S. Miani. *Set-theoretic Methods in Control*. Springer, 2008.

[21] G. Bledt and S. Kim. Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 6316–6323. IEEE, 2019.

[22] P. Bouffard, A. Aswani, and C. Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *International Conference on Robotics and Automation (ICRA)*, pages 279–284. IEEE, 2012.

[23] P. Boyle and M. Frean. Dependent Gaussian processes. *Advances in neural information processing systems*, 17:217–224, 2004.

[24] W. Bukhari and S. Hong. Real-time prediction and gating of respiratory motion using an extended Kalman filter and Gaussian process regression. *Physics in Medicine & Biology*, 60(1):233, 2014.

[25] W. Bukhari and S. Hong. Real-time prediction and gating of respiratory motion in 3D space using extended Kalman filters and Gaussian process regression network. *Physics in Medicine & Biology*, 61(5):1947, 2016.

[26] M. Böck and A. Kugi. Real-time nonlinear model predictive path-following control of a laboratory tower crane. *IEEE Transactions on Control Systems Technology*, 22(4):1461–1473, 2014.

[27] S. Caccamo, Y. Bekiroglu, C. Ek, and D. Kragic. Active exploration using Gaussian random fields and Gaussian process implicit surfaces. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 582–589. IEEE, 2016.

[28] E. Camacho and C. Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[29] G. Cao, E. Lai, and F. Alam. Gaussian process model predictive control of an unmanned quadrotor. *Journal of Intelligent & Robotic Systems*, 88(1):147–162, 2017.

[30] M. Capocelli, L. De Santis, A. Maurizi, P. Pozzilli, and V. Piemonte. Model predictive control for the artificial pancreas. *Biomedical Engineering Challenges: A Chemical Engineering Insight*, 1:75–96, 2018.

[31] A. Capone, A. Lederer, J. Umlauft, and S. Hirche. Data selection for multi-task learning under dynamic constraints. *IEEE Control Systems Letters*, 5(3):959–964, 2020.

[32] E. Constantinescu and M. Anitescu. Physics-based covariance models for Gaussian processes with multiple outputs. *International Journal for Uncertainty Quantification*, 3(1):47–71, 2013.

[33] N. Cressie. *Statistics for Spatial Data*. Wiley Series in Probability and Statistics, 1993.

[34] L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[35] S. Da Veiga and A. Marrel. Gaussian process modeling with inequality constraints. *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 21(3):529–555, 2012.

[36] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart. Hybrid predictive control for aerial robotic physical interaction towards inspection operations. In *International Conference on Robotics and Automation (ICRA)*, pages 53–58. IEEE, 2014.

[37] D. Devito, L. Kaplan, R. Dietl, M. Pfeiffer, D. Horne, B. Silberstein, M. Hardenbrook, G. Kiriyanthan, Y. Barzilay, A. Bruskin, et al. Clinical acceptance and accuracy assessment of spinal implants guided with spineassist surgical robot: Retrospective study. *Spine*, 35(24):2109–2115, 2010.

[38] J. Di Carlo, P. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 7440–7447. IEEE, 2018.

[39] N. Diolaiti, C. Melchiorri, and S. Stramigioli. Contact impedance estimation for robotic systems. *IEEE Transactions on Robotics*, 21(5):925–935, 2005.

[40] M. Dominici and R. Cortesao. Model predictive control architectures with force feedback for robotic-assisted beating heart surgery. In *International Conference on Robotics and Automation (ICRA)*, pages 2276–2282. IEEE, 2014.

[41] F. Domroes, C. Krewet, and B. Kuhlenkoetter. Application and analysis of force control strategies to deburring and grinding. *Modern Mechanical Engineering*, 03(02):11–18, 2013.

[42] A. Draeger, S. Engell, and H. Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15(5):61–66, 1995.

[43] D. Dupuy, R. Hong, B. Oliver, and S. Goldberg. Radiofrequency ablation of spinal tumors: Temperature distribution in the spinal canal. *American Journal of Roentgenology*, 175(5):1263–1266, 2000.

[44] R. Dürichen, X. Fang, T. Wissel, and A. Schweikard. Gaussian process models for respiratory motion compensation. In *International Congress and Exhibition on Computer Assisted Radiology and Surgery (CARS'14), Fukuoka*, pages 286–287, 2014.

[45] R. Dürichen, M. Pimentel, L. Clifton, A. Schweikard, and D. Clifton. Multi-task Gaussian process models for biomedical applications. In *International Conference on Biomedical and Health Informatics (BHI)*, pages 492–495. IEEE, 2014.

[46] R. Dürichen, M. A. Pimentel, L. Clifton, A. Schweikard, and D. A. Clifton. Multitask Gaussian processes for multivariate physiological time-series analysis. *IEEE Transactions on Biomedical Engineering*, 62(1):314–322, 2014.

[47] M. D'Souza, J. Gendreau, A. Feng, L. H. Kim, A. L. Ho, and A. Veeravagu. Robotic-assisted spine surgery: history, efficacy, cost, and future trends. *Robotic Surgery: Research and Reviews*, 6:9–23, 2019.

[48] M. El Ghoumari, H.-J. Tantau, and J. Serrano. Nonlinear constrained MPC: Real-time implementation of greenhouse air temperature control. *Computers and Electronics in Agriculture*, 49(3):345–356, 2005.

[49] Z. Erickson, H. Clever, G. Turk, C. Liu, and C. Kemp. Deep haptic model predictive control for robot-assisted dressing. In *International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[50] F. Ernst, R. Dürichen, A. Schlaefer, and A. Schweikard. Evaluating and comparing algorithms for respiratory motion prediction. *Physics in Medicine & Biology*, 58(11):3911, 2013.

[51] F. Ernst and A. Schweikard. Forecasting respiratory motion with accurate online support vector regression (svrpred). *International Journal of Computer Assisted Radiology and Surgery*, 4(5):439–447, 2009.

[52] F. Ernst and A. Schweikard. A survey of algorithms for respiratory motion prediction in robotic radiosurgery. *Informatik 2009–Im Focus das Leben*, 2009.

[53] P. Falugi and D. Mayne. Tracking a periodic reference using nonlinear model

predictive control. In *Conference on Decision and Control (CDC)*, pages 5096–5100. IEEE, 2013.

[54] Q. Fan, X. Yu, Y. Zhao, and S. Yu. A respiratory motion prediction method based on improved relevance vector machine. *Mobile Networks and Applications*, pages 1–10, 2020.

[55] G. Fang, X. Wang, K. Wang, K.-H. Lee, J. Ho, H.-C. Fu, D. Fu, and K.-W. Kwok. Vision-based online learning kinematic control for soft robots using local Gaussian process regression. *IEEE Robotics and Automation Letters*, 4(2):1194–1201, 2019.

[56] T. Faulwasser. *Optimization-based Solutions to Constrained Trajectory-tracking and Path-following Problems*. Number 3 in Contributions in Systems Theory and Automatic Control. Shaker Verlag, 2013.

[57] T. Faulwasser and R. Findeisen. A model predictive control approach to trajectory tracking problems via time-varying level sets of Lyapunov functions. In *Joint Conference on Decision and Control (CDC) and European Control Conference (ECC)*, pages 3381–3386, 2011.

[58] T. Faulwasser and R. Findeisen. Nonlinear model predictive control for constrained output path following. *IEEE Transactions on Automatic Control*, 61(4):1026–1039, 2016.

[59] T. Faulwasser, V. Hagenmeyer, and R. Findeisen. Constrained reachability and trajectory generation for flat systems. *Automatica*, 50(4):1151–1159, 2014.

[60] T. Faulwasser, J. Matschek, P. Zometa, and R. Findeisen. Predictive path-following control: Concept and implementation for an industrial robot. In *International Conference on Control Applications (CCA)*, pages 128–133. IEEE, 2013.

[61] T. Faulwasser, T. Weber, J. P. Zometa, and R. Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2016.

[62] A. Ferramosca, D. Limón, I. Alvarado, T. Alamo, and E. F. Camacho. MPC for tracking with optimal closed-loop performance. In *Conference on Decision and Control (CDC)*, pages 4055–4060. IEEE, 2008.

[63] R. Findeisen. *Nonlinear Model Predictive Control: A Sampled-Data Feedback Perspective*. Fortschritt-Berichte VDI, Reihe 8, Nr. 1087. VDI Verlag, Düsseldorf, 2006.

[64] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, volume 11, pages 119–141. Technische Universiteit Eindhoven, The Netherlands, 2002.

[65] R. Findeisen, H. Chen, and F. Allgöwer. Nonlinear preditive control for setpoint families. In *American Control Conference (ACC)*, pages 260–264, 2000.

[66] J. Fisac, A. Akametalu, M. Zeilinger, S. Kaynama, J. Gillula, and C. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, pages 2737–2752, 2018.

[67] P. Flores and H. M. Lankarani. *Contact Force Models for Multibody Dynamics*, volume 226. Springer, 2016.

[68] Franka Emika Panda data sheet: Robot arm and control. `https://s3-eu-central-1.amazonaws.com/franka-de-uploads/uploads/Datasheet-EN.pdf`, June 2021.

[69] Picture source for Franka Emika Panda and Vertebral Body. `https://wiredworkers.io/product/franka-emika-panda/,https://en.wikipedia.org/wiki/Lumbar_vertebrae`, June 2021.

[70] M. Gabiccini, A. Artoni, G. Pannocchia, and J. Gillis. A computational framework for environment-aware robotic manipulation planning. In *Robotics Research*, pages 363–385. Springer, 2018.

[71] J. Gangloff, R. Ginhoux, M. de Mathelin, L. Soler, and J. Marescaux. Model predictive control for compensation of cyclic organ motions in teleoperated laparoscopic surgery. *IEEE Transactions on Control Systems Technology*, 14(2):235–246, 2006.

[72] E. Garone, S. Di Cairano, and I. Kolmanovsky. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 2017.

[73] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca. Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization. *IEEE Robotics and Automation Letters*, 4(4):4147–4154, 2019.

[74] A. Gazis, O. Beuing, J. Franke, B. Jöllenbeck, and M. Skalej. Bipolar radiofrequency ablation of spinal tumors: predictability, safety and outcome. *The Spine Journal*, 14(4):604–608, 2014.

[75] A. Geist and S. Trimpe. Learning constrained dynamics with Gauss principle adhering Gaussian processes. *arXiv preprint arXiv:2004.11238*, 2020.

[76] M. Gillespie, C. Best, E. Townsend, D. Wingate, and M. Killpack. Learning nonlinear dynamic models of soft robots for model predictive control with neural networks. In *International Conference on Soft Robotics (RoboSoft)*, pages 39–45. IEEE, 2018.

[77] R. Ginhoux, J. Gangloff, M. de Mathelin, L. Soler, M. M. A. Sanchez, and J. Marescaux. Active filtering of physiological motion in robotized surgery using predictive control. *IEEE Transactions on Robotics*, 21(1):67–79, 2005.

[78] A. Girard. *Approximate methods for propagation of uncertainty with Gaussian process models*. PhD thesis, University of Glasgow, 2004.

[79] A. Girard, C. Rasmussen, and R. Murray-Smith. Multiple-step ahead prediction for non linear dynamic systems–a Gaussian process treatment with propagation of the uncertainty. *Advances in Neural Information Processing Systems*, 15:529–536, 2002.

[80] T. Gold, A. Völz, and K. Graichen. Model predictive interaction control for industrial robots. In *IFAC World Congress*, pages 10026–10033, 2020.

[81] T. Gold, A. Völz, and K. Graichen. Model predictive position and force trajectory tracking control for robot-environment interaction. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 7397–7402, 2020.

[82] A. Grancharova, J. Kocijan, and T. Johansen. Explicit stochastic predictive control of combustion plants based on Gaussian process models. *Automatica*, 44(6):1621–1631, 2008.

[83] L. Grüne and J. Pannek. *Nonlinear model predictive control*. Springer, 2017.

[84] L. Grüne. Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems. *SIAM Journal on Control and Optimization*, 48(2):1206–1228, 2009.

[85] W. Gueaieb, F. Karray, and S. Al-Sharhan. A robust adaptive fuzzy position/force control scheme for cooperative manipulators. *IEEE Transactions on Control Systems Technology*, 11(4):516–528, 2003.

[86] A. Haddadi and K. Hashtrudi-Zaad. Real-time identification of Hunt–Crossley dynamic models of contact environments. *IEEE Transactions on Robotics*, 28(3):555–566, 2012.

[87] M. Hanses, S. Adler, S. Wolff, M. Skalej, and N. Elkmann. Robotic assistance for spine interventions. In *Annual meeting of the German Society for Computer and Robot Assisted Surgery (CURAC)*, pages 231–236, 2016.

[88] T. Heirung, J. Paulson, J. O'Leary, and A. Mesbah. Stochastic model predictive control—how does it work? *Computers & Chemical Engineering*, 114:158–170, 2018.

[89] C. Herrmann, L. Ma, and K. Schilling. Model predictive control for tumor motion compensation in robot assisted radiotherapy. *IFAC World Congress*, 44(1):5968–5973, 2011.

[90] C. Herrmann and K. Schilling. Improving patient comfort using model predictive control in robot-assisted radiotherapy. In *International Conference on Robotics and Automation (ICRA)*, pages 5446–5452. IEEE, 2013.

[91] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

[92] L. Hewing, E. Arcari, L. Fröhlich, and M. N. Zeilinger. On simulation and trajectory prediction with Gaussian process dynamics. In *Conference on Learning*

for Dynamics and Control (L4DC), pages 1341–1348, 2020.

[93] L. Hewing, J. Kabzan, and M. Zeilinger. Cautious model predictive control using Gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736 –2743, 2019.

[94] L. Hewing, A. Liniger, and M. Zeilinger. Cautious NMPC with Gaussian process dynamics for autonomous miniature race cars. In *European Control Conference (ECC)*, pages 1341–1348, 2018.

[95] S. Hirche and M. Buss. Human-oriented control for haptic teleoperation. *Proceedings of the IEEE*, 100(3):623–647, 2012.

[96] P. Holzmann. Model-based position and force control for robotic manipulators. Master's thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2020.

[97] T. Horii, F. Giovannini, Y. Nagai, L. Natale, G. Metta, and M. Asada. Contact force estimation from flexible tactile sensor values considering hysteresis by Gaussian process. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 137–138. IEEE, 2014.

[98] R. Hovorka, V. Canonico, L. Chassin, U. Haueter, M. Massi-Benedetti, M. Federici, T. Pieber, H. Schaller, L. Schaupp, T. Vering, and M. Wilinska. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological Measurement*, 25(4):905–920, 2004.

[99] J. Huang, Y. Cao, C. Xiong, and H.-T. Zhang. An echo state Gaussian process-based nonlinear model predictive control for pneumatic muscle actuators. *IEEE Transactions on Automation Science and Engineering*, 16(3):1071–1084, 2018.

[100] Y. Huang, H. Wang, A. Khajepour, H. He, and J. Ji. Model predictive control power management strategies for HEVs: A review. *Journal of Power Sources*, 341:91–106, 2017.

[101] M. Ibrahim, J. Matschek, B. Morabito, and R. Findeisen. Hierarchical model predictive control for autonomous vehicle area coverage. In *IFAC Symposium on Automatic Control in Aerospace (ACA)*, pages 79–84, 2019.

[102] M. Ibrahim, J. Matschek, B. Morabito, and R. Findeisen. Improved area covering in dynamic environments by nonlinear model predictive path following control. In *IFAC Symposium on Mechatronic Systems*, pages 418–423, 2019.

[103] A. Isidori. *Nonlinear Control Systems*. Springer Science & Business Media, 2013.

[104] A. Jadbabaie and J. Hauser. On the stability of receding horizon control with a general terminal cost. *IEEE Transactions on Automatic Control*, 50(5):674–678, 2005.

[105] X. Jiang, J. Gao, X. Hong, and Z. Cai. Gaussian processes autoencoder for dimensionality reduction. In *Pacific-asia conference on knowledge discovery and data mining*, pages 62–73. Springer, 2014.

[106] C. Jidling, N. Wahlström, A. Wills, and T. B. Schön. Linearly constrained

Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1215–1224, 2017.

[107] A. Jöhl, S. Ehrbar, M. Guckenberger, S. Klöck, M. Meboldt, M. Zeilinger, S. Tanadini-Lang, and M. Schmid Daners. Performance comparison of prediction filters for respiratory motion tracking in radiotherapy. *Medical Physics*, 47(2):643–650, 2020.

[108] M. Kamel, M. Burri, and R. Siegwart. Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. In *IFAC World Congress*, pages 3518–3524, 2017.

[109] K. Kazim, J. Bethge, J. Matschek, and R. Findeisen. Combined predictive path following and admittance control. In *American Control Conference (ACC)*, pages 3153–3158. IEEE, 2018.

[110] B. Kern, C. Böhm, R. Findeisen, and F. Allgöwer. Receding horizon control for linear periodic time-varying systems subject to input constraints. In L. Magni, D. M. Raimondo, and F. Allgöwer, editors, *Nonlinear Model Predictive Control*, pages 109–117. Springer, 2009.

[111] M. Killpack, A. Kapusta, and C. Kemp. Model predictive control for fast reaching in clutter. *Autonomous Robots*, 40(3):537–560, 2016.

[112] M. Killpack and C. Kemp. Fast reaching in clutter while regulating forces using model predictive control. In *International Conference on Humanoid Robots (Humanoids)*, pages 146–153. IEEE, 2013.

[113] M. Kimmel and S. Hirche. Invariance control for safe human–robot interaction in dynamic environments. *IEEE Transactions on Robotics*, 33(6):1327–1342, 2017.

[114] E. Klenske, M. Zeilinger, B. Schölkopf, and P. Hennig. Gaussian process-based predictive control for periodic error correction. *IEEE Transactions on Control Systems Technology*, 24(1):110–121, 2016.

[115] S. Klose, A. Wahrburg, D. Clever, and H. Ding. A general admittance control approach for indirect force control of industrial manipulators. In *European Control Conference (ECC)*, pages 261–267, 2018.

[116] B. Kocer, T. Tjahjowidodo, and G. Seet. Model predictive UAV-tool interaction control enhanced by external forces. *Mechatronics*, 58:47–57, 2019.

[117] J. Kocijan. *Modelling and Control of Dynamic Systems using Gaussian Process Models*. Springer, 2016.

[118] J. Kocijan and R. Murray-Smith. Nonlinear predictive control with a Gaussian process model. *Lecture Notes in Computer Science*, 3355:185–200, 2005.

[119] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *American Control Conference (ACC)*, volume 3, pages 2214–2219. IEEE, 2004.

[120] J. Kocijan, R. Murray-Smith, C. Rasmussen, and B. Likar. Predictive control

with Gaussian process models. In *International Conference on Computer as a Tool*, volume 1, pages 352–356. IEEE, 2003.

[121] M. Kögel and R. Findeisen. Robust output feedback MPC for uncertain linear systems with reduced conservatism. In *IFAC World Congress*, pages 10685–10690, 2017.

[122] T. Koller, F. Berkenkamp, M. Turchetta, J. Bödecker, and A. Krause. Learning-based model predictive control for safe exploration and reinforcement learning. *arXiv preprint: arXiv:1906.12189*, 2019.

[123] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-based model predictive control for safe exploration. In *Conference on Decision and Control (CDC)*, pages 6059–6066. IEEE, 2018.

[124] F. Kühne, W. Lages, and J. da Silva Jr. Mobile robot trajectory tracking using model predictive control. In *Latin-American Robotics Symposium*, 2005.

[125] J. Ladoiye, D. Necsulescu, and J. Sasiadek. Control of force in surgical robots with random time delays using model predictive control. In *International Conference on Informatics in Control, Automation and Robotics*, pages 407–428. Springer, 2018.

[126] D. Lam, C. Manzie, and M. Good. Application of model predictive contouring control to an x-y table. In *IFAC World Congress*, pages 10325–10330, 2011.

[127] A. Lederer, A. Capone, J. Umlauft, and S. Hirche. How training data impacts performance in learning-based control. *IEEE Control Systems Letters*, 2020.

[128] A. Lederer, J. Umlauft, and S. Hirche. Uniform error bounds for Gaussian process regression with application to safe control. In *Advances in Neural Information Processing Systems*, pages 659–669, 2019.

[129] M. Lefranc and J. Peltier. Evaluation of the rosa™ spine robot for minimally invasive surgical procedures. *Expert Review of Medical Devices*, 13(10):899–906, 2016.

[130] I. Lenz, R. Knepper, and A. Saxena. DeepMPC: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, 2015.

[131] J. Levine. *Analysis and control of nonlinear systems: A flatness-based approach*. Springer Science & Business Media, 2009.

[132] B. Li, H. Jin, M. Fang, Y. Hu, and P. Zhang. Spinal physiological motion simulator and compensation method for a robotic spinal surgical system. In *International Conference on Robotics and Biomimetics (ROBIO)*, pages 229–234. IEEE, 2014.

[133] Y. Li and B. Hannaford. Gaussian process regression for sensorless grip force estimation of cable-driven elongated surgical instruments. *IEEE Robotics and Automation Letters*, 2(3):1312–1319, 2017.

[134] Z. Li, M. Wang, J. Ye, and H. Wu. A review of model based online identifica-

tion methods for robotic systems. In *Joint Symposium on Advanced Mechanical Science & Technology for Industrial Revolution 4.0*, pages 57–70. Springer, 2016.

[135] B. Likar and J. Kocijan. Predictive control of a gas–liquid separation plant based on a Gaussian process model. *Computers & Chemical Engineering*, 31(3):142–152, 2007.

[136] D. Limón, I. Alvarado, T. Alamo, and E. Camacho. Robust tube-based MPC for tracking of constrained linear systems with additive disturbances. *Journal of Process Control*, 20(3):248–260, 2010.

[137] D. Limón, T. Alamo, F. Salas, and E. Camacho. On the stability of constrained MPC without terminal constraint. *IEEE Transactions on Automatic Control*, 51(5):832–836, 2006.

[138] D. Limón, J. Calliess, and J. Maciejowski. Learning-based nonlinear model predictive control. *IFAC World Congress*, 50(1):7769–7776, 2017.

[139] D. Limón and T. Alamo. Tracking model predictive control. In J. Baillieul and T. Samad, editors, *Encyclopedia of Systems and Control*, pages 1475–1484. Springer, 2015.

[140] D. Limón, I. Alvarado, T. Alamo, and E. Camacho. MPC for tracking piecewise constant references for constrained linear systems. *Automatica*, 44(9):2382–2387, 2008.

[141] D. Limón, M. Pereira, D. M. de la Peña, T. Alamo, C. Jones, and M. Zeilinger. MPC for tracking periodic references. *IEEE Transactions on Automatic Control*, 61(4):1123–1128, 2016.

[142] H. Lin, C. Shi, B. Wang, M. Chan, X. Tang, and W. Ji. Towards real-time respiratory motion prediction based on long short-term memory neural networks. *Physics in Medicine & Biology*, 64(8):085010, 2019.

[143] G. Ling, K. Lindsten, O. Ljungqvist, J. Löfberg, C. Norén, and C. Larsson. Fuel-efficient model predictive control for heavy duty vehicle platooning using neural networks. In *American Control Conference (ACC)*, pages 3994–4001. IEEE, 2018.

[144] Y. Liu, C. Zeng, M. Fan, L. Hu, C. Ma, and W. Tian. Assessment of respiration-induced vertebral motion in prone-positioned patients during general anaesthesia. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 12(2):214–218, 2016.

[145] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgöwer. Constraint-tightening and stability in stochastic model predictive control. *IEEE Transactions on Automatic Control*, 62(7):3165–3177, 2016.

[146] U. Maeder and M. Morari. Offset-free reference tracking with model predictive control. *Automatica*, 46(9):1469–1476, 2010.

[147] L. Magni, G. De Nicolao, and R. Scattolini. Output feedback and tracking of

nonlinear systems with model predictive control. *Automatica*, 37(10):1601–1607, 2001.

[148] J. Mahler, S. Krishnan, M. Laskey, S. Sen, A. Murali, B. Kehoe, S. Patil, J. Wang, M. Franklin, P. Abbeel, et al. Learning accurate kinematic control of cable-driven surgical robots using data cleaning and Gaussian process regression. In *International Conference on Automation Science and Engineering (CASE)*, pages 532–539. IEEE, 2014.

[149] M. Maiworm, D. Limón, J. Manzano, and R. Findeisen. Stability of Gaussian process learning based output feedback model predictive control. In *Conference on Nonlinear Model Predictive Control (NMPC)*, pages 455–461, 2018.

[150] M. Maiworm, C. Wagner, R. Temirov, F. Tautz, and R. Findeisen. Two-degree-of-freedom control combining machine learning and extremum seeking for fast scanning quantum dot microscopy. In *American Control Conference (ACC)*, pages 4360–4366. IEEE, 2018.

[151] J. Manzano, D. Muñoz de la Peña, J. Calliess, and D. Limón. Online learning constrained model predictive control based on double prediction. *International Journal of Robust and Nonlinear Control*, 2020.

[152] T. Marlin. *Process control: Designing Processes and Control Systems for Dynamic Performance*. Chemical engineering series. McGraw-Hill, 2nd edition, 2000.

[153] J. Matschek, T. Bäthge, T. Faulwasser, and R. Findeisen. Nonlinear predictive control for trajectory tracking and path following: An introduction and perspective. In *Handbook of Model Predictive Control*, pages 169–198. Springer, 2019.

[154] J. Matschek, J. Bethge, P. Zometa, and R. Findeisen. Force feedback and path following using predictive control: Concept and application to a lightweight robot. In *IFAC World Congress*, pages 10243–10248, 2017.

[155] J. Matschek, E. Bullinger, F. von Haeseler, M. Skalej, and R. Findeisen. Mathematical 3D modelling and sensitivity analysis of multipolar radiofrequency ablation in the spine. *Mathematical Biosciences*, 284:51–60, 2017.

[156] J. Matschek and R. Findeisen. Learning supported model predictive control for tracking of periodic references. In *Proceedings of Machine Learning Research*, pages 511–520, 2020.

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

[158] J. Matschek, A. Himmel, K. Sundmacher, and R. Findeisen. Constrained Gaussian process learning for model predictive control. In *IFAC World Congress*,

pages 989 –994, 2020.

[159] J. Matschek, A. Himmel, F. von Haeseler, E. Bullinger, M. Skalej, and R. Findeisen. Mathematical modelling and sensitivity analysis of multipolar radiofrequency ablation in the spine. *IFAC Symposium on Biological and Medical Systems (BMS)*, 48(20):243–248, 2015.

[160] J. Matschek, R. Jordanowa, and R. Findeisen. Direct robotic force control with learning supported model predictive control. In *Conference on Control Technology and Applications (CCTA)*, pages 8–13, 2020.

[161] D. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, 1990.

[162] D. Mayne, S. Raković, R. Findeisen, and F. Allgöwer. Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42(7):1217–1222, 2006.

[163] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[164] D. Mayne, M. Seron, and S. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[165] J. Medina, H. Börner, S. Endo, and S. Hirche. Impedance-based Gaussian processes for modeling human motor behavior in physical and non-physical interaction. *IEEE Transactions on Biomedical Engineering*, 66(9):2499–2511, 2019.

[166] J. Medina, T. Lorenz, and S. Hirche. Synthesizing anticipatory haptic assistance considering human behavior uncertainty. *IEEE Transactions on Robotics*, 31(1):180–190, 2015.

[167] A. Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[168] E. Mielke, E. Townsend, D. Wingate, and M. Killpack. Human-robot comanipulation of extended objects: Data-driven models and control from analysis of human-human dyads. *arXiv preprint arXiv:2001.00991*, 2020.

[169] I. Mitsioni, Y. Karayiannidis, J. A. Stork, and D. Kragic. Data-driven model predictive control for the contact-rich task of food cutting. In *International Conference on Humanoid Robots (Humanoids)*, pages 244–250. IEEE, 2019.

[170] D. Müller, A. Mayer, and O. Sawodny. Model predictive force control for robots in compliant environments with guaranteed maximum force. In *American Control Conference (ACC)*, pages 1355–1360. IEEE, 2019.

[171] R. Murray-Smith, D. Sbarbaro, C. Rasmussen, and A. Girard. Adaptive, cautious, predictive control with Gaussian Process priors. *IFAC Symposium on System Identification (SYSID)*, 36(16):1155–1160, 2003.

[172] C. Nadeau, A. Krupa, and J. Gangloff. Automatic tracking of an organ section

with an ultrasound probe: Compensation of respiratory motion. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 57–64. Springer, 2011.

[173] C. Nielsen and M. Maggiore. Output stabilization and maneuver regulation: A geometric approach. *Systems & Control Letters*, 55:418–427, 2006.

[174] C. Nielsen and M. Maggiore. On local transverse feedback linearization. *SIAM Journal on Control and Optimization*, 47:2227–2250, 2008.

[175] A. Nikou, C. Verginis, S. Heshmati-Alamdari, and D. Dimarogonas. A nonlinear model predictive control scheme for cooperative manipulation with singularity and collision avoidance. In *Mediterranean Conference on Control and Automation (MED)*, pages 707–712. IEEE, 2017.

[176] A. Okamura. Methods for haptic feedback in teleoperated robot-assisted surgery. *Industrial Robot: An International Journal*, 31(6):499–508, 2004.

[177] L. Ortmann, D. Shi, E. Dassau, F. Doyle, S. Leonhardt, and B. Misgeld. Gaussian process-based model predictive control of blood glucose for patients with type 1 diabetes mellitus. In *Asian Control Conference (ASCC)*, pages 1092–1097. IEEE, 2017.

[178] C. Ostafew, A. Schöllig, and T. Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *International Conference on Robotics and Automation (ICRA)*, pages 4029–4036. IEEE, 2014.

[179] C. Ostafew, A. Schöllig, and T. Barfoot. Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. *The International Journal of Robotics Research*, 35(13):1547–1563, 2016.

[180] C. Ostafew, A. Schöllig, T. Barfoot, and J. Collier. Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *Journal of Field Robotics*, 33(1):133–152, 2016.

[181] D. Paluszczyszyn, P. Skworcow, O. Haas, K. Burnham, and J. Mills. Model predictive control for real-time tumor motion compensation in adaptive radiotherapy. *IEEE Transactions on Control Systems Technology*, 22(2):635–651, 2013.

[182] M. Pereira, D. M. de la Peña, D. Limón, I. Alvarado, and T. Alamo. Application to a drinking water network of robust periodic MPC. *Control Engineering Practice*, 57:50–60, 2016.

[183] C. Plagemann. *Gaussian processes for flexible robot learning.* PhD thesis, University of Freiburg, Department of Computer Science, 2008.

[184] L. Pöhler, J. Umlauft, and S. Hirche. Uncertainty-based human motion tracking with stable Gaussian process state space models. In *IFAC Conference on Cyber-Physical and Human Systems (CPHS)*, pages 8–14, 2019.

[185] J. Prüher and S. Särkkä. On the use of gradient information in Gaussian process quadratures. In *International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016.

[186] S. Qin and T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[187] D. Raimondo, D. Limón, M. Lazar, and E. Magni, Lalo Camacho. Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1):5–21, 2009.

[188] L. Ramrath, A. Schlaefer, F. Ernst, S. Dieterich, and A. Schweikard. Prediction of respiratory motion with a multi-frequency based extended Kalman filter. In *International Conference and Exhibition on Computer Assisted Radiology and Surgery (CARS)*, volume 21, pages 56–58, 2007.

[189] J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing Madison, WI, 2017.

[190] Y. Ren, Z. Chen, Y. Gu, Y. Liu, M. Jin, and H. Liu. Adaptive hybrid position force control of dual-arm cooperative manipulators with uncertain dynamics and closed-chain kinematics. *Journal of The Franklin Institute*, 354(17):7767–7793, 2017.

[191] S. Rhein, C. Oesterle, and K. Graichen. Optimal trajectory planning for interstitial hyperthermia processes. *IFAC Workshop on Control of Systems Governed by Partial Differential Equations (CPDE)*, 49(8):136–141, 2016.

[192] J. Riihimäki and A. Vehtari. Gaussian processes with monotonicity information. In *International Conference on Artificial Intelligence and Statistics*, pages 645–652, 2010.

[193] C. Riviere, A. Thakral, I. Iordachita, G. Mitroi, and D. Stoianovici. Predicting respiratory motion for active canceling during percutaneous needle insertion. In *International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 4, pages 3477–3480. IEEE, 2001.

[194] C. Riviere, J. Gangloff, and M. De Mathelin. Robotic compensation of biological motion to enhance surgical accuracy. *Proceedings of the IEEE*, 94(9):1705–1716, 2006.

[195] F. Roser, M. Tatagiba, and G. Maier. Spinal robotics: Current applications and future perspectives. *Neurosurgery*, 72:A12–A18, 2013.

[196] U. Rosolia and F. Borrelli. Learning model predictive control for iterative tasks: A computationally efficient approach for linear system. In *IFAC World Congress*, pages 3142–3147, 2017.

[197] U. Rosolia and F. Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

[198] U. Rosolia, A. Carvalho, and F. Borrelli. Autonomous racing using learning model predictive control. In *American Control Conference (ACC)*, pages 5115–5120. IEEE, 2017.

[199] U. Rosolia, X. Zhang, and F. Borrelli. Robust learning model predictive control for iterative tasks: Learning from experience. In *Conference on Decision and Control (CDC)*, pages 1157–1162. IEEE, 2017.

[200] R. Rothfuß. *Anwendung der flachheitsbasierten Analyse und Regelung nichtlinearer Mehrgrößensysteme*. VDI Fortschrittsberichte Reihe 8, Nr. 1087. VDI Verlag, 1997.

[201] M. Salzmann and R. Urtasun. Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 647–654. IEEE, 2010.

[202] M. Salzmann and R. Urtasun. Implicitly constrained Gaussian process regression for monocular non-rigid pose estimation. In *Advances in Neural Information Processing Systems*, pages 2065–2073, 2010.

[203] L. Santos, P. A. Afonso, J. Castro, N. Oliveira, and L. Biegler. On-line implementation of nonlinear MPC: An experimental case study. *Control Engineering Practice*, 9(8):847–857, 2001.

[204] S. Sayeh, J. Wang, W. Main, W. Kilby, and C. Maurer. Respiratory motion tracking for robotic radiosurgery. In *Treating Tumors that Move with Respiration*, pages 15–29. Springer, 2007.

[205] M. Schneider and W. Ertel. Robot learning by demonstration with local Gaussian process regression. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 255–260. IEEE, 2010.

[206] G. Schreiber, A. Stemmer, and R. Bischoff. The fast research interface for the KUKA lightweight robot. In *Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications (ICRA)*, pages 15–21, 2010.

[207] C. Schumann, C. Rieder, S. Haase, K. Teichert, P. Süss, P. Isfort, P. Bruners, and T. Preusser. Interactive multi-criteria planning for radiofrequency ablation. *International Journal of Computer Assisted Radiology and Surgery*, 10(6):879–889, 2015.

[208] C. Sferrazza, M. Muehlebach, and R. D'Andrea. Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control. In *Conference on Decision and Control (CDC)*, pages 5186–5192. IEEE, 2017.

[209] J. Sheng and M. Spong. Model predictive control for bilateral teleoperation systems with time delays. In *Canadian Conference on Electrical and Computer Engineering*, volume 4, pages 1877–1880. IEEE, 2004.

[210] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, 2010.

[211] B. Siciliano and L. Villani. *Robot Force Control*. Springer Science & Business Media, 2012.

[212] S. Sirouspour and A. Shahdi. Model predictive control for transparent teleoperation under communication time delay. *IEEE Transactions on Robotics*, 22(6):1131–1145, 2006.

[213] V. Skorokhod. *Basic principles and applications of probability theory*. Springer Science & Business Media, 2005.

[214] E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen. Derivative observations in Gaussian process models of dynamic systems. In *Advances in Neural Information Processing Systems*, pages 1057–1064, 2003.

[215] R. Soloperto, M. Müller, S. Trimpe, and F. Allgöwer. Learning-based robust model predictive control with state-dependent uncertainty. In *Conference on Nonlinear Model Predictive Control (NMPC)*, pages 442–447, 2018.

[216] A. Sorriento, M. B. Porfido, S. Mazzoleni, G. Calvosa, M. Tenucci, G. Ciuti, and P. Dario. Optical and electromagnetic tracking systems for biomedical applications: A critical review on potentialities and limitations. *IEEE Reviews in Biomedical Engineering*, 13:212–232, 2019.

[217] S. Stemmler, D. Abel, M. Schwenzer, O. Adams, and F. Klocke. Model predictive control for force control in milling. In *IFAC World Congress*, pages 15871–15876. Elsevier, 2017.

[218] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson. Force controlled robotic assembly without a force sensor. In *International Conference on Robotics and Automation (ICRA)*, pages 1538–1543. IEEE, 2012.

[219] A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel. Learning from the hindsight plan—episodic MPC improvement. In *International Conference on Robotics and Automation (ICRA)*, pages 336–343. IEEE, 2017.

[220] F. Tian, C. Lv, Z. Li, and G. Liu. Modeling and control of robotic automatic polishing for curved surfaces. *CIRP Journal of Manufacturing Science and Technology*, 14:55–64, 2016.

[221] M. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics*, pages 844–851, 2010.

[222] G. P. Tournier. *Six degrees of freedom estimation using monocular vision and Moiré patterns*. PhD thesis, Massachusetts Institute of Technology, 2006.

[223] J. Umlauft, T. Beckers, and S. Hirche. Scenario-based optimal control for Gaussian process state space models. In *European Control Conference (ECC)*, pages 1386–1392, 2018.

[224] J. Umlauft and S. Hirche. Feedback linearization based on Gaussian processes with event-triggered online learning. *IEEE Transactions on Automatic Control*, pages 4154–4169, 2019.

[225] J. Umlauft and S. Hirche. Learning stochastically stable Gaussian process state-space models. *IFAC Journal of Systems and Control*, 2020.

[226] J. Umlauft, A. Lederer, and S. Hirche. Learning stable Gaussian process state space models. In *American Control Conference (ACC)*, pages 1499–1504. IEEE, 2017.

[227] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl. acados: a modular open-source framework for fast embedded optimal control. *arXiv preprint arXiv:1910.13753*, 2019.

[228] L. Villani and J. De Schutter. Force control. In *Springer Handbook of Robotics*, pages 195–220. Springer, 2016.

[229] F. Viña, Y. Bekiroglu, C. Smith, Y. Karayiannidis, D. Kragic, et al. Predicting slippage and learning manipulation affordances through Gaussian process regression. In *International Conference on Humanoid Robots (Humanoids)*, pages 462–468. IEEE, 2013.

[230] A. Wahrburg and K. Listmann. MPC-based admittance control for robotic manipulators. In *Conference on Decision and Control (CDC)*, pages 7548–7554. IEEE, 2016.

[231] Y. Wang, C. Ocampo-Martinez, and V. Puig. Robust model predictive control based on Gaussian processes: Application to drinking water networks. In *European Control Conference (ECC)*, pages 3292–3297, 2015.

[232] Y. Wang, C. Ocampo-Martinez, and V. Puig. Stochastic model predictive control based on Gaussian processes applied to drinking water networks. *IET Control Theory & Applications*, 10(8):947–955, 2016.

[233] Y. Wang, C. Ocampo-Martínez, V. Puig, and J. Quevedo. Gaussian-process-based demand forecasting for predictive control of drinking water networks. In *International Conference on Critical Information Infrastructures Security*, pages 69–80. Springer, 2014.

[234] T. Weingartshofer, M. Schwegel, C. Hartl-Nesic, T. Glück, and A. Kugi. Collaborative synchronization of a 7-axis robot. In *IFAC Symposium on Mechatronic Systems (MECHATRONICS)*, pages 507–512, 2019.

[235] C. Williams and C. Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press Cambridge, MA, 2006.

[236] S. Xie and J. Ren. Iterative learning-based model predictive control for precise trajectory tracking of piezo nanopositioning stage. In *American Control Conference (ACC)*, pages 2922–2927. IEEE, 2018.

[237] X. Yang and J. Maciejowski. Fault tolerant control using Gaussian Processes and Model Predictive Control. *International Journal of Applied Mathematics and Computer Science*, 25(1):133–148, 2015.

[238] X. Yang and J. Maciejowski. Risk sensitive model predictive control with Gaussian process models. In *IFAC Symposium on System Identification (SYSID)*, pages 374–379, 2015.

[239] B. Zhang, M. Moser, E. Zhang, Y. Luo, C. Liu, and W. Zhang. A review of radiofrequency ablation: Large target tissue necrosis and mathematical modelling. *Physica Medica*, 32(8):961–971, 2016.

[240] R. Zhang, A. Xue, R. Lu, P. Li, and F. Gao. Real-Time Implementation of Improved State-Space MPC for Air Supply in a Coke Furnace. *IEEE Transactions on Industrial Electronics*, 61(7):3532–3539, 2014.

[241] W.-H. Zhu, S. Salcudean, S. Bachmann, and P. Abolmaesumi. Motion/force/image control of a diagnostic ultrasound robot. In *International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1580–1585. IEEE, 2000.

[242] G. Zorbas and T. Samaras. Simulation of radiofrequency ablation in real human anatomy. *International Journal of Hyperthermia*, 30(8):570–578, 2014.