Gaussian Processes in Control
–
Model Predictive Control with Guarantees and Control of Scanning Quantum Dot Microscopy

Dissertation
zur Erlangung des akademischen Grades

Doktoringenieur
(Dr.-Ing.)

von **M.Sc. M.Sc. Michael Maiworm**
geb. am 18.04.1985 in Olpe, Deutschland

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

Gutachter: Prof. Dr.-Ing. Rolf Findeisen
Prof. Dr.-Ing. Daniel Limón Marruedo
Dr. rer. nat. Christian Wagner

eingereicht am 30.06.2021
Promotionskolloquium am 28.07.2021

"All we have to decide is what to do
with the time that is given to us."

Gandalf, in "The Fellowship of the Ring" by J. R. R. Tolkien

Contents

Abstract

This thesis considers Gaussian processes and their applications in control theory. As a representative of many algorithms from machine learning, Gaussian processes are able to utilize the increasingly available data in today's digitalized and interconnected world. They allow to learn certain underlying relations and functional dependencies, which can then be leveraged for improved system understanding, control performance, and facilitate new applications.

The first part of this thesis uses dynamic Gaussian processes models in a model predictive control formulation denoted as recursive Gaussian process model predictive control (rGP-MPC). The key property of the approach is that it includes newly available data if it contributes sufficient novel information to the model. Based on new data, it recursively performs the necessary computations to reduce computational costs. The scheme is formulated in such a way that it can be applied to many processes, including those without accessible states, changing process conditions, or fast dynamics. Even though that the prediction model changes during operation, conditions are derived, which guarantee that the controller is nominally and inherently robustly stable. The approach is validated in simulations, illustrating increased performance due to the online learning approach, especially in situations where only limited training data and/or limited process knowledge is available. The derived feasibility and stability guarantees are not confined to rGP-MPC. They hold for a wide class of model predictive control approaches that use Gaussian processes and other machine learning algorithms, as long as the outlined conditions are satisfied.

Secondly, online learning of disturbance signals via Gaussian processes is considered and employed within a two-degree-of-freedom control approach for scanning quantum dot microscopy, which is a recently developed microscopy technique that generates images of electrostatic potentials of nanostructures such as atoms and molecules. The two-degree-of-freedom controller is the first advanced control scheme developed for scanning quantum dot microscopy. It comprises two feedback controllers that track specific electrical quantities during the scanning process. These quantities are used in post-processing to generate the final images. Improving the scanning/tracking speed and precision is of major importance. For this reason, two feedforward controllers are developed, of which one utilizes Gaussian processes. Both simulations and experiments validate the performance of the presented control approach. It increases the scan speed, renders the tracking more robust, and allows to scan larger surface areas than before.

Deutsche Kurzfassung

Diese Dissertation befasst sich mit der Anwendung und Theorie von Gaußprozessen in der Regelungstechnik. Als einer von vielen Algorithmen aus dem Bereich des maschinellen Lernens sind Gaußprozesse in der Lage verfügbare Informationen in den zunehmend anfallenden Messdaten der heutigen digitalisierten Welt zu nutzen um zu Grunde liegende funktionale Abhängigkeiten zu lernen. Dies lässt sich zur verbesserten Regelung bestehender Systeme oder für neue Anwendungen einsetzen.

Der erste Teil befasst sich mit recursive Gaussian process model predictive control (rGP-MPC), ein Regelungskonzept welches Prozessmodelle verwendet die auf Gaußprozessen basieren. Hierbei werden neu verfügbare Datenpunkte nur dann im Lernen berücksichtigt insofern diese ausreichend neue Informationen beinhalten. Die notwendigen Berechnungen werden rekursiv für eine effizientere Berechnung durchgeführt. rGP-MPC lässt sich für viele Anwendungen einsetzen, z. B. für Prozesse mit unvollständigen Zustandsinformationen, Prozesse mit sich ändernden Prozessbedingungen oder mit schnellen Dynamiken. Es werden Bedingungen angegeben unter denen das Regelungskonzept nominal sowie inhärent robust stabil ist, auch wenn sich das Gaußsche Prädiktionsmodell während des Betriebs ändert. Die Stabilität sowie das verbesserte Regelverhalten, insbesondere im Fall von limitierten Daten und/oder limitiertem Prozesswissen, werden in Simulationen validiert. Bemerkenswerterweise sind die Stabilitätsgarantien nicht auf den rGP-MPC Ansatz beschränkt, sondern gelten für eine breite Klasse von modellprädiktiven Regelungen die Gaußprozesse oder andere Verfahren des maschinellen Lernens für das Prädiktionsmodell verwenden.

Der zweite Teil dieser Arbeit untersucht die Verwendung von Gaußprozessen zum Lernen externer Referenz- bzw. Störsignale, insbesondere im Kontext als Teil einer Zwei-Freiheitsgrade-Regelung für die sogenannte *Rasterquantenpunktmikroskopie*. Diese kürzlich entwickelte Mikroskopietechnik erlaubt es Bilder der elektrostatischen Potentiale von Nanostrukturen, wie zum Beispiel von Atomen oder Molekülen, zu erzeugen. Die vorgestellte Zwei-Freiheitsgrade-Regelung ist das erste Regelverfahren für die Rasterquantenpunktmikroskopie. Sie besteht aus zwei unterschiedlichen Reglern die spezifische elektrische Signale während des Scanprozesses verfolgen. Diese Signale werden nach einer Aufnahme zur Erzeugung des Bildes verwendet. Es werden zwei Vorsteuerungen vorgestellt, eine basierend auf Gaußprozessen, die eine robustere Signalverfolgung erlauben und damit höhere Scangeschwindigkeiten sowie das Scannen größerer Objektproben ermöglichen. Die hergeleiteten Regler werden sowohl simulativ als auch experimentell validiert.

List of Publications

Peer-reviewed Articles in International Journals

[J1] C. Wagner, M. F. Green, M. Maiworm, P. Leinen, T. Esat, N. Ferri, N. Friedrich, R. Findeisen, A. Tkatchenko, R. Temirov, and F. S. Tautz. Quantitative imaging of electric surface potentials with single-atom sensitivity. *Nature Materials*, 18(8):853–859, 2019.

[J2] M. Maiworm, D. Limón, and R. Findeisen. Online learning-based model predictive control with Gaussian process models and stability guarantees. *International Journal of Robust and Nonlinear Control*, pages 1–28. 2021.

Peer-reviewed Articles in Conference Proceedings

[P1] M. Maiworm, T. Bäthge, and R. Findeisen. Scenario-based model predictive control: Recursive feasibility and stability. In *International Symposium on Advanced Control of Chemical Processes (ADCHEM)*, pages 50–56. IFAC, 2015.

[P2] R. Findeisen, M. A. Grover, C. Wagner, M. Maiworm, R. Temirov, F. S. Tautz, M. V. Salapaka, S. Salapaka, R. D. Braatz, and S. O. R. Moheimani. Control on a molecular scale: A perspective. In *American Control Conference (ACC)*, pages 3069–3082. IEEE, 2016.

[P3] M. Maiworm, C. Wagner, R. Temirov, F. S. 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.

[P4] M. Maiworm, D. Limón, J. M. Manzano, and R. Findeisen. Stability of Gaussian process learning based output feedback model predictive control. In *Conference on Nonlinear Model Predictive Control (NMPC)*, pages 551–557. IFAC, 2018.

[P5] M. Pfefferkorn, M. Maiworm, C. Wagner, F. S. Tautz, and R. Findeisen. Fusing online Gaussian process-based learning and control for scanning quantum dot microscopy. In *59th Conference on Decision and Control (CDC)*, pages 5525–5531. IEEE, 2020.

To be Submitted

[S1] M. Maiworm, C. Wagner, T. Esat, P. Leinen, R. Temirov, F. S. Tautz, and R. Findeisen. Control of Scanning Quantum Dot Microscopy. *Journal paper*

[S2] J. Matschek, M. Maiworm, J. Bethge, A. Savchenko, and R. Findeisen. Learning Based Model Predictive Control: Overview and Perspective. *Journal paper*

List of Symbols and Abbreviations

The following list contains abbreviations and symbols that are used in this thesis. Local symbols and local variations of the listed symbols are not included. Vectors, matrices, and sequences (of vectors or scalars) are set using bold variables. For matrices we use slanted upper case letters (e.g. $\boldsymbol{Y}$), for vectors slanted lower case (e.g. $\boldsymbol{y}$), and for sequences upright lower case (e.g. $\mathbf{y}$). Sets are denoted by calligraphic upper case letters (e.g. $\mathcal{Y}$). We deviate from this notation in the case of established definitions, such as for instance, $\mathbb{N}$ and $\mathbb{R}$ for the sets of natural and real numbers, or the class of $\mathcal{K}$-functions (see also Appendix A.1).

Abbreviations

2DOF two-degree-of-freedom

AFM atomic force microscope

GP Gaussian process

ISS input-to-state stable/stability

MPC model predictive control

NARX nonlinear autoregressive model with exogenous input

OCP optimal control problem

rGP recursive Gaussian process

SQDM scanning quantum dot microscopy

Functions

$|\cdot|$ absolute value

$\mathcal{O}(\cdot)$ big O for computational complexity

$\varrho(\cdot,\cdot)$ covariance function/kernel

$\operatorname{cov}[\cdot,\cdot]$ covariance

$\det(\cdot)$ determinant of a matrix

$\operatorname{diag}(\cdot)$ diagonal matrix

$\ell(\cdot)$ MPC stage cost

$\mathrm{E}[\cdot]$ expected value

$\kappa_\mathrm{f}(\cdot)$ MPC terminal control law

$\kappa_\mathrm{MPC}(\cdot)$ MPC control law

∇ gradient operator

$\mathcal{N}(\cdot,\cdot)$ normal/Gaussian distribution

$\|\cdot\|$ Euclidean norm

$m_+(\cdot)$ GP posterior mean function

$\sigma_+^2(\cdot)$ GP posterior variance function

$\Pr[\cdot]$ probability

$V_\mathrm{f}(\cdot)$ MPC terminal cost function

$V_N(\cdot)$ MPC cost function

$F(\cdot)$ NARX state space function

$f(\cdot)$ general function, NARX output function

$g(\cdot)$ Gaussian process

$m(\cdot)$ GP prior mean function

$p(\cdot)$ probability density function

Indices, sub-, and superscripts

$(\cdot)_k$ discrete time index

$\hat{(\cdot)}$ predicted/estimated value of a quantity

Matrices

$\boldsymbol{I}$ identity matrix

$\boldsymbol{K}$, $\boldsymbol{\Sigma}$ covariance matrix

$\boldsymbol{W}$ input/regressor training data

Scalars

ϵ noise signal

Φ electrostatic potential

σ^2 variance

k discrete time

N prediction horizon length

n number of training data points

$V^{\mp}$ negative/positive dip position

z GP output

a_d dither signal amplitude

Δf frequency change

e^{p} prediction error

$\bar{n}$ maximum number of training data points

T_{scan} scan time

V_{b} bias voltage

ω_d dither signal frequency

ω_{H} high-pass filter cutoff frequency

ω_{L} low-pass filter cutoff frequency

ω_{PLL} PLL cutoff frequency

Sets

Ω robust invariant set

Υ region of attraction

$\mathcal{D}$ training data set

$\mathbb{N}$ set of natural numbers

$\mathbb{R}$ set of real numbers

$\mathcal{U}$ input constraints set

$\mathcal{X}$ state constraints set

$\mathcal{X}_{\mathrm{f}}$ terminal constraints set

$\mathcal{X}_N$ feasible set

$\mathcal{Y}$ output constraints set

Sequences

$\mathbf{u}$ input sequence

$\mathbf{x}$ state sequence

Vectors

$\boldsymbol{e}$ error

$\boldsymbol{p}$ SQDM tip position

$\boldsymbol{\theta}$ GP hyperparameters

$\boldsymbol{u}$ system input

$\boldsymbol{w}$ regressor

$\boldsymbol{x}$ system state

$\boldsymbol{y}$ system output

$\boldsymbol{z}$ output training data

1 Introduction and Motivation

The world of today becomes ever more complex and that at an unprecedented speed. Every day new technologies emerge that require and increase the level of systems' autonomy. Popular examples are flying drones (used, e.g., in surveillance, logistics, and cinematography), self-driving cars, robots with human interaction, or self-organizing assembly streets in manufacturing. Besides, more and more of these systems are connected through local or even global communication systems.

For many autonomous systems – and their respective underlying functionalities – control and decision making are key components and thus, the design of appropriate control and decision systems is paramount. Some of the aspects that go into the design process are typical challenges in control. For example, nonlinear system behavior, safety issues expressed as constraints of the involved variables, or uncertainty in the systems themselves or their surroundings. In the case of autonomous systems, however, additional challenges arise. For instance, they have to be mostly independent and flexible, i.e., they need to be able to adapt to changing process and environmental conditions on their own. Furthermore, they might also have to cooperate with other autonomous systems or humans in their working environment. To cope with these challenges, many systems exchange and gather information from their environment at an increasing rate. This is facilitated, among other developments, by the ongoing digitalization of today's world. Data is automatically collected and stored, which can be used for proper system operation, statistical analysis, and general improvement of systems' performance. In order to leverage the information of all this data acquisition for control and decision systems, traditional methods from control are increasingly combined with methods from machine learning. From this combination emerge new interesting applications, as well as questions regarding the (provable) properties of the resulting control algorithms. This thesis is intended to a shed a bit more light on these aspects.

1.1 Machine Learning

The term *machine learning* was first introduced in [163] and can be considered as a subfield of artificial intelligence. It denotes algorithms that enable a computer agent to solve specific problems in a way that resembles human learning by improving from experience, i.e., using acquired data. To this end, machine learning puts more emphasis on the data aspect than other more traditional methods that use rigid functional instructions. The utilized data is usually denoted as *training data* and used, for

instance, to build models to make predictions or take decisions. The employed methods are closely related to other fields, such as computational statistics, probability theory, stochastic systems, and mathematical optimization to name a few. Some popular examples of machine learning algorithms are decision trees, artificial neural networks (e.g. recurrent and deep learning networks), support vector machines, or Gaussian processes. Applications are manifold and span from speech [66] and image recognition [102], over medical diagnosis [136] and email filtering [8], to machine translations (e.g. linguee and deeple [201]), which many of us use in daily life. Some machine learning algorithms even gained significant media attention when they beat for the first time human professional players in traditional board games (e.g. chess and Go [172]) and also competitive computer games [190].

Approaches of machine learning are often divided into the three categories *supervised learning*, *unsupervised learning*, and *reinforcement learning* (see Fig. 1.1). The arguably most prominent learning scheme is supervised learning, which is characterized by the objective to determine a mapping

$$ML : \mathcal{W} \to \mathcal{Z} \ , \tag{1.1}$$

given a labeled training data set $\mathcal{D} = \{(\boldsymbol{w}_i, \boldsymbol{z}_i)\}_{i \in \mathbb{N}}$, where the variables $\boldsymbol{w} \in \mathcal{W}$ are the independent input variables (also denoted as attributes or features) and the variables $\boldsymbol{z} \in \mathcal{Z}$ are the dependent output variables (also denoted as targets). The mapping (1.1) is usually used to predict the targets $\boldsymbol{z}$, given some previously unseen inputs $\boldsymbol{w}$. Supervised learning is probably used most frequently in control, in particular in the form of regression algorithms because in theory any functional dependency can be learned or approximated. A prominent example in model-based control is learning the (open or closed-loop) model of the controlled process [42, 43, 77, 141, 157, 176, 185]. This has two important advantages. First, the typical and time consuming modeling process (via first principles and conducted by a human expert) can be skipped and second, newly available data during operation allows to adapt and improve the model to changing process and environmental conditions. Further examples in combination

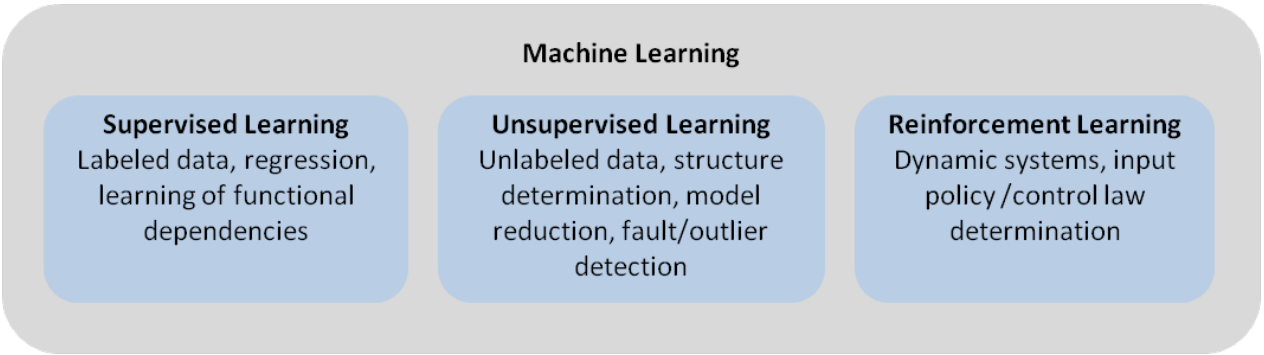


Figure 1.1: Common learning paradigms in machine learning. Note that they do not exclude each other.

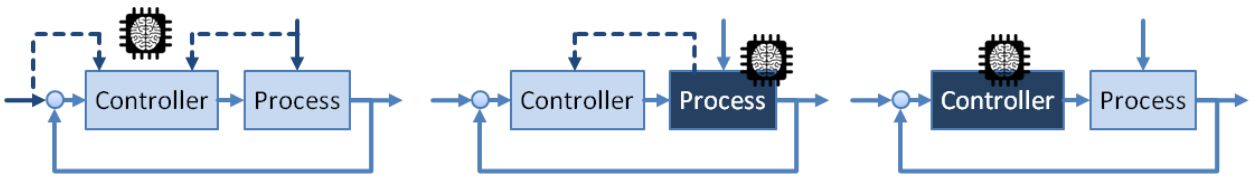


Figure 1.2: Possibilities for the inclusion of machine learning in control: Learning external reference or disturbance signals (left), learning a model of the controlled process (middle), or learning the controller or parts of it (right).

with control are learning of disturbances or unknown references [84, 92, 125, 198, 199], the cost function in optimization-based approaches [16, 19, 45, 123, 124], or learning directly the control policy or their parameters [5, 35, 83, 114, 208], see Fig. 1.2.

In contrast to supervised learning, only a set of independent variables $\{\boldsymbol{w}_i\}_{i\in\mathbb{N}}$ is available in unsupervised learning, i.e., one cannot differentiate between independent and dependent variables. There the objective can be, for example, to determine the structure of or similarities within the data itself, as is done in model reduction (differentiating between dominant and nondominant parts of a model, e.g. see [97]) or the determination of outliers within data. Thus, unsupervised learning can be employed as a preprocessing step in supervised learning and control.

Whereas supervised and unsupervised learning are rather general concepts, reinforcement learning [179] considers specifically dynamic environments or processes. Therein it seeks to learn suitable input actions/decisions to influence the state of a system, such that a numerical reward or a respective cumulative reward over time, also called value function[1], is maximized. This does not only allow to compute specific input values but also to find suitable control policies (or feedback laws). The reward or value function can be modeled in different ways, including linear function approximations, neural networks, or Gaussian processes [157, 179].

Besides the achievements of machine learning algorithms, they do have also some drawbacks. One of them is that they often lack interpretability. In contrast to state-of-the-art mathematical modeling that uses, for instance, first principles (also called *white box*) models (models arising, e.g., from energy or mass balances), machine learning algorithms often model or learn only an input-output behavior (also called *black box* modeling). Thus, internal working principles or insightful relations cannot be derived and interpretability from a physical point of view is low.

Another drawback is the high computational load that many machine learning algorithms exhibit, at least during the learning phase. This rendered potential algorithms impractical for many application cases in the past. Fortunately, the involved compu-

[1]The value function of reinforcement learning is closely related to the cost function in model predictive control and reinforcement learning itself is closely related to techniques of optimal control and dynamic programming [104, 180]. One of the main differences is that in reinforcement learning no model of the dynamic process is necessary or typically used.

tations become more and more feasible. This is driven, on one hand, by algorithmic developments from the scientific community but mostly by the increase of computational power and services such as cloud computing. This is also one of the main reasons, besides good performance, that machine learning is also increasingly used for and in combination with control and automation.

One particular method that benefits from these developments are Gaussian processes (GPs, [87, 158]), which are the focus of this thesis. GPs are stochastic processes that are particularly well suited for generating maps, whose output is corrupted by normally/Gaussian distributed noise. Given their stochastic nature, the tendency of overfitting is low and their expected model quality can be evaluated via confidence intervals of the predictions. Gaussian processes can be used to model or approximate a wide variety of functions that can be used to capture static maps or dynamic systems. They allow to combine data-driven learning, by using measurements as training points, and a priori knowledge via user defined mean and covariance functions. This incorporates good interpolation quality with extrapolation capabilities. Furthermore, Gaussian processes can be utilized for purely data derived models [24, 31, 32, 90, 91, 106, 135, 141, 184, 185] or they can be combined in a hybrid way with other, for instance, deterministic models [4, 17, 20, 76, 94, 132, 147, 176, 203]. For these reasons, GPs have been increasingly utilized for control in the last two decades, both in theory and practice.

1.2 Contributions

In this thesis, we consider and develop new methods for fusing and combining control and Gaussian processes. We show how Gaussian processes can contribute, (*i*) to the improvement of established control paradigms (model predictive control in particular) while maintaining stability guarantees and, (*ii*) to the development of designated controllers for new cutting-edge technologies (scanning quantum dot microscopy). The following provides a brief summary of the contributions of this thesis, divided into the respective chapters. More detailed lists of the respective contributions can be found in the introductory sections of each of the Chapters 3 and 4.

Chapter 3: Recursive Gaussian Process Model Predictive Control
Recursive Gaussian process model predictive control (rGP-MPC) is presented, a model predictive control scheme that can be used for a wide class of application cases. It employs Gaussian process prediction models of the controlled processes, which can be learned online using only input-output data. Hence, no physical system model is required. The possible application cases are further increased by the use of an output feedback formulation, which does not require the individual process states to be available. In addition, rGP-MPC can also be applied to processes with fast dynamics.

To this end, a recursive GP learning approach is adopted that includes updates of the training data set and respective updates of the inverse covariance matrix. Besides the possibility for wide application, conditions for guaranteed recursive constraint satisfaction, as well as nominal and inherent robust stability are derived. Additionally, a structured calculation of the required terminal MPC components, based on the linearized GP posterior mean function and linear matrix inequalities, is presented. These contributions were published in [121] and [120].

Chapter 4: Control of Scanning Quantum Dot Microscopy
Chapter 4 considers an important and challenging application example of machine learning in control. The first (two-degree-of-freedom) control framework for scanning quantum dot microscopy (SQDM) is presented, a novel microscopy technique that generates images of the electrostatic potentials of nanostructures. The control framework is a key enabling component that turns SQDM into a well applicable microscopy technique. It consists of two different feedback and two different feedforward controllers tailored to SQDM. In one of the feedforward controllers Gaussian processes are utilized to increase performance and correct operation by online learning and compensation of a signal that can be interpreted as a disturbance to the closed-loop. The control framework allows to scan the sample continuously, thereby generating electrostatic potential images in less time, of larger surface areas, and with higher resolution than before. This also puts SQDM in line with other microscopy techniques like scanning tunneling microscopy and atomic force microscopy. Parts of these contributions were published in [122] and used for [196].

1.3 Thesis Outline

Following this introduction, Chapter 2 deals with Gaussian processes. We first review the basics, in particular of Gaussian process regression in supervised learning, and after that the literature concerning the application of Gaussian process regression in control and focus on the categories of reference and disturbance learning, as well as learning of dynamic models. These two categories are of particular importance for this thesis.

In Chapter 3, we present the rGP-MPC scheme that employs Gaussian process regression to learn prediction models of the controlled process and update them online during operation. After an introduction to model predictive control and a literature review of model predictive control in combination with Gaussian processes, we state the considered problem definition, move on to the online learning scheme, and formulate the rGP-MPC optimal control problem. We prove, as one of the first groups, that the scheme is inherently robustly stable, despite the changing Gaussian process prediction model, which is also validated in simulations. The scheme is notably not

limited to Gaussian processes. The provided guarantees hold also for other machine leaning algorithms.

In Chapter 4, we present the control framework for scanning quantum dot microscopy. We start with an explanation of the working principle of this novel microscopy technique, followed by the developed two-degree-of-freedom control framework of this thesis that encompasses two designated feedback controllers and two feedforward signal generators, of which one is based on learning with Gaussian processes. We validate the control framework and extensively investigate its properties and performance in simulations. At last we discuss implementation details for the experiment and present experimentally acquired images.

The thesis is concluded in Chapter 5.

2 An Introduction and Review of Gaussian Processes

This chapter presents Gaussian processes (GPs, [87, 158]), which play a key role in Chapters 3 and 4 of this thesis. They can be used for classification and regression problems, though the latter is the one that is more important in the context of control. For this reason, Gaussian process regression is presented in detail in Section 2.1. In Section 2.2 we conceptualize Gaussian process regression in the realm of control systems, in particular for reference/disturbance learning and learning of dynamic models. We also provide a literature overview that serves as a basis for the contributions following in Chapters 3 and 4. A short summary is presented in Section 2.3.

2.1 Gaussian Process Regression

In this section we discuss Gaussian process regression. To this end we start with the standard Gaussian (normal) distribution of a one dimensional variable (Section 2.1.1), move on to the Gaussian distribution in higher dimensions and finally introduce Gaussian processes as a generalization to the space of functions. Then, in Section 2.1.2 we present them in the context of supervised learning and present the core equations that will be used also in Chapters 3 and 4.

2.1.1 Basics

Many processes are subject to stochastic disturbances or other influencing factors that render the process itself stochastic in nature. To describe the stochastic properties, a probability density function can be assigned. Of the many existing density functions, the Gaussian (normal) distribution, which we consider in the following, is the most prominent one[1].

The probability density function of a 1-dimensional Gaussian (normal) distribution (Fig. 2.1) is

$$\mathcal{N}(\mu, \sigma^2): \quad f(g) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(g-\mu)^2}{\sigma^2}\right] \tag{2.1}$$

with the mean value μ and the variance σ^2 (or standard deviation σ respectively). If a scalar random variable $g \in \mathbb{R}$ is normally distributed according to (2.1), then we have $\mu = \mathrm{E}[g]$ and $\sigma^2 = \mathrm{E}[(g-\mu)^2]$, where $\mathrm{E}[\cdot]$ is the expected value. To indicate that g is

[1] Note that if the true distribution is not Gaussian (e.g. a Gamma or Poisson distribution), then assuming a Gaussian distribution is only an approximation.

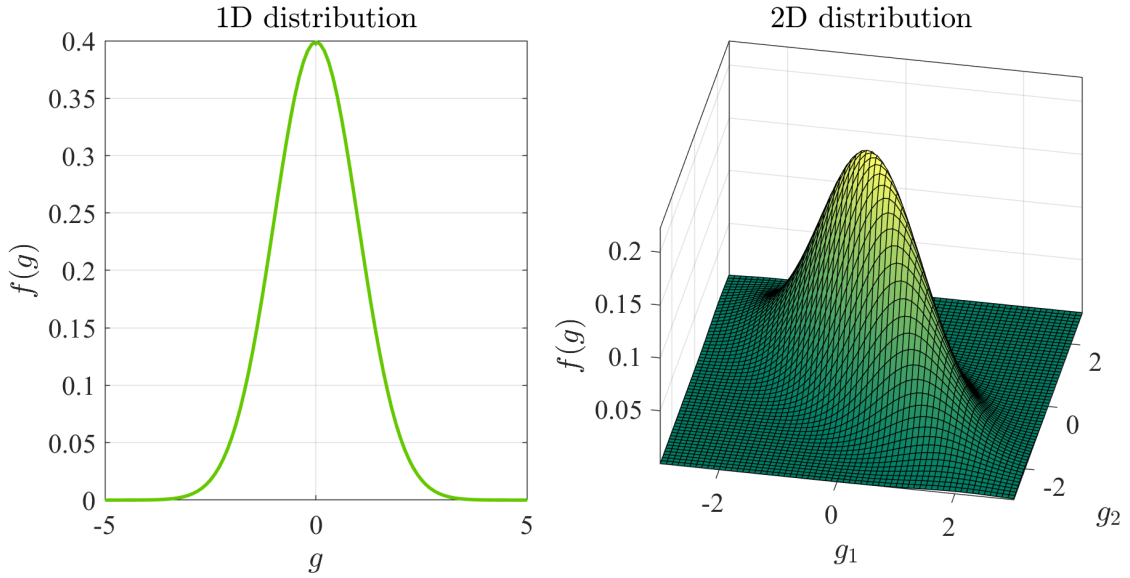


Figure 2.1: Probability density functions of a 1D Gaussian distribution (left) and a 2D multivariate Gaussian distribution (right). The mean is in both cases zero, whereas the variance is $\sigma^2 = 1$ in the 1D case and $\boldsymbol{\Sigma} = \left[\begin{smallmatrix} 1 & -0.7 \\ -0.7 & 1 \end{smallmatrix}\right]$ in the 2D case. The mutual dependence of g_1 and g_2 in the right figure is due to the nonzero off-diagonal elements -0.7 in the covariance matrix $\boldsymbol{\Sigma}$.

normally distributed, we write $g \sim \mathcal{N}(\mu, \sigma^2)$ or $p(g) = \mathcal{N}(\mu, \sigma^2)$, where $p(\cdot)$ denotes the probability density function. Note that the expected value μ and variance σ^2 are constants and that the distribution of the random variable g depends only on these parameters.

Many applications require multiple variables. The n-dimensional multivariate Gaussian distribution is given by

$$\mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}): \quad f(\boldsymbol{g}) = \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma})}} \exp\left[-\frac{1}{2}(\boldsymbol{g}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{g}-\boldsymbol{\mu})\right] \tag{2.2}$$

with mean vector $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$, which is symmetric and positive definite. If a n-dimensional random vector $\boldsymbol{g} = [g_1 \ \cdots \ g_n]^\mathsf{T}$ has a multivariate Gaussian distribution according to (2.2) we write

$$\boldsymbol{g} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

with $\boldsymbol{\mu} = \left[\mathrm{E}[g_1] \ \cdots \ \mathrm{E}[g_n]\right]^\mathsf{T}$ and $\Sigma_{i,j} = \mathrm{E}[(g_i - \mu_i)(g_j - \mu_j)]$, where $i, j \in \{1, \ldots, n\}$. As in the one dimensional case, the multivariate distribution of $\boldsymbol{g}$ depends only on the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Note that for $i = j$ we have $\Sigma_{i,j} = \sigma_i^2$, i.e., the variances of each element of $\boldsymbol{g}$ are located on the diagonal of $\boldsymbol{\Sigma}$. On the other hand, all off diagonal entries $\Sigma_{i,j}$ with $i \neq j$ are covariances and quantify the mutual dependencies between g_i and g_j (see Fig. 2.1). Accordingly, if all off diagonal covariance entries in $\boldsymbol{\Sigma}$ are zero, then all random variables g_i are independently normally

distributed. In that case we do not have a n-dimensional multivariate distribution but n 1-dimensional Gaussian distributions.

Given such distributions, a Gaussian process $g(\boldsymbol{w}) : \mathbb{R}^{n_w} \to \mathbb{R}$ is a stochastic process and is a generalization of the Gaussian probability distribution to functions. It is usually written in the form

$$g(\boldsymbol{w}) \sim p(g(\boldsymbol{w})) = \mathcal{GP}\big(m(\boldsymbol{w}), \varrho(\boldsymbol{w}, \boldsymbol{w}')\big) \ ,$$

where $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^{n_w}$ is a deterministic input, called vector of *regressors* (or just regressor for short), $m(\boldsymbol{w}) = \mathrm{E}[g(\boldsymbol{w})]$ is the GP mean function, and

$$\varrho(\boldsymbol{w}, \boldsymbol{w}') = \mathrm{cov}[g(\boldsymbol{w}), g(\boldsymbol{w}')] = \mathrm{E}\left[\big(g(\boldsymbol{w}) - m(\boldsymbol{w})\big)\big(g(\boldsymbol{w}') - m(\boldsymbol{w}')\big)\right]$$

denotes the GP covariance (or *kernel*) function. Note that the covariance function $\varrho(\boldsymbol{w}, \boldsymbol{w}')$ is written as a function of the inputs $\boldsymbol{w}, \boldsymbol{w}'$ but yields the covariance between the outputs $g(\boldsymbol{w}), g(\boldsymbol{w}')$.

The GP mean and covariance are now functions and completely define a Gaussian process. For a given input $\boldsymbol{w}$, the mean function $m(\boldsymbol{w})$ denotes the most likely value of the output of $g(\boldsymbol{w})$ and the covariance function yields the corresponding variance $\varrho(\boldsymbol{w}, \boldsymbol{w}) = \sigma_g^2$. For different $\boldsymbol{w}$ and $\boldsymbol{w}'$, $\varrho(\boldsymbol{w}, \boldsymbol{w}')$ yields the respective covariances. Hence, in contrast to the Gaussian distributions discussed above, a GP $g(\cdot)$ is a stochastic process, whose probabilistic parameters, mean and (co-)variance, are not any longer constants but *functions* that depend on another deterministic input parameter, or set of input parameters $\boldsymbol{w}$. Accordingly, different inputs $\boldsymbol{w}$ can lead to different random variables $g(\boldsymbol{w})$, which are all normally distributed but with possibly different mean and variance.

Furthermore, n different inputs $\boldsymbol{w}$ to a GP lead to a n-dimensional multivariate Gaussian (2.2). For this reason, in [158] a GP is also defined as *a collection of random variables, any finite number of which have a joint Gaussian distribution.*[2] In other words, a GP is a collection of infinitely many normally distributed random variables for which each finite n-dimensional subset of them is a n-variate Gaussian distribution. In particular, for each finite number of sampling points $\boldsymbol{W} = [\boldsymbol{w}_1 \ \cdots \ \boldsymbol{w}_n]^\mathsf{T} \in \mathbb{R}^{n \times n_w}$, the GP provides a n-variate normal distribution of $g(\boldsymbol{w}_i)$

$$\boldsymbol{g}(\boldsymbol{W}) = [g(\boldsymbol{w}_1) \ \cdots \ g(\boldsymbol{w}_n)]^\mathsf{T} \sim \mathcal{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ ,$$

with mean vector $\boldsymbol{\mu} = [m(\boldsymbol{w}_1) \ \cdots \ m(\boldsymbol{w}_n)]^\mathsf{T}$ and covariance matrix $\boldsymbol{\Sigma} = \varrho(\boldsymbol{W}, \boldsymbol{W}) = [\varrho(\boldsymbol{w}_i, \boldsymbol{w}_j)] \in \mathbb{R}^{n \times n}$. A GP can also be seen as a distribution over possible functions in a continuous domain that maps from deterministic input signals $\boldsymbol{w}$ to stochastic output signals $g(\boldsymbol{w})$. Example functions are illustrated in Fig. 2.2.

[2] Consider, for instance, a time series generated by a Gaussian process (e.g. Fig. 2.2 with $\boldsymbol{w} = t$.), then each point in time is another dimension in the multivariate distribution and the covariance matrix describes the relations between the points.

2.1.2 GP Supervised Learning

In the realm of machine learning algorithms, Gaussian processes are often used for (see [97] for an exception) supervised learning (Chapter 1), i.e., labeled input-output data $\{(\boldsymbol{w}_i, z_i)\}_{i\in\mathbb{N}}$ is used to determine a mapping of the form (1.1), which is particularly relevant for control.

The general goal is to approximate and generate predictions of maps of the form

$$z = f(\boldsymbol{\nu}) + \epsilon \tag{2.3}$$

with deterministic inputs $\boldsymbol{\nu} \in \mathbb{R}^{n_\nu}$ and output $z \in \mathbb{R}$ that is assumed to be corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_\mathrm{n}^2),\ \epsilon \in \mathbb{R}$ with zero mean and noise variance σ_n^2. The approximation of (2.3) consists in inferring the underlying but unknown *latent* function $f(\boldsymbol{\nu})$ using a GP $g(\boldsymbol{w})$ and measured input-output data points $(\boldsymbol{\nu}, z)$.

Remark 1. *We have used different variables $\boldsymbol{\nu}$ and $\boldsymbol{w}$ to denote the inputs to the latent function and the Gaussian process because the two do not necessarily have to be equal. For instance, the input regressor vector $\boldsymbol{w}$ that is used for modeling might contain less elements than $\boldsymbol{\nu}$ if, e.g., not all elements in $\boldsymbol{\nu}$ can be measured. In the rest of this chapter, however, we consider $\boldsymbol{w} = \boldsymbol{\nu}$ for simplicity of presentation.*

2.1.2.1 Posterior Distribution

In order to approximate (2.3), we require a suitable prior mean function $m(\boldsymbol{w})$ and covariance function $\varrho(\boldsymbol{w}, \boldsymbol{w}')$, where the only constraint on $\varrho(\boldsymbol{w}, \boldsymbol{w}')$ is that the resulting covariance matrix is symmetric and positive definite. The GP prior $g(\boldsymbol{w})$ has then to be trained using a set of n measured input-output data points $(\boldsymbol{w}, z)$. This training data set is $\mathcal{D} = \{\boldsymbol{W}, \boldsymbol{z}\}$ with input data $\boldsymbol{W} = [\boldsymbol{w}_1\ \cdots\ \boldsymbol{w}_n]^\mathsf{T} \in \mathbb{R}^{n\times n_w}$ and output data $\boldsymbol{z} = [z_1\ \cdots\ z_n]^\mathsf{T} \in \mathbb{R}^n$. Furthermore, we now consider $\boldsymbol{w}$ to be a new (unseen) test input for which we are interested in the predicted output. The joint distribution of $\boldsymbol{g}(\boldsymbol{W})$ and $g(\boldsymbol{w})$ is

$$\begin{bmatrix}\boldsymbol{g}(\boldsymbol{W})\\ g(\boldsymbol{w})\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}m(\boldsymbol{W})\\ m(\boldsymbol{w})\end{bmatrix}, \begin{bmatrix}\boldsymbol{K} & \varrho(\boldsymbol{W}, \boldsymbol{w})\\ \varrho(\boldsymbol{w}, \boldsymbol{W}) & \varrho(\boldsymbol{w}, \boldsymbol{w})\end{bmatrix}\right) \tag{2.4}$$

with $m(\boldsymbol{W}) = [m(\boldsymbol{w}_1)\ \cdots\ m(\boldsymbol{w}_n)]^\mathsf{T} \in \mathbb{R}^{n\times 1}$, $\varrho(\boldsymbol{w}, \boldsymbol{W}) = [\varrho(\boldsymbol{w}, \boldsymbol{w}_1)\ \cdots\ \varrho(\boldsymbol{w}, \boldsymbol{w}_n)] \in \mathbb{R}^{1\times n}$, $\varrho(\boldsymbol{W}, \boldsymbol{w}) = \varrho(\boldsymbol{w}, \boldsymbol{W})^\mathsf{T}$, and the covariance matrix $\boldsymbol{K}$ that also accounts for the measurement noise σ_n^2 via $\boldsymbol{K} = \varrho(\boldsymbol{W}, \boldsymbol{W}) + \sigma_\mathrm{n}^2\boldsymbol{I} = [\varrho(\boldsymbol{w}_i, \boldsymbol{w}_j) + \sigma_\mathrm{n}^2\delta_{ij}] \in \mathbb{R}^{n\times n}$, where $\boldsymbol{I}$ is the identity matrix and δ_{ij} the Kronecker delta.

To obtain an explicit prediction for a test point $\boldsymbol{w}$, the joint Gaussian prior distribution (2.4) has to be restricted to contain only those functions that go through or pass (if $\sigma_\mathrm{n}^2 \neq 0$) the points in the training data set $\mathcal{D}$. This can be achieved by conditioning the joint prior (2.4) on the training data [50, 191], leading to a GP posterior

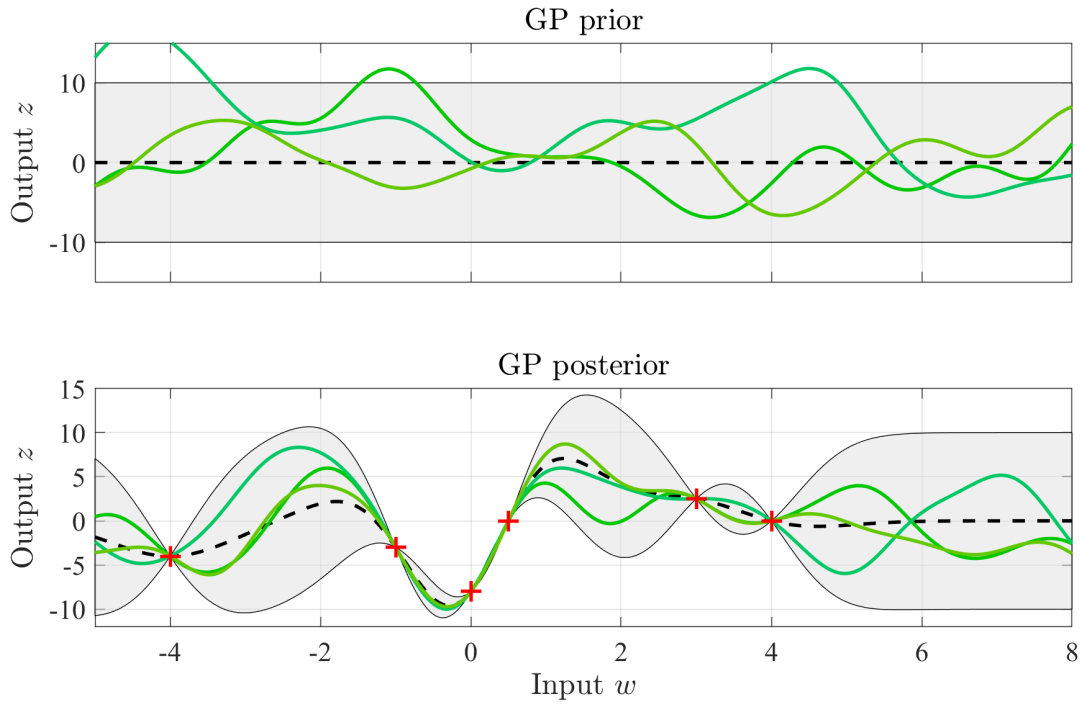


Figure 2.2: Gaussian process inference: The top figure depicts a GP prior distribution with the dashed black line representing the mean function $m(\boldsymbol{w})$ and the green lines representing random function realizations drawn from the prior distribution. The grey shaded area is the 95 % (twice the standard deviation) confidence interval computed via $\varrho(\boldsymbol{w}, \boldsymbol{w}')$. When data points $\mathcal{D}$ are added (bottom figure, red crosses), the GP posterior with $m_+(\boldsymbol{w}|\mathcal{D})$ and $\sigma_+^2(\boldsymbol{w}|\mathcal{D})$ is inferred from this data. When the inputs w leave the training data region (for $w > 5$) the posterior tends towards the prior.

distribution

$$g(\boldsymbol{w}|\mathcal{D}) \sim \mathcal{N}\big(m_+(\boldsymbol{w}|\mathcal{D}), \sigma_+^2(\boldsymbol{w}|\mathcal{D})\big) \tag{2.5}$$

with posterior mean function $m_+(\boldsymbol{w}|\mathcal{D})$ and posterior variance $\sigma_+^2(\boldsymbol{w}|\mathcal{D})$ given by

$$m_+(\boldsymbol{w}|\mathcal{D}) = m(\boldsymbol{w}) + \varrho(\boldsymbol{w}, \boldsymbol{W})\boldsymbol{K}^{-1}(\boldsymbol{z} - m(\boldsymbol{W})) \tag{2.6a}$$

$$\sigma_+^2(\boldsymbol{w}|\mathcal{D}) = \varrho(\boldsymbol{w}, \boldsymbol{w}) - \varrho(\boldsymbol{w}, \boldsymbol{W})\boldsymbol{K}^{-1}\varrho(\boldsymbol{W}, \boldsymbol{w}) \ . \tag{2.6b}$$

This conditioning of the prior on the training data is also called *inference*. The inference step from prior to posterior distribution is illustrated in Fig. 2.2.

The posterior mean function $m_+(\boldsymbol{w}|\mathcal{D})$ is the sought estimator of the latent function $f(\cdot)$. It yields an estimate $\hat{z} = m_+(\boldsymbol{w}|\mathcal{D})$ for the target z at the test point $\boldsymbol{w}$ and given training data $\mathcal{D}$. The posterior variance $\sigma_+^2(\boldsymbol{w}|\mathcal{D})$ yields the corresponding variance at the test point, which allows quantifying the model quality by calculating confidence intervals.

Remark 2 (Posterior notation)**.** *Note that the notation for the GP posterior distribution is not unique in the literature. Compare, for instance, the notations in [158] and*

[87]. We have adopted the notation of [158], where (2.5) *is denoted as the posterior GP but is written with* $\mathcal{N}$ *and not* $\mathcal{GP}$*. The main reason is probably (nothing is said in this regard) that by conditioning the GP prior on the training data, we automatically obtain a n-variate normal distribution. Furthermore, for one specific test input* $\boldsymbol{w}$ *the resulting output* g *is then normally distributed with*

$$p\Big(g(\boldsymbol{w}|\mathcal{D})\Big) = \mathcal{N}\Big(m_+(\boldsymbol{w}|\mathcal{D}), \sigma_+^2(\boldsymbol{w}|\mathcal{D})\Big) : \\ f(g) = \frac{1}{\sqrt{2\pi\sigma_+^2(\boldsymbol{w}|\mathcal{D})}} \exp\left[-\frac{1}{2}\frac{(g - m_+(\boldsymbol{w}|\mathcal{D}))^2}{\sigma_+^2(\boldsymbol{w}|\mathcal{D})}\right] .$$

Remark 3 (Infinitely many realizations)**.** *As the posterior is conditioned on the training data points* $\mathcal{D}$*, it rejects all possible function realizations that do not go through or nearby (if* $\sigma_\mathrm{n}^2 \neq 0$*) these points. However, the number of possible function realizations is still infinite.*

2.1.2.2 Mean and Covariance Function

The key elements for a Gaussian process to yield a sensible model, describing the mapping (2.3), are the prior mean and covariance function. Both depend on a set of parameters $\boldsymbol{\theta}$, i.e., $m(\boldsymbol{w}|\boldsymbol{\theta})$ and $\varrho(\boldsymbol{w}, \boldsymbol{w}'|\boldsymbol{\theta})$, which are called *hyperparameters* to delimit them from parameters used in parametric system models. Very often just a constant zero prior mean $m(\boldsymbol{w}|\boldsymbol{\theta}) = c = 0$ is used [68, 91, 122]. However, other choices include, for instance, the use of a deterministic base model as the prior mean function [161, 203].

Regarding the covariance function, it is often assumed or known that the target function (2.3) can be modeled by a member of the space of smooth functions C^∞. A covariance function that provides this property is the *squared exponential covariance function*[3] *with automatic relevance determination*

$$\varrho(\boldsymbol{w}, \boldsymbol{w}'|\boldsymbol{\theta}) = \sigma_\mathrm{f}^2 \exp\left(-\frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}')^\mathsf{T}\boldsymbol{\Lambda}(\boldsymbol{w} - \boldsymbol{w}')\right) , \tag{2.7}$$

with $\boldsymbol{\Lambda} = \mathrm{diag}(l_1^{-2}, \ldots, l_{n_w}^{-2})$. The resulting hyperparameters are $\boldsymbol{\theta} = \{\sigma_\mathrm{f}^2, \boldsymbol{\Lambda}\}$, where σ_f^2 is called the signal variance and represents a vertical scaling factor of the resulting values from (2.7). The parameters l_j are called length scale parameters. The larger l_j, the less vary the resulting function realizations with respect to the associated regressor w_j and hence, the smaller the influence of w_j and the less important it is. The reciprocal $1/l_j^2$ can be interpreted as a weighting factor for the regressor w_j. This allows to determine the minimum required number of regressors n_w in $\boldsymbol{w}$ by excluding those with particularly large length scales [88, 202].

[3] Note that the squared exponential covariance function is sometimes also denoted as *Gaussian radial basis function*; especially in the field of neural networks or support vector machines.

Most of the publications that involve Gaussian processes use (2.7) in one way or another. However, other choices include, for instance, the combination of (2.7) with a linear kernel [2, 184, 202], a periodic kernel [84, 125, 144], or the product of a linear and a Matérn kernel [17, 94]. Note that one can compose infinitely many covariance functions from basic ones using addition and multiplication operations [48, 49, 158].

2.1.2.3 Hyperparameter Optimization

Since both the mean function $m(\boldsymbol{w}|\boldsymbol{\theta})$ and the covariance function $\varrho(\boldsymbol{w}, \boldsymbol{w}'|\boldsymbol{\theta})$ depend on the hyperparameters $\boldsymbol{\theta}$, a good set of hyperparameters is required to match the behavior of the latent function. The hyperparameters can, in principle, be chosen by prior knowledge and physical insight, though this can be challenging. Instead, they are usually determined using the available training data $\mathcal{D} = \{\boldsymbol{W}, \boldsymbol{z}\}$. Given a prior probabilistic belief of the hyperparameters' distribution $p(\boldsymbol{\theta})$ (often a uniform distribution is assumed, i.e., all values of the hyperparameters are equally probable), the goal is to infer the hyperparameters' posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{W}, \boldsymbol{z})$. Theoretically, this can be done by application of Bayes' theorem

$$p(\boldsymbol{\theta}|\boldsymbol{W}, \boldsymbol{z}) = \frac{p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{z}|\boldsymbol{W})} \ .$$

Here $p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\theta})$ is the likelihood and $p(\boldsymbol{z}|\boldsymbol{W})$ the marginal likelihood (or evidence). Unfortunately, the involved integrals are intractable in most cases and numerical approximations have to be employed. One such approximation, for example, consists in using the Markov chain Monte Carlo method, which is, however, computationally expensive. Another approach consists in maximizing the likelihood $p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\theta})$ (see, e.g., [87] for a good explanation in this regard), which works well when the likelihood is very peaked, as is often the case for a large training data set [161]. In particular, the approach that has become standard is to compute a point estimate of the most likely hyperparameters by maximizing the log likelihood

$$\log(p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\theta})) = -\frac{1}{2}\boldsymbol{z}^{\mathsf{T}}\boldsymbol{K}^{-1}\boldsymbol{z} - \frac{1}{2}\log(\det(\boldsymbol{K})) - \frac{n}{2}\log(2\pi) \tag{2.8}$$

with respect to $\boldsymbol{\theta}$.

Remark 4 (GP learning)**.** *In the context of Gaussian processes, the term "learning" is often used ambiguously. It is used for both the inference step from prior to posterior and for the optimization of the hyperparameters. However, these two steps have to be performed always. Therefore, GPs are often also distinguished whether the learning is performed only once offline or online during operation. Then, learning can refer to (i) inference and hyperparameter optimization performed offline [91, 106, 146], (ii) inference and hyperparameter optimization performed online [84, 135, 144], or (iii) a hybrid approach that combines offline hyperparameter optimization and online*

inference updates, based on newly available training data [77, 146, 147]. The latter case is also considered in this work.

2.1.2.4 Drawback

Besides the many advantages that Gaussian processes offer, their main drawback is the computational complexity with respect to the number of training data points. The computations of the covariance matrix inversion $\boldsymbol{K}^{-1}$ in (2.6a), (2.6b), and (2.8) scale with $\mathcal{O}(n^3)$, where n is the number of training data points. This severely limits the application of GP models for fast processes, where small sampling times are required; especially in the case of relatively large training data sets with several hundred or thousands of data points. If online or close to online hyperparameter optimization is needed, this drawback becomes even more pronounced. In Chapter 3, we tackle the issue in the context of model predictive control and show how to reduce the computational costs.

2.2 Gaussian Process Learning in Control

Supervised Gaussian process regression can be utilized to learn a wide variety of different mappings, as long as the provided training data contains enough information. Whatever is learned with a Gaussian process, the posterior mean $m_+(\boldsymbol{w}|\mathcal{D})$ with regressor vector $\boldsymbol{w}$ and training data set $\mathcal{D}$ yields the approximation or estimate of the mapping one is interested in. In this section we focus on the application of GP regression in the context of control. Although many different parts and aspects of a closed-loop can be learned with Gaussian processes, we focus on the two categories that are of particular importance to this thesis. These are reference and disturbance learning, as well as learning of models that represent dynamic systems (see Fig. 2.3).

Learning of dynamic models plays a central role in Chapter 3, where we combine it with model predictive control and show under which circumstances robust stability can be guaranteed despite an online changing GP model. On the other hand, GP disturbance learning is a central element in Chapter 4, where a prediction of the disturbance is required to improve the closed-loop performance of a controller for the new microscopy technique scanning quantum dot microscopy.

In the following we review different modeling approaches for reference and disturbance learning (Section 2.2.1) and dynamic models learning (Section 2.2.2) and show how these approaches are reflected in the GP regressor vector $\boldsymbol{w}$ and the target z.

2.2.1 Reference and Disturbance Learning

Reference signals to a closed-loop system (see Fig. 2.3) depend usually only on time and are predetermined by the respective application or are chosen by an operator

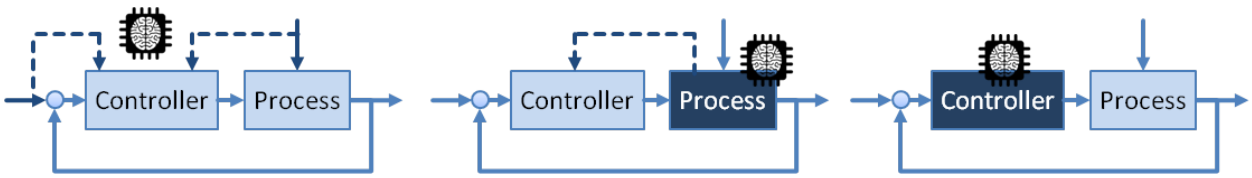


Figure 2.3: Possibilities for the inclusion of machine learning in control: Learning external reference or disturbance signals (left), learning a model of the controlled process (middle), or learning the controller or parts of it (right).

beforehand. External disturbances on the other hand are often signals from other dynamic systems or environments that influence the closed-loop system of interest. Despite this difference, reference and disturbance signals can often be transformed into one another and can then be considered conceptually equivalent. For this reason we treat them together in the following.

Notice that most of the examples in the literature are denoted as disturbance modeling. If a model of such a disturbance signal is desired to enhance closed-loop control, one has mainly two options; either model the disturbance only as a time dependent signal (to some extent, this works well for periodic or quasi periodic signals) or try to model the underlying disturbance dynamics. Another possibility for the case of disturbance signals is considered in this section, which will also be investigated in more detail in Chapter 4.

NAR Models One approach to model and predict the evolution of reference/disturbance signals y_k is via *nonlinear autoregressive* (NAR) models [111]

$$y_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k-m_y}) \ .$$

Here the input argument of $f(\cdot)$ comprises y_k and its m_y previous values, which also determine the NAR model order. The GP target is then $z = y_{k+1}$ and the regressor vector $\boldsymbol{w} = (y_k, y_{k-1}, \ldots, y_{k-m_y})$.

For instance, in [198, 199] a model predictive controller for a drinking water network was developed. The unknown water demand was considered as a disturbance and modeled via the combination of a deterministic model and a GP-NAR model. Further examples can also be found in the realm of time series modeling and prediction. For instance, GPs have been employed for faulty measurement detection and reconstruction in urban traffic [92], data-driven forecasting of building energy consumption [143], or prediction of respiratory signals [27] and mine gas emissions [46].

Time Dependent Signals For the case of periodic or quasi periodic reference/disturbance signals, another approach is to model the signal to be only dependent on time, i.e.,

$$y_k = f(k)$$

with $z = y_k$ and $\boldsymbol{w} = k$ for the GP. In that case, a covariance function with periodic term has to be employed. For instance, in [84, 125, 144] the covariance function

$$\varrho(k, k') = \sigma_\mathrm{f}^2 \cdot \exp\left(-\frac{(k-k')^2}{2l_1^2}\right) \cdot \exp\left(-\frac{2\sin^2(\frac{\pi}{\lambda}(k-k'))}{l_2^2}\right)$$

with hyperparameters $\boldsymbol{\theta} = \{\sigma_\mathrm{f}^2, l_1, l_2, \lambda\}$ was used. In [84], a model predictive control scheme for periodic error correction of a telescope mount was developed. A similar MPC controller was considered in [144] for the control of blood glucose for patients, where the GP model was used to model the individual human insulin sensitivity. [125] investigated robot-assisted surgery, where the time-varying reference depends on the patient's breathing motion. A GP model with time as regressor was used to learn this reference and provide a prediction for a MPC controller.

Note again that this type of modeling with Gaussian processes is only possible for periodic or quasi periodic signals because otherwise, given a finite training data set $\mathcal{D} = \{(\boldsymbol{w}^i, z^i)\}_{i\in\mathbb{N}}$ with $z = y$, $\boldsymbol{w} = k$, the input time k will eventually leave the training data set during operation. Then, the posterior mean function will return to the prior mean and thus, will not generate any longer useful predictions.

Learning Static Maps We will use another, application-driven approach, to model and learn a disturbance signal. In Chapter 4, we will deal with image generation of electrostatic potentials of surface nanostructures (single atoms or molecules, or nanoscopic structures built from several atoms/molecules), which depend on spatial coordinates (see Fig. 2.4). These potentials are unknown before operation but a model of them is highly desirable to generate predictions for a controller.[4] The dependence of the electrostatic potential as a function of spatial coordinates can be modeled by general static maps of the form

$$y = f(\boldsymbol{u})$$

with output $y \in \mathbb{R}$ and where the map $f(\cdot)$ depends only on a vector of independent input variables $\boldsymbol{u} \in \mathbb{R}^{n_u}$. It does not depend on time k, nor previous values of y or $\boldsymbol{u}$. For the GP we then have $z = y$ and $\boldsymbol{w} = \boldsymbol{u}$.

Learning such static maps with Gaussian processes has been done in the context of control, e.g., to evaluate and learn unknown cost functions and constraints. For instance, in [123, 124] a linear quadratic regulator (LQR) was used to control a pole balancing example. The unknown cost function depended on the LQR controller parameters and was learned by a GP. The same application case was considered in [45], only that the cost function depended on the parameters of multivariate PID controllers. The approach was generalized in [16, 19], where cost functions of general closed-loop systems in dependence of the controller parameters were learned. In [154], a hierarchical MPC scheme was considered and a GP learned cost function that depended

[4] This explanation is a simplification of the actual image generation process. For more details see Chapter 4.

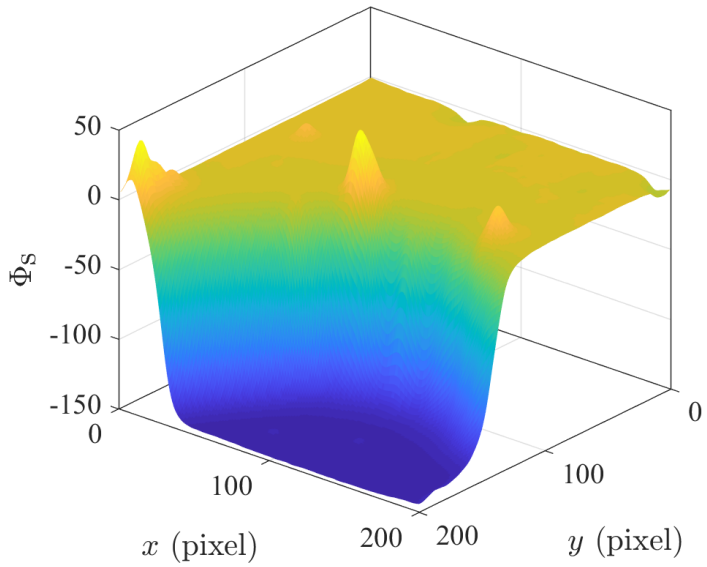


Figure 2.4: Electrostatic potential Φ_s of a nanostructure (measured at a specific distance from the surface) as a function of spatial coordinates x and y. For more details see Chapter 4.

on the prediction horizon and design parameters of an inner control loop. Similarly, the cost function, together with the constraints, was learned in [26]. In [126], force feedback robot control was considered, where a GP learned a function that maps the robot states to the generated force between the robot and its environment.

2.2.2 Dynamic Model Learning

One is often interested in learning of dynamic models (see Fig. 2.3), which we divide into the two classes of input-output models and state space models. Both model classes can be used in model-based controllers such as model predictive control. The input-output models are of particular importance to this thesis.

State Space Models State space models of the form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ y_k &= h(\boldsymbol{x}_k) \end{aligned}$$

with state $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$, input $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$, state dynamics $f(\cdot)$, and output map $h(\cdot)$ are usually the go-to model class for model-based controllers because they can, in principle, capture the complete dynamics of a system. If Gaussian processes are employed to model a state space system, then usually n_x independent GPs are used to model the evolution of each state.[5] To model the j-th state equation, data pairs $(z, \boldsymbol{w})$ with target $z = x_{j,k+1}$ and regressor $\boldsymbol{w} = (\boldsymbol{x}_k, \boldsymbol{u}_k)$ are required to train the j-th GP.

[5] An exception to this approach can be found in [17, 85], where a single GP is used to model the complete state space model. Further explanations can be found in [15]. See also [41] and [161].

When it comes to learning of state space systems with Gaussian processes the approaches can roughly be divided into the following two categories. Into the first category fall approaches that model the process exclusively with a Gaussian process, i.e., no a priori deterministic model is available. One of the first works of this type is [157], where a GP state space model was employed for model-based reinforcement learning. Other works that considered and extended this combination are [42, 43, 185]. A predictive control approach, based on a receding horizon LQR, together with a GP state space model was used in [24] to control a cart-pole system. The approach was also validated in experiments. In [148], a state space GP model was linearized along trajectories and used for policy search via dynamic programming. In [82], model-based reinforcement learning was formulated as a model predictive control problem and combined and experimentally validated with a GP state space model of the controlled processes. The combination of state space models and MPC was applied to linear time-varying systems in [30] and in [31, 32] local linearized GP models were considered for deployment in MPC. In contrast to other works, where the posterior mean function is employed as the model predictor, in [184] various deterministic model instances are drawn from the GP posterior distribution and used in a scenario MPC approach. The works [188, 189] considered general closed-loop systems, i.e., not confined to a specific class of controllers, and established a posteriori stability certificates and regions based on GP state space models.

The second category contains approaches that combine a Gaussian process with a deterministic state space model, often also denoted as hybrid modelling approaches. For instance, using first principles a deterministic base model can be derived. This, however, does not usually capture every aspect of the system dynamics and the GP is then employed to learn the unknown part. A typical example is that a deterministic linear model approximation is available and the remaining nonlinear part is learned by a suitable Gaussian process [77, 176, 203]. One of the first works that combined Gaussian processes and deterministic state space models is [85], where a GP dynamics model was combined with a model of ordinary differential equations to control a blimp via reinforcement learning. Another work is [118], where this model combination was employed in a fault-tolerant MPC approach. The follow-up work [203] used the deterministic base model as the prior mean function to the GP. In [146] and [147], the combination of a GP and deterministic state space model was applied to the control of autonomous vehicles and validated in experiments. In [18], the combined state space model is linearized and used to design a linear robust $\mathcal{H}_\infty$ controller. Application to robot control was considered in [142], where local GP models were considered in different regions of the state space to account for unknown nonlinearities. Further references that also include the topic of guarantees (e.g. safety, stability) are discussed in Section 3.1.3.

Besides the advantages of state space systems modeled by Gaussian processes, which is also expressed in the large number of publications on the topic, there is one major

drawback, which is of general nature and holds for all state space models. In order to be able to obtain an adequate model, data points of all inputs, outputs, and especially states have to be available, i.e., the states have to be accessible, either by measurements or by means of state estimation.

Input-Output Models Another possibility to learn the dynamics of a system with a Gaussian process is to generate input-output models of the form

$$y_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k-m_y}, \boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \ldots, \boldsymbol{u}_{k-m_u}) \ .$$

This model class is denoted as *nonlinear autoregressive model with exogenous input* (NARX) and is an extension of the NAR model (see Section 2.2.1) by an external input $\boldsymbol{u}_k$ and its predecessors. The model order is determined by m_y and m_u, which are, not necessarily but often, chosen to be equal. If, for instance, $m_y = m_u = 1$, then a NARX model of order one is obtained.

Remark 5. *Regarding the GP state space models of the previous subsection, note that each GP that models one state equation $x_{j,k+1} = f_j(\boldsymbol{x}_k, \boldsymbol{u}_k)$ can be considered as a NARX model of order zero.*

There are many examples of Gaussian process-based NARX models employed for control, in particular for model predictive control. For instance, one of the first combinations of Gaussian processes with model predictive control can be found in [91] and [135], where GP-NARX models were employed as the prediction models. Follow up works of the same authors are [90] and [89] that applied these models to other processes, though still in simulations. In [106], the combination of model predictive control and a GP-NARX model was applied for the first time in an experiment for a gas-liquid separation plant. Further works consider, for example, explicit model predictive control [64, 65], the use of local linearized GP models that also include derivative measurements in the training data [13], or the application to building control in [141], where the authors used a hybrid kernel that combined a squared exponential kernel, with the NARX state as input, and a periodic kernel with time as input.

The main advantages, in comparison to state space models, are that no information on internal system states is required and only one GP suffices to model the target output. However, if such a model shall be used in a controller, then certain observability assumptions have to be satisfied such that a NARX model is sufficient to describe the dynamics of a wide class of systems [103]. Furthermore, the performance of NARX models is usually inferior to those of state space models because not the complete dynamics can be captured. This can lead, for instance, to larger settling times of the closed-loop. However, this can be (to a certain extent) accounted for if the model order is increased, i.e., if more previous inputs and outputs are considered.

In Chapter 3, we also consider the use of GP-NARX models for model predictive

control and investigate the question how nominal and robust stability can be guaranteed in the case of an online learned prediction model and model-plant mismatch.

2.3 Summary

In this chapter we presented the necessary foundations of Gaussian processes, on which we build upon in the following chapters. Using the one- and multivariate normal distribution we showed how Gaussian processes generalize the normal distribution to the space of functions and how they can be used in supervised learning to solve the regression problem. In this context, we also presented the equations of the posterior distribution, which will be utilized in Chapters 3 and 4. We furthermore reviewed and categorized the deployment of Gaussian process regression for control in the literature, with a special focus on reference/disturbance learning and learning of dynamic models. Thereby, setting up the frame for the following chapters because in Chapter 3 we use GP dynamic models for model predictive control and in Chapter 4 to learn a disturbance signal that is then used to generate images of electrostatic potentials of nanostructures.

3 Gaussian Process based Model Predictive Control with Guarantees

We present a model predictive control scheme that uses Gaussian process prediction models and is capable of online learning. To this end, we employ a GP-NARX prediction model and derive conditions under which the MPC scheme guarantees recursive constraint satisfaction, as well as nominal and robust stability in the case of online learning and model-plant mismatch. We denote the scheme as recursive Gaussian process model predictive control (rGP-MPC).

The chapter starts with an introduction (Section 3.1) to the topic of model predictive control and its challenges, in particular data-driven generation and adaptation of prediction models with a special focus on Gaussian processes. Afterwards we outline the concept of rGP-MPC. Section 3.2 provides the mathematical basics of model predictive control, together with a detailed discussion of feasibility and stability required later for the main theoretical results of this chapter. In Section 3.3, we formulate the considered control problem and present the online learning scheme of rGP-MPC in Section 3.4 and the resulting optimal control problem in Section 3.5. The stability properties, together with the necessary conditions, are derived in Section 3.6. A practical approach to determine the model predictive control terminal components is presented in Section 3.7. The rGP-MPC scheme is investigated and verified in simulations in Section 3.8.

3.1 Model Predictive Control and Learning

Model predictive control (MPC, [160]) is a control scheme that is naturally capable of dealing with multi-input multi-output systems and that allows taking constraints already in the design process into account. Basically, MPC is reiterated optimal control, where a model of a real process $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ with state $\boldsymbol{x}_k$ and input $\boldsymbol{u}_k$ at time instant k is used for prediction (see Fig. 3.1) and to solve a finite horizon optimal control problem (OCP). The first element of a computed optimal input sequence is applied to the plant and the OCP is re-solved at the next time step. By this means, it is accounted for possible disturbances and uncertainties to the system that occur between two consecutive time steps. Re-solving the OCP at the next time instant shifts the prediction horizon one step into the future, which is why MPC is also denoted as *receding horizon control* [131]. The basic procedure applied by a model predictive control scheme can be summarized as follows:

1. Obtain the state $\boldsymbol{x}_k$ at the current time step k.
2. Solve a finite horizon optimal control problem by means of predicting the system evolution and determining an optimal input sequence such that it minimizes a given cost function.
3. Apply the first part of the optimal input sequence. Go back to 1.

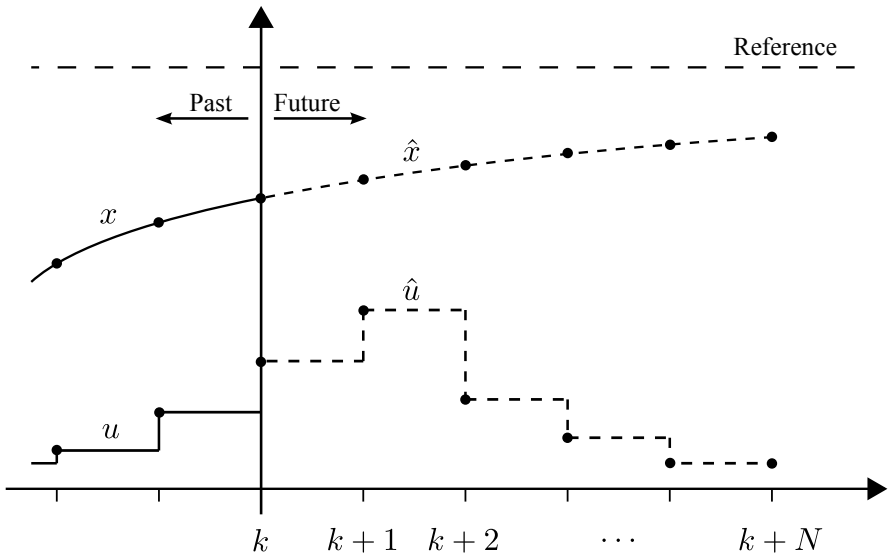


Figure 3.1: MPC illustration: For a particular initial condition $\boldsymbol{x}_k$ at time instant k, a predicted open-loop sequence of inputs $\hat{\boldsymbol{u}}$ is computed up to a prediction horizon N. The input sequence, together with the resulting predicted states $\hat{\boldsymbol{x}}$, are computed in such a way that they minimize a given cost function.

In terms of performance, model predictive control can be superior to other control approaches because the prediction of the process under consideration allows to compute control actions based on future outcomes and also facilitates taking preview information about references and disturbances into account if available. Moreover, in contrast to many other control approaches, constraints on the input, the state, and the output can be directly addressed already in the design process[1]. This has led to manifold scientific interest, as well as practical applications (see, e.g., the reviews [115, 129]). In addition, a mature theoretical framework has been developed over the last decades that enables to provide guarantees regarding, for instance, recursive constraint satisfaction and stability.

However, besides the benefits that model predictive control offers, there are also some challenges. For instance, the standard MPC formulation requires full state information, i.e., the complete state $\boldsymbol{x}_k$ has to be measurable. If this is not the case, then one might employ state estimation methods such as Luenberger observers, Kalman

[1] In other control schemes, such as, for example, state feedback or PID control, constraint satisfaction is accounted for only after the controller design, e.g. by means of anti-windup.

filters, or more advanced estimation techniques, such as moving horizon estimation [159]. If this is also not possible, then output feedback MPC schemes that just utilize the measured system output are an alternative. This type of MPC scheme will also be a subject of this chapter.

In comparison to other control methods, another challenge are the potentially high computational costs of model predictive control. A (possibly nonlinear and nonconvex) optimal control problem must be solved within the given time frame of less than the sampling time. In practice, this originally restricted the deployment of MPC to systems with large time constants, e.g. process industry plants. However, advanced algorithms capable of solving optimal control problems in short time (e.g. acados [187]) and the steadily increasing computational power of digital processors nowadays also allows the deployment of MPC for more demanding applications, e.g. embedded systems for mechatronics [209]. Yet, model predictive control is not only restricted to classical engineering tasks but can be employed as well for such extraordinary applications as HIV treatment strategies in medicine [75] and many more [129].

3.1.1 Prediction Model

For many applications it can be challenging to obtain an appropriate prediction model, which is fundamental to the performance of MPC. Dynamic models for prediction are often based on first principles modeling approaches. Doing so in practice, however, can be very time consuming and therewith expensive or even impossible. Furthermore, if the underlying process or environmental conditions change, a once good model can degrade and therefore needs to be adapted. An alternative to first principles models is to derive prediction models directly from measured data. The resulting models, which are also called black or grey box models [111], can in principle, be learned or refined during operation by including newly available data. Thereby, they are able to account for changing process dynamics or a changing process environment. For this reason, the data-driven approach (in particular the use of Gaussian process learning, see Chapter 2) to generate and update prediction models for MPC is addressed in this chapter.

Although data-driven modeling is not a new field of research, it gained significant attention over the last years due to increasing computational power, the possibility to widely collect data, and the rise of machine learning algorithms, such as neural networks, deep learning, support vector machines, or Gaussian processes. Especially the use of Gaussian processes within model predictive control has attracted significant interest in recent years [31, 84, 91, 121, 147, 203]. However, combining GPs with MPC leads to multiple challenges, e.g., the computational load increases cubically with the number of training data points. This also increases the overall necessary computations to solve the resulting optimal control problem. Furthermore, the utilization of GPs

in an optimal control problem can render the resulting optimization very nonlinear, even for a small number of data points, which increases the probability of obtaining suboptimal or infeasible solutions. Despite these challenges, GPs provide several advantages. For instance, they do not only allow to compute a prediction of the system evolution but also a prediction variance (an effective measure of the uncertainty of the learned model), they are less susceptible to overfitting, and they have, under not too stringent circumstances, universal approximation capabilities for a large class of functions [178], thereby allowing to model the underlying dynamics of a wide variety of nonlinear systems.

3.1.2 Online Learning

As outlined in Section 2.1.2, Gaussian process learning is in general computationally demanding. Thus, online adjustments of GP models can be challenging. In order to reduce the computational load of using and learning Gaussian processes, two main approaches can be distinguished. The first approach basically fixes the maximum number of training data points, while the second approach employs the typically present sparsity [98, 174]. We consider the first approach, which often entails the drawback that the GP might not be able to model the system with sufficient accuracy throughout the full operation space. To compensate for this, one can resort to online learning (or adaptation) of the Gaussian process during operation. This way, also time-varying systems or changing environmental conditions can be accounted for. On the downside, some of the computation time that is saved by reducing the number of training data points is in turn spent by the learning process, which includes updates of the training data set and covariance matrix, recalculation of the covariance matrix inverse, and especially hyperparameter optimization in each time step. While these computationally expensive calculations can be often performed offline, only very few publications exist that combine MPC with online learning of GPs. The required computations take too long to control most processes. Thus, GPs are still mostly trained/learned offline [31, 76, 77, 146]. Exceptions are, for instance, the work [144], where the system has a large time constant in the order of hours or [84], which provides a hyperparameter optimization tailored to the specific application.

In Section 3.4 we present an online learning approach for Gaussian processes that adapts itself only when necessary. The adaptations are performed in a recursive manner, making use of the GP components of the previous step, thereby effectively reducing the computational overhead.

3.1.3 Learning with Guarantees

Independently of how a prediction model is generated (i.e. using first principles or data-driven methods), there is always a certain process-model error or model uncertainty present that can also change over time and which in turn limits the prediction

quality of the model. Furthermore, this uncertainty can destroy the guarantees for recursive constraint satisfaction and stability of MPC (see Sections 3.2.2 and 3.2.3). Traditionally, one way to deal with this situation is to resort to robust MPC schemes, such as, for instance, min-max MPC [167], tube-based MPC [130], multi-scenario approaches [113, 119], or stochastic approaches [151] that take the uncertainty explicitly into account.

In order to establish guarantees for the specific combination of model predictive control and Gaussian processes different approaches exists. One early approach is not to enforce stability by design but to include the GP posterior variance in the cost function of the optimal control problem instead. This avoids steering the plant into regions where the model validity is questionable [13, 89, 135]. In addition, one can perform a posteriori stability verification. For instance, the region of attraction of a closed-loop system is learned in [17] using a combined deterministic and GP model and a Lyapunov function for uncertain systems. Instead of just learning the region of attraction for a fixed closed-loop system, it is furthermore maximized by optimizing the controller parameters in [20]. Another verification approach is given in [188, 189], where invariant sets for the validation of stability in a closed-loop with GP models are calculated. Another possibility is to use invariant *safe sets* and employ a two-layer control framework, where a safe controller is combined with a control policy that optimizes performance [4, 57, 94, 192]. For example, in [4] and [57], a safety framework for general closed-loop systems was developed. The safety guarantees were achieved by the construction of safe regions in the state space using reachability analysis. In [11] and [21] two different prediction models are used in parallel, where the first is a nominal model used to guarantee robust stability using tubes and the other can be a general learning-based model (e.g. a Gaussian process) used to optimize performance. [94] provides (high probability) safety guarantees, utilizing ellipsoidal confidence regions constructed using the GP model and propagating it forward in time. The two-layer framework was extended to three layers in [14]. A tube-based MPC scheme was also considered in [176] together with GPs, which are also used to derive robust stability. To this end, uncertainty sets that are based on the GP variance are used to construct tightened state and input constraint sets. Since the uncertainty sets hold probabilistically, the same is true for the stability result. The aforementioned approaches are based on the assumption of full state information and the use of invariant terminal regions.

On the other hand, nominal MPC, which does not take uncertainty explicitly into account, can itself provide a certain degree of inherent robustness if certain conditions are fulfilled. A suitable stability concept to analyze and describe such properties is *input-to-state stability* (ISS). Input-to-state stability has also been presented in [107] as a unifying framework to robust MPC, generalizing, e.g., results on tube-based and min-max MPC. It was also shown that if a system under a predictive controller is ISS, then this property is preserved even in the case of suboptimal solutions of the

involved optimal control problem. At the expense of a potentially smaller domain of attraction, the advantage of guaranteeing inherent robust stability lies in its simplicity. The already involved ingredients in MPC merely have to satisfy certain properties (e.g. uniform continuity). The aforementioned methods in the literature on the other hand are conceptually more complex and/or more computationally expensive than the nominal MPC case because different control layers with backup controllers are required [4, 14, 57, 94, 192], different prediction models are employed that have to be evaluated in parallel [21], or tubes have to be computed [176].

In Section 3.6, we use the ISS concept for rGP-MPC and derive conditions for inherent robust stability.

Contributions

The contributions associated with the presented rGP-MPC scheme of this chapter are as follows.

- The developed rGP-MPC scheme can be **used for a wide class of application cases**.
  - Processes for which **no physical system model** is available: This is achieved by modeling the process using only measurement data and a Gaussian process regression model (as discussed in Section 2.2.2), which is furthermore learned and refined online during operation. This allows to account for a changing process, process environment, and the effect of a possible limited training data set (this entails limited process knowledge, e.g. lack of training data in important regions of the operation space). The GP prediction error and variance are utilized to determine which data points are added to the training data set.
  - Processes whose **internal states are not accessible** by measurement or estimation: This is achieved using an output feedback formulation. The approach can, however, be easily extended to the state space case.
  - Processes with **fast dynamics** can be handled due to the reduction of the involved computations: The computational load associated with the evaluation and update of the GP equations is reduced via
    - a recursive data inclusion approach with recursive computations of the inverse covariance matrix and the involved Cholesky factor,
    - the possibility to use a limited number of training data points, and
    - excluding hyperparameter optimization.

    The computational load associated with the solution of the optimization problem is reduced
    - by not using a terminal region.

This is especially beneficial in the case when no state constraints are present because then the optimization has to deal with input constraints only.

- Conditions are derived for which the rGP-MPC is equipped with **guarantees**
  - for **recursive constraint satisfaction**, and
  - for nominal and inherent robust (input-to-state) **stability**.

  These guarantees can be extended to general prediction models (including traditional data-based or other machine learning methods) that are learned or updated online.

- A structured **calculation of the terminal MPC components**, based on the linearized GP posterior mean function and linear matrix inequalities.

The core of these results (GP-based prediction model for output feedback MPC, proof of stability) was outlined in [121]. Further contributions (recursive online learning scheme, an extended stability proof, calculation of terminal components) were presented in [120].

3.2 Model Predictive Control

In this section, we provide the mathematical foundation of model predictive control, which will be required to establish stability and recursive constraint satisfaction of rGP-MPC.

In Section 3.2.1, we present the general model predictive control formulation for set-point stabilization[2]. In Section 3.2.2, we discuss feasibility, including initial and recursive feasibility, where many of the theoretical elements are introduced. These also play a key role in establishing stability in Section 3.2.3 and will be extended later to the rGP-MPC case in Section 3.6.

3.2.1 MPC Formulation

We consider discrete[3] time-invariant and constrained nonlinear state space systems of the form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{u}_k &\in \mathcal{U} \\ \boldsymbol{x}_k &\in \mathcal{X} \ , \end{aligned} \tag{3.1}$$

[2]Model predictive control is not restricted to set-point stabilization. For instance, it can also be used for trajectory tracking and path following problems [53, 54].

[3]Model predictive control is often formulated in discrete time because in practice it is always implemented by means of digital processors that work with a finite sampling time. However, the true nature of many systems lies in continuous time and hence, there also exists the corresponding theory for MPC to address these problems [53, 55].

where $k \in \mathbb{N}_0$ denotes the discrete time[4], $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ the state, and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ the input.

The state and input constrained sets are subsets of the respective spaces, i.e., $\mathcal{X} \subset \mathbb{R}^{n_x}$ and $\mathcal{U} \subset \mathbb{R}^{n_u}$. Input constraints usually arise due to actuator saturation, e.g. a valve cannot be opened more than 100% and not less than 0%. State constraints often arise as a consequence of process operation conditions, for example, if for safety reasons a temperature should not exceed a predefined limit.

A particular initial condition $\boldsymbol{x}_k$ at time instant k and a sequence of inputs $\mathbf{u}_k = \{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \ldots\}$ applied to (3.1) result in a state sequence $\mathbf{x}_k = \{\boldsymbol{x}_k, \boldsymbol{x}_{k+1}, \ldots\}$. The state sequence as a whole depends on the initial condition and the applied input sequence, i.e., $\mathbf{x}_k = \mathbf{x}_k(\boldsymbol{x}_k, \mathbf{u}_k)$. However, for the sake of a simpler notation we do not emphasize this dependence.

For the standard MPC formulation, together with its theoretical results as presented in the following, the common objective is regulation to and stabilization of the origin. Thus, it is assumed that the system possesses an equilibrium point at the origin, i.e., $f(0,0) = 0$. From Section 3.3 on we will consider regulation to and stabilization of arbitrary (forced) reference points $\boldsymbol{x}_{\mathrm{ref}}$.

3.2.1.1 Prediction Model

In order to compute the future behavior of the process (3.1), a *prediction* model

$$\begin{aligned} \hat{\boldsymbol{x}}_{k+i+1|k} &= \hat{f}\big(\hat{\boldsymbol{x}}_{k+i|k}, \hat{\boldsymbol{u}}_{k+i|k}\big) \\ \hat{\boldsymbol{x}}_{k|k} &= \boldsymbol{x}_k \end{aligned} \tag{3.2}$$

is used, which is evaluated over a finite prediction horizon $N \in \mathbb{N}$ and where $i \in \mathcal{I}_{0:N-1} = \{0, 1, \ldots, N-1\}$. The variable k denotes the global discrete time, whereas i denotes the internal controller time within the prediction horizon N. The hat notation $\hat{(\cdot)}$ denotes a predicted variable and the subset notation $(\cdot|k)$ emphasizes that the prediction is based on the information available at the specific time step k. The current measurement $\boldsymbol{x}_k = \hat{\boldsymbol{x}}_{k|k}$ serves as the initial condition for the prediction.

Given a predicted input sequence $\hat{\mathbf{u}}_{k|k} = \{\hat{\boldsymbol{u}}_{k|k}, \hat{\boldsymbol{u}}_{k+1|k}, \ldots, \hat{\boldsymbol{u}}_{k+N-1|k}\}$ and initial condition $\boldsymbol{x}_k$, the predicted state evolution is $\hat{\mathbf{x}}_{k|k} = \{\hat{\boldsymbol{x}}_{k|k}, \hat{\boldsymbol{x}}_{k+1|k}, \ldots, \hat{\boldsymbol{x}}_{k+N|k}\}$ with $\hat{\boldsymbol{x}}_{k|k} = \boldsymbol{x}_k$ (see Fig. 3.1). Note that $\hat{\mathbf{x}}_{k|k}$ has $N+1$ elements, whereas $\hat{\mathbf{u}}_{k|k}$ has N elements because no further input is needed in the last stage. The difference between the signal sequences $\mathbf{u}_k$ and $\mathbf{x}_k$ of the real process (3.1) and the sequences $\hat{\mathbf{u}}_{k|k}$ and $\hat{\mathbf{x}}_{k|k}$ of the prediction model (3.2) is that the predicted sequences are computed for each time instant k and used as internal variables of the controller, whereas $\mathbf{u}_k$ and $\mathbf{x}_k$ describe the *applied* control inputs and *actual* system evolution over time.

[4]The discrete time k and the continuous time t are connected by $t = kT$, where T is the sampling period.

3.2.1.2 Cost Function

The model predictive controller determines an open-loop input sequence $\hat{\mathbf{u}}_{k|k}$ by means of minimizing a cost function over the finite horizon N for the initial condition $\boldsymbol{x}_k$. The cost function

$$V_N\big(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}\big) = \sum_{i=0}^{N-1} \ell\big(\hat{\boldsymbol{x}}_{k+i|k}, \hat{\boldsymbol{u}}_{k+i|k}\big) + V_\mathrm{f}\big(\hat{\boldsymbol{x}}_{k+N|k}\big) \tag{3.3}$$

consists of the stage cost $\ell(\boldsymbol{x}, \boldsymbol{u})$ with $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geq 0}$ and the terminal cost $V_\mathrm{f}(\boldsymbol{x})$ with $V_\mathrm{f} : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$. The stage cost penalizes deviations of the predicted state and input evolutions from the origin at each prediction step, whereas the terminal cost emphasizes penalizing the deviation of the last predicted state. To this end, the following Assumption 1 has to be satisfied. Furthermore, as we see in Section 3.2.3.1, both play a key role in establishing stability of the origin.

Assumption 1 (Positive definite cost)**.** *Both the stage and the terminal cost are positive definite and zero at the origin, i.e.,* $\ell(0,0) = 0$, $\ell(\boldsymbol{x}, \boldsymbol{u}) > 0 \ \forall (\boldsymbol{x}, \boldsymbol{u}) \neq (0,0)$, *and* $V_\mathrm{f}(0) = 0$, $V_\mathrm{f}(\boldsymbol{x}) > 0 \ \forall \boldsymbol{x} \neq 0$.

Note that the cost function (3.3) does not explicitly depend on the state sequence $\hat{\mathbf{x}}_{k|k}$ because the latter results from the given initial condition $\boldsymbol{x}_k$ and the input sequence $\hat{\mathbf{u}}_{k|k}$.

A common control task is to drive the state to the origin as fast as possible but at the same time the control energy should not exceed certain limits. Hence, there is an inherent trade-off that has to be addressed when choosing a particular stage cost. Determining the best cost function with respect to the specific control objective is therefore a task of its own [73].

3.2.1.3 Optimal Control Problem

The resulting optimal control problem used in model predictive control is

$$\begin{aligned}
\mathbb{P}_N(\boldsymbol{x}_k): \quad \underset{\hat{\mathbf{u}}_{k|k}}{\text{minimize}} \quad & V_N\big(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}\big) \\
\text{subject to} \quad & \forall i \in \mathcal{I}_{0:N-1}: \\
& \hat{\boldsymbol{x}}_{k+i+1|k} = \hat{f}\big(\hat{\boldsymbol{x}}_{k+i|k}, \hat{\boldsymbol{u}}_{k+i|k}\big) \\
& \hat{\boldsymbol{x}}_{k|k} = \boldsymbol{x}_k \\
& \hat{\boldsymbol{u}}_{k+i|k} \in \mathcal{U} \\
& \hat{\boldsymbol{x}}_{k+i|k} \in \mathcal{X} \\
& \hat{\boldsymbol{x}}_{k+N|k} \in \mathcal{X}_\mathrm{f} \ .
\end{aligned} \tag{3.4}$$

It minimizes the cost function (3.3) at each time step k with respect to the open-loop input sequence $\hat{\mathbf{u}}_{k|k}$ and subject to the state and input constraints, while satisfying the modeled system dynamics (3.2) and the initial condition $\boldsymbol{x}_k$. Additionally, the last predicted state $\hat{\boldsymbol{x}}_{k+N|k}$ is forced to lie in a terminal region $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$ that contains the origin. The terminal region plays a crucial role when it comes to establishing recursive feasibility (see Section 3.2.2.2).

The solution to $\mathbb{P}_N(\boldsymbol{x}_k)$ is denoted by $\hat{\mathbf{u}}^*_{k|k} = \{\hat{\boldsymbol{u}}^*_{k|k}, \hat{\boldsymbol{u}}^*_{k+1|k}, \dots, \hat{\boldsymbol{u}}^*_{k+N-1|k}\}$ where the superscript * indicates the optimal solution. The resulting optimal state sequence is denoted as $\hat{\mathbf{x}}^*_{k|k} = \{\hat{\boldsymbol{x}}^*_{k|k}, \hat{\boldsymbol{x}}^*_{k+1|k}, \dots, \hat{\boldsymbol{x}}^*_{k+N|k}\}$. The optimal cost is $V_N^*(\boldsymbol{x}_k) = V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}^*_{k|k})$ and usually denoted as *value function*. The first element[5] of the optimal input sequence $\hat{\mathbf{u}}^*_{k|k}$, i.e., $\hat{\boldsymbol{u}}^*_{k|k}$, is applied to the process and defines the MPC control law $\boldsymbol{u}_k = \kappa_\mathrm{MPC}(\boldsymbol{x}_k) = \hat{\boldsymbol{u}}^*_{k|k}$. The successor state is then $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_\mathrm{MPC}(\boldsymbol{x}_k))$ and at the next time step, the procedure is repeated and $\mathbb{P}_N(\boldsymbol{x}_k)$ is solved again.

Note that model predictive control differs substantially from conventional control methods because $\boldsymbol{u}_k = \kappa_\mathrm{MPC}(\boldsymbol{x}_k) = \hat{\boldsymbol{u}}^*_{k|k}$ is not a real control law in the usual feedback sense.

As is well known, the recursive solution to an optimal control problem does not automatically lead to stability [81]. One can in general not even guarantee that the problem is *feasible*, i.e., that a solution to the optimal control problem exists at the next time instant. Fortunately, a sound theory has been developed by now that provides the necessary assumptions and components to guarantee feasibility and stability [128, 160]. This usually utilizes concepts from Lyapunov stability and set theory; the latter in particular to guarantee recursive feasibility. In the following subsections we provide insight into the underlying concepts and methods.

3.2.2 Feasibility

Feasibility of model predictive control includes the notion of *initial* and *recursive* feasibility. Initial feasibility basically translates to the existence of a solution to the initial optimal control problem at $k = 0$, whereas recursive feasibility guarantees that if the optimal control problem can be solved at $k = 0$, it can also be solved at all future time steps $k > 0$. We start with the notion of initial feasibility where we introduce the concept of admissible inputs, which in turn leads to the important notion of the feasible set. We also present the first two basic assumptions that will be needed for recursive feasibility and later on stability. Afterwards, we proceed to recursive feasibility, introducing further assumptions.

In the following we consider the nominal case where the prediction model is also considered as the real process, i.e., there is no error between the prediction model $\hat{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ and the plant $f(\boldsymbol{x}_k, \boldsymbol{u}_k)$. For narrative convenience, we drop the estimate notation $\hat{(\cdot)}$ and the dependence on the actual time instant $(\cdot|k)$ for the rest of this

[5] It is also possible to apply more elements of the input sequence, see, e.g., [93, 117].

section. As we are going to discuss feasibility of optimal control problems in general, the difference between the real and predicted variables is not necessary. Occasionally, we write $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ as an abbreviation to $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$. Most of the content presented in the following is based on [107, 108, 160].

3.2.2.1 Initial Feasibility

Initial feasibility of a model predictive control scheme translates to the question whether the initial optimal control problem admits a solution. In static optimization, the problem

$$\underset{\boldsymbol{u}\in\mathcal{U}}{\text{minimize}}\ J(\boldsymbol{u}) \tag{3.5}$$

admits a solution according to *Weierstrass's theorem* if J is a continuous function and if $\mathcal{U}$ is a compact and nonempty set [116, 160]. In this case, (3.5) is said to be feasible and $\mathcal{U}$ is called the feasible set.

In optimal control problems of the form $\mathbb{P}_N(\boldsymbol{x}_k)$, besides the constrained set $\mathcal{U}$, we are facing additional constraints in the form of a dynamical system $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ and the constrained sets $\mathcal{X}$ and $\mathcal{X}_\mathrm{f}$. In addition, one is interested in finding a sequence of control inputs that transfers the initial state $\boldsymbol{x}_k$ within a finite prediction horizon to the terminal region $\mathcal{X}_\mathrm{f}$ (and eventually to the origin) with minimal cost, while satisfying the state and input constraints. An input sequence that satisfies these constraints is called an *admissible*[6] input.

Definition 1 (Admissible input)**.** *An input sequence* $\mathbf{u} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{N-1}\}$ *is said to be admissible if for the initial condition* $\boldsymbol{x}_0$*, the resulting state sequence* $\mathbf{x} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$*, and for all* $i \in \mathcal{I}_{0:N-1}$ *the conditions*

$$\begin{aligned} (i): &\quad \boldsymbol{u}_i \in \mathcal{U} \\ (ii): &\quad \boldsymbol{x}_i \in \mathcal{X} \\ (iii): &\quad \boldsymbol{x}_N \in \mathcal{X}_\mathrm{f} \end{aligned}$$

hold, i.e., $\mathbf{u}$ *satisfies the input constraints and the resulting state sequence* $\mathbf{x}$ *satisfies the state and terminal constraint.*

Note that because $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$, also $\boldsymbol{x}_N \in \mathcal{X}$ holds. Moreover, an admissible input does not necessarily need to minimize the cost function, i.e., it is only a solution candidate for the optimal control problem $\mathbb{P}_N(\boldsymbol{x}_k)$.

Admissibility of an input sequence $\mathbf{u}$ is defined for a specific initial state $\boldsymbol{x}_0$ and there are initial states for which an admissible input exists and others for which no admissible input can be found. This is especially important in model predictive control because progressing with time, the initial value is often the only element that changes

[6] In the literature, admissible inputs are sometimes also called *feasible* inputs.

from the current optimal control problem to the one at the next time step. This motivates the definition of the *feasible set*.

Definition 2 (Feasible set)**.** *The set of all initial states $\boldsymbol{x}_0$ for which there exists at least one admissible input sequence* $\mathbf{u}$ *is called the feasible set*

$$\mathcal{X}_N := \left\{ \boldsymbol{x}_0 \in \mathcal{X} : \ \exists \mathbf{u} \text{ according to Definition 1} \right\} .$$

Due to Definition 1, $\mathcal{X}_N \subseteq \mathcal{X}$.

The optimal control problem $\mathbb{P}_N(\boldsymbol{x}_k)$ can only admit a solution if $\boldsymbol{x}_k \in \mathcal{X}_N$. This, however, is only a necessary but not a sufficient condition because the admissible input sequences associated with the points in $\mathcal{X}_N$ might not minimize the cost function. Hence, further assumptions have to hold to establish a certificate for the existence of an *optimal* solution, i.e., an input sequence that leads to constraint satisfaction and also minimizes the cost function.

Assumption 2 (Continuity of system and cost)**.** *The functions $f(\boldsymbol{x}, \boldsymbol{u})$, $\ell(\boldsymbol{x}, \boldsymbol{u})$, and $V_\mathrm{f}(\boldsymbol{x})$ are continuous.*

The underlying initial value problem $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ with initial condition $\boldsymbol{x}_0$ has to admit a solution. It does so if $f(\boldsymbol{x}, \boldsymbol{u})$ is continuous.

Assumption 3 (Properties of constrained sets)**.** *The set $\mathcal{X}$ is closed and the sets $\mathcal{U}$ and $\mathcal{X}_\mathrm{f}$ are compact (closed and bounded); each set contains the origin.*

With these assumptions, the following proposition can be made.

Proposition 1 (Existence of solutions to OCPs)**.** *Suppose Assumption 2 and 3 hold. Then, for each $\boldsymbol{x}_k \in \mathcal{X}_N$ a solution to $\mathbb{P}_N(\boldsymbol{x}_k)$ exists.*

Proof. The complete proof is given in [160]. The basic idea is that Assumption 2 establishes continuity of the cost function $V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k})$ and Assumption 3 leads to a compact set of admissible input sequences for every $\boldsymbol{x} \in \mathcal{X}_N$. Applying then Weierstrass's theorem yields the existence of a solution to $\mathbb{P}_N(\boldsymbol{x}_k)$ for every $\boldsymbol{x}_k \in \mathcal{X}_N$. □

Proposition 1 guarantees the existence of solutions to optimal control problems for initial values $\boldsymbol{x}_k$ that lie in $\mathcal{X}_N$. This means that if it has to be decided whether a particular OCP admits a solution or not, we merely have to check if the initial condition is contained in $\mathcal{X}_N$, as long as Assumption 2 and 3 are satisfied. Again, the solution which is guaranteed to exist by Proposition 1 satisfies the constraints and minimizes the cost function. If we consider mere constraint satisfaction without minimizing the cost function[7], then Assumption 2 and 3 do not have to be fulfilled; $\boldsymbol{x}_k \in \mathcal{X}_N$ is already a sufficient condition.

[7] This is suboptimal control, see [168] for further information.

Remark 6 (Some remarks regarding Proposition 1)**.**

- *Proposition 1 is a slightly shortened version of the corresponding proposition in [160], where also the case of a closed but unbounded set $\mathcal{U}$ is considered.*

- *If the cost function $V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k})$ is continuous, the value function $V_N^*(\boldsymbol{x}_k)$ is not automatically continuous (see [160]). Furthermore, the conditions to Proposition 1 might be very strong (depending on the specific application case) and one might also want to explicitly allow for discontinuous value functions. Note that this does not necessarily hinder the existence of solutions to optimal control problems (see [7, 69, 160]).*

- *Proposition 1 does not state how $\mathcal{X}_N$ may be computed. In static optimization, it is directly defined by the constraints. In optimal control, it depends on the constraints (e.g. $\mathcal{X}_N \subseteq \mathcal{X}$), the length of the prediction horizon (the longer the horizon N, the larger $\mathcal{X}_N$), and the controllability properties of the given system. For instance, if the origin is an unstable equilibrium and the system is uncontrollable, even without constraints, there does not exist any input sequence $\mathbf{u}$ that will transfer the state to the origin or the terminal region. Thus, $\mathcal{X}_N$ would be empty and the optimal control problem would not admit a solution. In principle, $\mathcal{X}_N$ can only be computed for simple cases, such as linear systems and polyhedral constraints. For further information concerning the computation of $\mathcal{X}_N$, see [23, 160].*

3.2.2.2 Recursive Feasibility

In the preceding section, we have seen which conditions have to be satisfied to establish feasibility of an optimal control problem. MPC is repeated optimal control, where at each time step k the same OCP has to be solved but with a different initial condition $\boldsymbol{x}_k$. The question which then arises in MPC is: If the initial OCP at $k = 0$ has a solution, do all successive OCP also have a solution? An equivalent formulation would be: Does the model predictive controller confine the state to the set $\mathcal{X}_N$? If this is the case, the MPC problem is called recursively feasible.

Unfortunately, initial feasibility does not automatically lead to recursive feasibility, because, according to Definition 2, any point in $\mathcal{X}_N$ admits an admissible sequence pair $(\mathbf{u}, \mathbf{x})$ where $\mathbf{x}$ stays in $\mathcal{X}$ for the considered time interval. However, since $\mathcal{X}_N$ is usually a subset of $\mathcal{X}$, $\mathbf{x}$ might leave the set $\mathcal{X}_N$ at a particular instant in time for an $i \in \mathcal{I}_{0:N-1}$, and for this instant, the OCP does not any longer admit a solution.

Definition 3 (Recursive feasibility)**.** *The MPC controller is recursively feasible if and only if for all initially feasible $\boldsymbol{x}_k \in \mathcal{X}_N$ and for all sequences of admissible control inputs $\mathbf{u}$, the MPC optimization problem remains feasible for all time.*

This definition is sometimes also called *strong recursive feasibility*, whereas a weaker form uses only *optimal* instead of just *admissible* inputs [112]. Note that using admissible input sequences (as in Definition 3) is equivalent to the use of suboptimal solutions, i.e., the resulting input leads, in general, to higher costs as compared to the optimal input [128].

To establish recursive feasibility, as in Definition 3, we first provide some definitions regarding set properties.

Definition 4 (Positive and control invariant sets)**.**

- *A set $\mathcal{S}$ is positive invariant for $\boldsymbol{x}^+ = f(\boldsymbol{x})$ if for all $\boldsymbol{x} \in \mathcal{S}$ we get $\boldsymbol{x}^+ \in \mathcal{S}$.*
- *A set $\mathcal{S}$ is control invariant for $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ if for all $\boldsymbol{x} \in \mathcal{S}$ there exists a $\boldsymbol{u} \in \mathcal{U}$ such that $\boldsymbol{x}^+ \in \mathcal{S}$.*

For recursive feasibility, one has to ensure that the state $\boldsymbol{x}$ stays in the set $\mathcal{X}_N$ for all time instants, i.e., $\mathcal{X}_N$ must be control invariant for $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ with the model predictive control law $\boldsymbol{u} = \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k)$. The necessary assumption on this regard is the following.

Assumption 4 (Control invariant terminal region)**.** *The terminal region $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$ is compact, contains the origin, and is control invariant for $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{u} \in \mathcal{U}$.*

Proposition 2 (Recursive feasibility of MPC)**.** *Suppose Assumption 2 to 4 hold. Then $\mathcal{X}_N$ is positive invariant for $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k))$.*

Proof. For the complete proof to Proposition 2 see [160]. The underlying idea is to begin with $\mathcal{X}_0 := \mathcal{X}_\mathrm{f}$ and set

$$\mathcal{X}_1 = \{\boldsymbol{x} \in \mathcal{X} : \ \exists \boldsymbol{u} \in \mathcal{U} \text{ such that } f(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_0\} \ . \tag{3.6}$$

So $\mathcal{X}_1$ is the set of all states for which a control can be found such that the state is moved to $\mathcal{X}_\mathrm{f}$. This implies that $\mathcal{X}_0 \subseteq \mathcal{X}_1$. Since $\mathcal{X}_\mathrm{f}$ is control invariant by Assumption 4, it follows that also $\mathcal{X}_1$ is control invariant. By backward recursion and induction, one arrives at the conclusion that $\mathcal{X}_N$ is positive invariant. Note that the existence of a corresponding $\boldsymbol{u}$ for all $\boldsymbol{x} \in \mathcal{X}_N$ in (3.6) is guaranteed by Proposition 1. □

Hence, recursive feasibility can be guaranteed for initial conditions that lie in $\mathcal{X}_N$ and a suitable terminal region $\mathcal{X}_\mathrm{f}$. Obviously we have $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}_N \subseteq \mathcal{X}$ (Fig. 3.2). This terminal region must be control invariant with some $\boldsymbol{u} \in \mathcal{U}$, i.e., we assume the terminal region is rendered control invariant by a controller that is usually denoted as *terminal controller* $\kappa_\mathrm{f}(\boldsymbol{x})$. In Section 3.7, one possibility is shown how to compute a terminal controller together with the corresponding terminal region.

So the key idea for establishing recursive feasibility is to steer the state with MPC to the terminal region $\mathcal{X}_\mathrm{f}$ and then switch to the terminal controller that renders

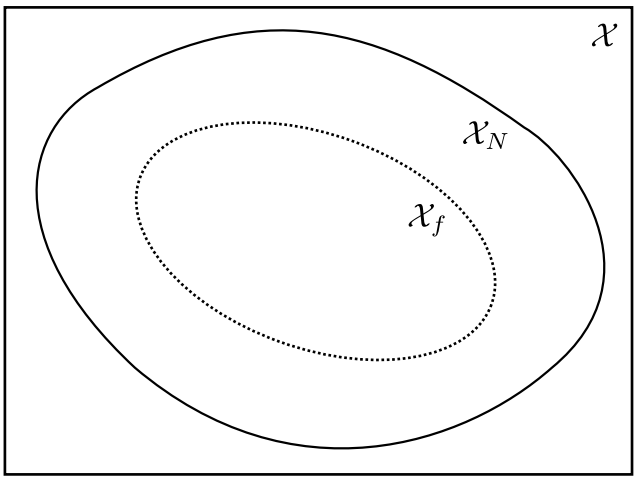


Figure 3.2: Illustration of the constrained set $\mathcal{X}$, the set of feasible initial conditions $\mathcal{X}_N$, and the terminal region $\mathcal{X}_\mathrm{f}$.

$\mathcal{X}_\mathrm{f}$ invariant. However, note that the terminal controller is usually not applied (the MPC control law is applied the whole time), it is more of a tool to establish recursive feasibility.

3.2.3 Stability

In 1960, R.E. Kalman stated in his famous paper [81] that optimality does not automatically lead to stability. Since then, different approaches to establish stability of model predictive control have been proposed. For a good and short overview of the different stabilizing MPC approaches see [39, 131], for a historical overview [117].

In this section, we present some of these literature results because they establish the basis for stability of the rGP-MPC scheme. In Section 3.2.3.1, we focus on establishing nominal stability in the classical way, where the key ingredients are the terminal region $\mathcal{X}_\mathrm{f}$, the terminal cost function $V_\mathrm{f}(\boldsymbol{x})$, and a certain type of stage cost $\ell(\boldsymbol{x}, \boldsymbol{u})$. In Section 3.2.3.2, we move on to robust stability, using the concept of input-to-state stability.

3.2.3.1 Nominal Stability

We provide two results that establish stability of the origin under a model predictive controller. The first (classical approach) uses the components of MPC (including the terminal region $\mathcal{X}_\mathrm{f}$) to establish stability. Lyapunov theory (see Appendix A.2.1) is used to establish stability of the origin for the closed-loop system $\boldsymbol{x}^+ = f(\boldsymbol{x}, \kappa_\mathrm{MPC}(\boldsymbol{x}_k))$. The key therein is to use the value function as a Lyapunov function candidate. In the following, we provide the necessary conditions for the fully constrained case (input and state constraints) that make the value function indeed a Lyapunov function. The second approach extends the results to the case of an optimal control problem without terminal region $\mathcal{X}_\mathrm{f}$.

Assumption 5 (Bounds on stage and terminal costs)**.** *The stage cost $\ell(\boldsymbol{x}, \boldsymbol{u})$ and the terminal cost $V_\mathrm{f}(\boldsymbol{x})$ satisfy*

$$\begin{aligned} \ell(\boldsymbol{x}, \boldsymbol{u}) &\geq \alpha_1(\|\boldsymbol{x}\|) && \forall \boldsymbol{x} \in \mathcal{X}_N,\ \forall \boldsymbol{u} \in \mathcal{U} \\ \alpha_2(\|\boldsymbol{x}\|) \leq V_\mathrm{f}(\boldsymbol{x}) &\leq \alpha_3(\|\boldsymbol{x}\|) && \forall \boldsymbol{x} \in \mathcal{X}_\mathrm{f} \end{aligned}$$

in which $\alpha_1(\cdot)$, $\alpha_2(\cdot)$, and $\alpha_3(\cdot)$ are $\mathcal{K}_\infty$ functions.

A $\mathcal{K}_\infty$ function is a function that is continuous, strictly increasing, zero at zero, and unbounded (see Definition 10 in Appendix A.1). Therefore, Assumption 5 is an extension to the positive definiteness of Assumption 1 and hence, completely implies Assumption 1.

Assumption 6 (Basic stability assumption)**.** *For all $\boldsymbol{x} \in \mathcal{X}_\mathrm{f}$, there exists an admissible $\boldsymbol{u} \in \mathcal{U}$ satisfying*

$$\begin{aligned} \boldsymbol{x}^+ &\in \mathcal{X}_\mathrm{f} \\ V_\mathrm{f}(\boldsymbol{x}^+) &\leq V_\mathrm{f}(\boldsymbol{x}) - \ell(\boldsymbol{x}, \boldsymbol{u}) \end{aligned}$$

with $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$.

For all points within the terminal region, the successor point is also contained in that region and has less or equal cost. Thus, Assumption 6 implies partially Assumption 4. Note that Assumption 5 and 6 ensure that $V_\mathrm{f}(\boldsymbol{x})$ is a Lyapunov function on $\mathcal{X}_\mathrm{f}$ for the system $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ and therefore, the origin is asymptotically stable on the positive invariant set $\mathcal{X}_\mathrm{f}$ (see Appendix A.2), while satisfying state and input constraints. Such a Lyapunov function can be constructed if a, at least, local valid description of the process at the origin is available. This can, for instance, be a linearized version of the nonlinear prediction model, see also Section 3.7.

With the given assumptions, the following theorem can be established.

Theorem 1 (MPC stability [160])**.** *Suppose that Assumption 2 to 6 are satisfied and let $\kappa_\mathrm{MPC}(\boldsymbol{x}_k)$ be the predictive controller resulting from the solution to the optimal control problem* (3.4)*. Then, the origin is asymptotically stable with a region of attraction $\mathcal{X}_N$ for the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_\mathrm{MPC}(\boldsymbol{x}_k))$.*

Proof. For the complete proof of Theorem 1 see [160]. Assumption 2 to 4 need to be satisfied to guarantee that $\mathbb{P}_N(\boldsymbol{x}_k)$ is recursively feasible. Assumption 5 and 6 ensure that the value function $V_N^*(\boldsymbol{x}_k)$ is a Lyapunov function (see Definition 14 in Appendix A.2.1) for the closed-loop system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_\mathrm{MPC}(\boldsymbol{x}_k))$ on the set $\mathcal{X}_N$. Hence, the origin is asymptotically stable in the region of attraction $\mathcal{X}_N$. □

Note that here, the region of attraction, in which the origin is asymptotically stable, is equal to the feasible set $\mathcal{X}_N$, i.e., it is the largest possible region of attraction. However, as we will see below, the region of attraction can also be smaller than the feasible set.

Stability without Terminal Region

In the following, we review a result from [108], which will be important for the forthcoming developments. It establishes stability without the terminal region $\mathcal{X}_\mathrm{f}$ being part of the optimal control problem.

Consider the following optimal control problem

$$\begin{aligned} \underset{\hat{\mathbf{u}}_{k|k}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell\big(\hat{\boldsymbol{x}}_{k+i|k}, \hat{\boldsymbol{u}}_{k+i|k}\big) + \lambda V_\mathrm{f}\big(\hat{\boldsymbol{x}}_{k+N|k}\big) \\ \text{subject to} \quad & \forall i \in \mathcal{I}_{0:N-1} : \\ & \hat{\boldsymbol{x}}_{k+i+1|k} = \hat{f}\big(\hat{\boldsymbol{x}}_{k+i|k}, \hat{\boldsymbol{u}}_{k+i|k}\big) \\ & \hat{\boldsymbol{x}}_{k|k} = \boldsymbol{x}_k \\ & \hat{\boldsymbol{u}}_{k+i|k} \in \mathcal{U} \\ & \hat{\boldsymbol{x}}_{k+i|k} \in \mathcal{X} \ , \end{aligned} \tag{3.7}$$

which differs from OCP (3.4) in two aspects. First, the terminal constraint $\hat{\boldsymbol{x}}_{k+N|k} \in \mathcal{X}_\mathrm{f}$ is dropped and second, the terminal cost V_f is multiplied by a factor $\lambda \geq 1$.

To guarantee recursive feasibility and stability, the following assumptions are required.

Assumption 7. *Let $V_\mathrm{f}(\boldsymbol{x})$ be a Lyapunov function and let the terminal region be a level set of the terminal cost $\mathcal{X}_\mathrm{f} = \{\boldsymbol{x} \in \mathbb{R}^{n_x} : V_\mathrm{f}(\boldsymbol{x}) \leq \nu\}$, with $\nu > 0$, such that $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$.*

As before, Assumption 5 and 6 ensure that $V_\mathrm{f}(\boldsymbol{x})$ is a Lyapunov function on $\mathcal{X}_\mathrm{f}$. Assumption 7 introduces a direct connection between $\mathcal{X}_\mathrm{f}$ and V_f by defining the terminal region via the terminal cost and introducing the parameter ν that is utilized further below.

Assumption 8. *Let d be a positive constant such that $\ell(\boldsymbol{x}, \boldsymbol{u}) > d$ for all $\boldsymbol{x} \notin \mathcal{X}_\mathrm{f}$ and for all $\boldsymbol{u} \in \mathcal{U}$.*

Using these assumptions, the following stability result can be formulated.

Theorem 2 (MPC stability without terminal region [108])**.** *Suppose that Assumption 2 to 8 are satisfied and let $\kappa_\mathrm{MPC}(\boldsymbol{x}_k)$ be the predictive controller resulting from the solution to the optimal control problem* (3.7) *without terminal region. Then, the origin is asymptotically stable with a region of attraction $\Upsilon(\lambda) = \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k) \leq N \cdot d + \lambda \cdot \nu\}$ for the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_\mathrm{MPC}(\boldsymbol{x}_k))$.*

Proof. Theorem 2 is a reformulation of Theorem 3 and its preceding results in [108]. It is first shown that the set $\Upsilon = \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k) \leq N \cdot d + \nu\}$ is a region of attraction of the origin for the optimal control problem without terminal constraint. Afterwards, it is shown that this region can be enlarged by multiplying the terminal

cost with the factor $\lambda \geq 1$ resulting in the region of attraction $\Upsilon(\lambda) = \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k) \leq N \cdot d + \lambda \cdot \nu\}$. At last, it is established that for any state of the feasible set $\boldsymbol{x}_k \in \mathcal{X}_N$ of (3.7), there exists a $\lambda \geq 1$ such that $\boldsymbol{x}_k \in \Upsilon(\lambda)$. □

Note that here, the region of attraction is not equal to but only a subset of the feasible set, i.e., $\Upsilon(\lambda) \subset \mathcal{X}_N$.

The advantages of an optimal control problem without terminal region are two-fold.

1. Practical: The necessary constraints are reduced. This is particularly interesting if no further state constraints are considered, i.e., $\mathcal{X} = \mathbb{R}^{n_x}$. Then, only input constraints $\mathcal{U}$ have to be considered, which makes the optimal control problem computationally much easier to solve.

2. Theoretical: For any $\boldsymbol{x}_k \in \Upsilon(\lambda)$, the resulting state sequence stays in $\Upsilon(\lambda)$ because it is invariant. Hence, the state constraints $\mathcal{X}$ never become active. For non-active state constraints, it can be shown that the value function $V_N^*(\boldsymbol{x}_k)$ is uniformly continuous in $\boldsymbol{x}_k$ [107].

3.2.3.2 Robust (input-to-state) Stability

So far, a perfect prediction model (without uncertainties) was assumed. Now we move on to the case that the process $f(\cdot)$ and the (nominal) prediction model $\hat{f}(\cdot)$ are not the same, i.e., there is some uncertainty associated with the prediction model. The causes of uncertainty can be, for instance, noise, unknown disturbances, uncertain model parameters, or model-plant mismatch. We denote this uncertainty by a general error signal $\boldsymbol{e}_k$ and implicitly account for it by writing the process as $f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{e}_k)$. This leads to the closed-loop system

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), \boldsymbol{e}_k) \ , \tag{3.8}$$

employing the model predictive controller $\kappa_{\mathrm{MPC}}(\boldsymbol{x}_k)$ to the real process.

In order to maintain the guarantees of recursive feasibility and stability in the case of uncertainty, different robust MPC schemes have been developed. *Input-to-state* stability can be utilized to establish when the optimal control problem is inherently robust, i.e., the nominal MPC scheme is stabilizing despite (possibly small) uncertainties. For some basic definitions and results with respect to input-to-state stability we refer to Appendix A.2.2.

Remark 7. *We use the general implicit nomenclature $f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{e}_k)$ to account for the uncertainty, similar to [107]. In Section 3.6, we will consider the specific form $f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{e}_k) = \hat{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{e}_k$.*

Definition 5 (MPC input-to-state stability)**.** *The origin of the closed-loop* (3.8) *is input-to-state stable (ISS) if there exist a $\mathcal{KL}$-function*[8] $\beta(\cdot,\cdot)$ *and a $\mathcal{K}$-function* $\gamma(\cdot)$ *such that*

$$\|\boldsymbol{x}_k\| \leq \beta(\|\boldsymbol{x}_0\|, k) + \gamma\Big(\sup_{k\geq 0}\|\boldsymbol{e}_k\|\Big)$$

holds for all initial states $\boldsymbol{x}_0$, all uncertainties $\boldsymbol{e}_k$, and for all $k \in \mathbb{N}_0$.

Input-to-state stability combines nominal stability as well as uniformly bounded influence of uncertainty in a single condition. It implies asymptotic stability of the undisturbed (nominal) system (with $\boldsymbol{e}_k \equiv 0$) and a bounded effect of the uncertainty on the state evolution. Furthermore, if the error signal $\boldsymbol{e}_k$ fades, the uncertain system asymptotically converges to the equilibrium.

Assumption 9 (Bounded uncertainty)**.** *Assume that $\boldsymbol{e}_k \in \mathcal{E}$ $\forall k$, where $\mathcal{E}$ is a compact set that contains the origin.*

This assumption is required to be able to still guarantee feasibility and recursive constraint satisfaction. If the uncertainty was unbounded and counteracted the controller, it would be impossible to guarantee these properties.

The following theorem is a restatement of Theorem 4 in [107] for the case of an optimal control problem without terminal region. This will serve as a starting point in Section 3.6.2 to prove input-to-state stability of the rGP-MPC scheme.

Theorem 3 (MPC ISS [107])**.** *Suppose that Assumption 2 to 9 are satisfied. Let $\kappa_{\mathrm{MPC}}(\boldsymbol{x})$ be the predictive controller resulting from the solution to the optimal control problem* (3.7) *without terminal region and let $\mathcal{X}_N$ be its feasible region. Let furthermore the true process $f(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{e}_k)$ be uniformly continuous in $\boldsymbol{e}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$, all $\boldsymbol{e}_k \in \mathcal{E}$, and all $\boldsymbol{u}_k \in \mathcal{U}$. If one of the two conditions*

1. *The optimal cost $V_N^*(\boldsymbol{x}_k)$ is uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$.*
2. *The nominal model $\hat{f}(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k))$ is uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$.*

holds, then the closed-loop system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), \boldsymbol{e}_k)$ controlled by the nominal predictive controller fulfills the ISS property in a robust invariant set $\Omega \subseteq \mathcal{X}_N$ for a sufficiently small bound of the uncertainties.

Proof. The complete proof is provided in [107], where Theorem 2 considers the case of general closed-loop systems, i.e., with arbitrary controllers, and Theorem 4 considers the case of a closed-loop controlled by model predictive control.

[8] See Appendix A.1 for the definition.

1. It is shown in Condition 1 of Theorem 2 (Condition 2 of Theorem 4) that if there exists a uniformly continuous Lyapunov function (the value function in MPC) for the nominal system and if the closed-loop $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), \boldsymbol{e}_k)$ is uniformly continuous in $\boldsymbol{e}_k$ for all $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$, then the Lyapunov function is a ISS-Lyapunov function and the origin of the closed-loop is input-to-state stable (see Appendix A.2.2).

2. In the proof regarding Condition 2, it is shown how to (through a clever construction) obtain a general Lyapunov function for the nominally asymptotically stable system. Using the assumption of Condition 2 that the nominal model is uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ it is shown that this Lyapunov function is uniformly continuous, which was the assumption of Condition 1.

□

Remark 8. *One might conclude that Condition 1 and Condition 2 are independent (i.e. uniform continuity of the Lypapunov function and of the nominal model), which is, however, not the case. In fact, Condition 2 and the corresponding proof show one way how Condition 1 can be fulfilled.*

Besides the results in [107], there is also a proposition that establishes under which conditions the optimal cost function is uniformly continuous. We omit these results and refer to Section 3.6.2, where we will utilize them to prove input-to-state stability of the rGP-MPC scheme.

3.3 Control Problem Formulation

We consider nonlinear discrete-time systems that can be represented by nonlinear autoregressive models with exogenous input (NARX, see also Section 2.2.2)

$$y_{k+1} = f(\boldsymbol{x}_k, u_k) + \epsilon \tag{3.9a}$$
$$\text{s.t. } u_k \in \mathcal{U} \tag{3.9b}$$
$$y_k \in \mathcal{Y} \ . \tag{3.9c}$$

Here, $u_k \in \mathbb{R}$ denotes the input, $y_k \in \mathbb{R}$ the output, and $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the NARX "state vector"

$$\boldsymbol{x}_k = \begin{bmatrix} y_k & y_{k-1} & \cdots & y_{k-m_y} & u_{k-1} & \cdots & u_{k-m_u} \end{bmatrix}^\mathsf{T} \tag{3.10}$$

that consists of the current and past outputs and inputs, and where m_y, m_u determine the NARX model order $n_x = m_y + m_u + 1$. The output is corrupted by Gaussian noise[9] $\epsilon \sim \mathcal{N}(0, \sigma_\mathrm{n}^2)$ with zero mean and noise variance σ_n^2. Input and outputs are restricted

[9] In real processes the measurement noise is always bounded, e.g., due to the limitations of the involved data acquisition systems. For the moment, however, we consider the case of normal unbounded Gaussian noise.

to lie in the constraint compact sets $\mathcal{U} \subset \mathbb{R}$ and $\mathcal{Y} \subset \mathbb{R}$, where $\mathcal{U}$ are hard constraints and $\mathcal{Y}$ can be hard or soft constraints that we denote by $\mathcal{Y}_\mathrm{h}$ and $\mathcal{Y}_\mathrm{s}$ respectively. The NARX state and the output are connected via $y_k = \boldsymbol{c}^\mathsf{T}\boldsymbol{x}_k$ with $\boldsymbol{c}^\mathsf{T} = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}$.

The considered control objective is set-point stabilization and optimal set-point change, i.e., we want to steer the system from an initial point $(\boldsymbol{x}_0, u_0)$ to a target reference point $(\boldsymbol{x}_\mathrm{ref}, u_\mathrm{ref})$, while satisfying the constraints and stabilizing the system at the target. We employ model predictive control, which uses the prediction model

$$\hat{y}_{k+1} = \hat{f}(\boldsymbol{x}_k, u_k) \tag{3.11}$$

that is capable of predicting future output values of (3.9a) with sufficient accuracy. The model shall be based on input-output measurement data. The approach should furthermore be able to adapt to a changing process or process environment and shall apply to a wide variety of processes, including those with fast dynamics. Hence, the computational load of the approach has to be reduced. We outline an approach to learn the system model approximation $\hat{f}(\boldsymbol{x}_k, u_k)$ from measured input-output data using a Gaussian process that is capable of online learning during operation, based on newly available data.

Note that the real process has also a (possibly unknown) state space representation. We choose the input-output formulation because this will be also the nature of the prediction model. In comparison with the state space formulation of the previous section, the input-output formulation of (3.9) is more general in the sense that it allows applying the developed control scheme also to systems, whose state is not fully accessible, thereby allowing to apply the control scheme to a broader class of processes. However, this generality comes at the cost of a usually inferior performance because the process dynamics can be captured only to a certain extent.

Remark 9. *For simplicity of presentation we consider a NARX model with one output modeled by a Gaussian process. It can be extended to more outputs or state space models, where for each output/state an individual GP is used [84, 146, 147]. Note that the theoretical results obtained in Section 3.6 are also valid, without limitation, for the multi-output case.*

3.4 Online Learning of Gaussian Process based Prediction Models

We outline an approach to learn a prediction model only from measured input-output data using a Gaussian process to tackle the control problem formulated in Section 3.3. This results in a GP-NARX prediction model, which is used in a model predictive control scheme.

In Section 2.1.2 we reviewed Gaussian process regression, where the posterior mean function $m_+(\boldsymbol{w}|\mathcal{D})$ (2.6a) is the desired estimator of an unknown output latent func-

tion. We will use the posterior mean to model the NARX process function $f(\boldsymbol{x}_k, u_k)$ of (3.9a), leading to the MPC prediction model

$$\hat{y}_{k+1} = \hat{f}(\boldsymbol{x}_k, u_k) := \hat{z} = m_+(\boldsymbol{w}_k|\mathcal{D}_k) \ , \tag{3.12}$$

where the regressor $\boldsymbol{w}$ and the training data set $\mathcal{D}$ now depend on the time k. In order to cope with the requirement of adapting to changing processes and process environments, we present an online learning approach of the Gaussian process, based on the concept of evolving GPs (Section 3.4.1) that will update the training data set during operation whenever beneficial. For this reason $\mathcal{D}_k$ might depend on k and therefore, the covariance matrix and its inverse have to be recalculated online. For numerical stability and efficiency, we use the Cholesky factor decomposition to compute the covariance matrix inverse (Section 3.4.2). To reduce the computational load, we update the involved Cholesky factor recursively, making use of the previously available factor.

Remark 10. *As we aim for the potential application of the resulting control scheme to fast processes, we do not consider online hyperparameter optimization because this is the computationally most expensive part and cannot be dealt with in a recursive fashion.*

3.4.1 Evolving Gaussian Processes

In order to efficiently refine the GP model online, we seek to update the training data set $\mathcal{D}_k$, possibly at each time step k during operation. To this end, we resort to the concept of so-called *evolving* GPs [87, 153], which can be used, for instance, if the underlying system to be modeled by a GP is time-varying and needs to be adjusted/learned online or if the training data is only available for certain regions of the operating space and one wants to expand operation beyond these regions. Basically one has Gaussian processes whose training data set $\mathcal{D}_k$ is updated online using some type of information criterion. Different criteria can be used to select new data points to be added and already existing points to be removed if necessary.

The general idea is to include an incoming data point to the training data set only if it contributes enough new valuable information, which can be defined in different ways and depends on the specific application. Possible options are the use of the information gain, entropy difference, or the expected likelihood [170, 173]. We employ the Gaussian process as a prediction model in model predictive control and are therefore particularly interested in how accurate the current model can predict the output value at the next time step and how confident this prediction is. To this end, given a new data point $(\boldsymbol{w}_k, y_{k+1})$ of current regressor $\boldsymbol{w}_k$ and resulting output y_{k+1}, we first define the prediction error via

$$e^{\mathrm{p}} := y_{k+1} - \hat{y}_{k+1} = f(\boldsymbol{x}_k, u_k) + \epsilon - m_+(\boldsymbol{w}_k|\mathcal{D}_k) \ , \tag{3.13}$$

and use the following rule that determines a new training data candidate $\mathcal{D}'_{k+1}$.

Rule 1 (New training data set candidate)**.** *At the current time step k with regressor $\boldsymbol{w}_k$ and training data set $\mathcal{D}_k$ compute posterior mean $\hat{y}_{k+1} = m_+(\boldsymbol{w}_k|\mathcal{D}_k)$ and posterior variance $\sigma_+^2 = \sigma_+^2(\boldsymbol{w}_k|\mathcal{D}_k)$. Once the next output y_{k+1} is available, the new data point is $(\boldsymbol{w}_k, y_{k+1})$ and*

if $|e^{\mathrm{p}}| > \bar{e}^{\mathrm{p}}$ *OR* $\sigma_+^2 > \bar{\sigma}^2$ ***then***
$\mathcal{D}'_{k+1} = \mathcal{D}_k \cup (\boldsymbol{w}_k, y_{k+1})$
end if

where $\bar{e}^{\mathrm{p}}$ and $\bar{\sigma}^2$ are pre-specified thresholds and $\mathcal{D}'_{k+1}$ is the new training data set candidate for $k+1$.

Thus, if the prediction error e^{p} is larger than the threshold $\bar{e}^{\mathrm{p}}$, the data point is considered to be included in the training data set $\mathcal{D}_k$ because the current posterior model is not able to predict the output with the specified accuracy. If it is smaller but the resulting posterior variance $\sigma_+^2(\boldsymbol{w}|\mathcal{D}_k)$ is larger than the threshold $\bar{\sigma}^2$, the data point is also a candidate because the current posterior model is not sufficiently confident in its prediction. This allows including data points that are relevant to attain a certain prediction quality and to effectively limit the necessary number of data points in $\mathcal{D}_k$. This becomes especially important for long operation times and many encountered data points with new information during operation.

Remark 11. *Since update Rule 1 also considers outliers for inclusion, we propose to combine it with an additional update rule presented in Theorem 4 (Section 3.6.1) for filtering. The application order of both update rules is also contained in Algorithm 1.*

As the available computational power is always limited and depending on the specific system, this can require the limitation of the maximum number of points in $\mathcal{D}_k$ to a constant number $\bar{n} \in \mathbb{N}$.[10] If this limit is reached, data points have to be removed to maintain the size of $\mathcal{D}_k$. Again, different criteria can be employed to determine which data point shall be deleted. For instance, the point in the training data set with the lowest benefit for the model quality (e.g. the data point that is most accurately predicted under the current posterior) can be deleted. This however can be computationally expensive because the prediction has to be evaluated for every of the $\bar{n}+1$ training data points at each time instant k. For online implementation, we employ a more simple approach that deletes the oldest point contained in $\mathcal{D}_k$. Note that the calculations do not need to be performed at all times. They could also be performed in larger time intervals.

Remark 12. *The concept of evolving GPs, in particular the outlined data handling approach, leads to a training data set $\mathcal{D}_k$ that captures the system dynamics in an (evolving) subregion of the whole operating region. Thus, information about already*

[10] This approach is also sometimes denoted as *truncated* GP [33].

visited regions can, in principle, be lost when moving towards other regions and have to be regained when visited again. This could be counteracted, for instance, by exploiting multiple GPs for different regions or by GP blending [21, 132].

Remark 13. *In principle, the smaller the thresholds $\bar{e}^{\mathrm{p}}$ and $\bar{\sigma}^2$, the better the prediction. However, then also the overhead for the computational evaluation for adding and removing data points increases. In addition, the smaller the thresholds, the smaller the region in which the training data set captures the system behavior, given the case that only a finite number of training data points is allowed. Hence, the selection of the thresholds $\bar{e}^{\mathrm{p}}$ and $\bar{\sigma}^2$ is an application specific trade-off and might be chosen heuristically by the user. Some general guidelines are (i), a lower bound for $\bar{\sigma}^2$ is the measurement noise variance and (ii), $\bar{e}^{\mathrm{p}}$ could be chosen proportional to $1/\bar{n}\sum_{i=1}^{\bar{n}}|\boldsymbol{z} - m_+(\boldsymbol{W}|\mathcal{D}_k)|$, i.e., to the mean value of all the absolute values of the prediction errors, based on the current training data set $\mathcal{D}_k$. In the same way $\bar{\sigma}^2$ could be chosen.*

3.4.2 Avoiding Numerical Ill Conditioning by Cholesky Decomposition

The squared exponential covariance function (2.7), as well as other smooth covariance functions, can lead to numerical problems when computing the inverse $\boldsymbol{K}^{-1}$ [138, 145]. The reason is that smoothness requires that two nearby points are strongly correlated. In that case, the corresponding rows/columns of these points in $\boldsymbol{K}$ are very similar, which leads to a large condition number (the ratio of the largest and smallest eigenvalue). This results in numerical problems when computing the inverse $\boldsymbol{K}^{-1}$ with computational cost $\mathcal{O}(n^3)$, as required for the posterior mean (2.6a), the posterior variance (2.6b), or the log likelihood (2.8). These problems become even worse if (2.6a) and (2.6b) are nested within an optimization procedure like model predictive control. One way to alleviate this problem is by adding an additional noise or *jitter* term to the diagonal of the covariance matrix [138]. A more effective approach, however, is to avoid the numerical instabilities that arise in the explicit computation of the matrix inverse by performing the required computations using the Cholesky decomposition, which is numerically more stable.

Given a system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ with a symmetric positive matrix $\boldsymbol{A} = \boldsymbol{A}^{\mathsf{T}}$, we denote the solution by $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} := \boldsymbol{A}\backslash\boldsymbol{b}$. The Cholesky decomposition of $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}$, where $\boldsymbol{R} = \mathrm{chol}(\boldsymbol{A})$ is an upper triangular matrix that is called the *Cholesky factor*. It can be used to obtain the solution to the linear equations system via $\boldsymbol{x} = \boldsymbol{R}\backslash(\boldsymbol{R}^{\mathsf{T}}\backslash\boldsymbol{b})$.

In order to use the Cholesky factor to calculate the posterior mean $m_+(\boldsymbol{w}|\mathcal{D})$ (2.6a) and variance $\sigma_+^2(\boldsymbol{w}|\mathcal{D})$ (2.6b), we define

$$\boldsymbol{\alpha} := \boldsymbol{K}^{-1}(\boldsymbol{z} - m(\boldsymbol{W}))$$
$$\boldsymbol{\beta} := \boldsymbol{K}^{-1}\varrho(\boldsymbol{W}, \boldsymbol{w})$$

and obtain

$$\begin{aligned} m_+(\boldsymbol{w}|\mathcal{D}) &= m(\boldsymbol{w}) + \varrho(\boldsymbol{w}, \boldsymbol{W})\boldsymbol{\alpha} \\ \sigma_+^2(\boldsymbol{w}|\mathcal{D}) &= \varrho(\boldsymbol{w}, \boldsymbol{w}) - \varrho(\boldsymbol{w}, \boldsymbol{W})\boldsymbol{\beta} \ , \end{aligned}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can be computed with the Cholesky decomposition $\boldsymbol{K} = \boldsymbol{R}^\mathsf{T}\boldsymbol{R}$ via

$$\begin{aligned} \boldsymbol{\alpha} &= \boldsymbol{R}\backslash\Big(\boldsymbol{R}^\mathsf{T}\backslash(\boldsymbol{z} - m(\boldsymbol{W}))\Big) \\ \boldsymbol{\beta} &= \boldsymbol{R}\backslash\Big(\boldsymbol{R}^\mathsf{T}\backslash\varrho(\boldsymbol{W}, \boldsymbol{w})\Big) \ . \end{aligned} \tag{3.14}$$

The resulting computational cost of computing $\boldsymbol{R}$ is $\mathcal{O}(n^3/6)$ and the cost of computing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is $\mathcal{O}(n^2)$ [158].

If the training data set $\mathcal{D}_k$ does not change, the Cholesky decomposition $\boldsymbol{K} = \boldsymbol{R}^\mathsf{T}\boldsymbol{R}$ and the computation of $\boldsymbol{\alpha}$ have to be performed only once at the beginning, whereas $\boldsymbol{\beta}$ has to be recomputed for every new test point $\boldsymbol{w}$. If $\mathcal{D}_k$ changes, i.e., with each inclusion or removal of a data point, the covariance matrix $\boldsymbol{K}$ has to be updated for an appropriate evaluation of the GP posterior. If a data point is included, a row and column have to be added to $\boldsymbol{K}$. If a data point is removed, the respective row and column associated with this point have to be removed. These changes require, in principle, a full recalculation of the Cholesky factor $\boldsymbol{R}$, which is the most expensive computation. To reduce this computational load we employ the approach of Osborne in [145] to recalculate the Cholesky factor recursively, taking advantage of the available factor of the previous step. The precise procedure is outlined in Sec. A.3 of the appendix.

Remark 14. *The recursive update of the Cholesky factor can only be applied if the hyperparameters $\boldsymbol{\theta}$ do not change because otherwise, every single element of $\boldsymbol{K}$ changes and a recursive approach is not applicable anymore.*

Remark 15. *Note that sometimes [79, 152, 186] not the Cholesky decomposition but the covariance matrix inverse $\boldsymbol{K}^{-1}$ is recursively computed, which is based on the partitioned block inverse using the* Woodbury matrix identity*. Presumably for the numerical issues outlined above, this approach has never been used in combination with model predictive control. It has, however, been used in the signal processing literature, where it is strongly connected to the concept of* kernel recursive least-squares *[152, 186].*

Due to the recursive nature, both in the data inclusion approach and the Cholesky decomposition, we denote the resulting Gaussian process as recursive GP (rGP). The most important steps of the rGP are presented as part of Algorithm 1 in Section 3.5.

3.5 rGP-MPC Formulation

In this section, we summarize the output feedback model predictive control formulation, based on the rGP-NARX model for prediction.

3.5.1 GP-NARX Prediction Model

In Section 3.6, we establish input-to-state stability in terms of the evolution of the state $\boldsymbol{x}_k$ and not the output y_k. For this reason, we first reformulate the GP output prediction in terms of the predicted NARX state $\hat{\boldsymbol{x}}_k$. We start by setting $k := k+1$ in $\hat{\boldsymbol{x}}_k$ and arrive at

$$\hat{\boldsymbol{x}}_{k+1} = \left[\hat{y}_{k+1}, y_k, \ldots, y_{k+1-m_y}, u_k, \ldots, u_{k+1-m_u}\right] ,$$

where only $\hat{y}_{k+1}$ is an estimated quantity as all the other elements are or have been measured/computed. Since the predicted output $\hat{y}_{k+1}$ is computed by (3.12) we obtain the NARX prediction model

$$\hat{\boldsymbol{x}}_{k+1} = \hat{F}(\hat{\boldsymbol{x}}_k, u_k | \mathcal{D}_k) := \left[m_+(\boldsymbol{w}_k|\mathcal{D}_k), y_k, \ldots, y_{k+1-m_y}, u_k, \ldots, u_{k+1-m_u}\right] , \tag{3.15}$$

which we also denote as the *nominal* model.
Correspondingly, for the NARX model of the real process (3.9a) we have

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \left[y_{k+1}, y_k, \ldots, y_{k+1-m_y}, u_k, \ldots, u_{k+1-m_u}\right] \\ &= \left[f(\boldsymbol{x}_k, u_k) + \epsilon, y_k, \ldots, y_{k+1-m_y}, u_k, \ldots, u_{k+1-m_u}\right]\end{aligned}$$

and due to (3.13) this can be reformulated as

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \left[m_+(\boldsymbol{w}_k|\mathcal{D}_k) + e^{\mathrm{p}}, y_k, \ldots, y_{k+1-m_y}, u_k, \ldots, u_{k+1-m_u}\right] \\ &= \hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k) + \boldsymbol{d}e^{\mathrm{p}} =: F(\boldsymbol{x}_k, u_k, e^{\mathrm{p}})\end{aligned} \tag{3.16}$$

with $\boldsymbol{d} = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}^{\mathsf{T}}$, i.e., the real NARX model can be represented as the superposition of the nominal/prediction model and the prediction error.

3.5.2 Resulting Optimal Control Problem

In order to steer system (3.9) to and stabilize at the target reference point $(\boldsymbol{x}_{\mathrm{ref}}, u_{\mathrm{ref}})$, as detailed in the problem formulation in Section 3.3, we use the prediction model (3.15) at each time step k in the optimization problem

$$\begin{aligned}\underset{\hat{\mathbf{u}}_{k|k}}{\text{minimize}} \quad & V_N\left(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}\right) \\ \text{subject to} \quad & \forall i \in \mathcal{I}_{0:N-1} : \\ & \hat{\boldsymbol{x}}_{k+i+1|k} = \hat{F}\left(\hat{\boldsymbol{x}}_{k+i|k}, \hat{u}_{k+i|k}|\mathcal{D}_k\right) \\ & \hat{\boldsymbol{x}}_{k|k} = \boldsymbol{x}_k \\ & \hat{u}_{k+i|k} \in \mathcal{U} \\ & \hat{\boldsymbol{x}}_{k+i|k} \in \mathcal{X} \ .\end{aligned} \tag{3.17}$$

Note that we do not employ a terminal region (see (3.7)). Here, the constraint set $\mathcal{X}$ is a combination of multiple instances of $\mathcal{Y}_\mathrm{h}$, depending on the specific composition of $\boldsymbol{x}_k$.[11] If only soft output constraints $\mathcal{Y}_\mathrm{s}$ are considered, the hard constraints $\mathcal{X}$ can be removed.

We consider the cost function

$$V_N\big(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}\big) = \sum_{i=0}^{N-1} \ell\big(\hat{\boldsymbol{x}}_{k+i|k}, \hat{u}_{k+i|k}\big) + \lambda V_\mathrm{f}\big(\hat{\boldsymbol{x}}_{k+N|k} - \boldsymbol{x}_\mathrm{ref}\big) \ ,$$

where $V_\mathrm{f}(\cdot)$ is the terminal cost function, which is weighted by $\lambda \geq 1$. The employed positive stage cost is given by

$$\ell(\hat{\boldsymbol{x}}_k, \hat{u}_k) = \ell_\mathrm{s}(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_\mathrm{ref}, \hat{u}_k - u_\mathrm{ref}) + \ell_\mathrm{b}(\hat{y}_k) \ ,$$

where $\ell_\mathrm{s}(\cdot)$ penalizes input and state deviations from the reference point $(\boldsymbol{x}_\mathrm{ref}, u_\mathrm{ref})$ and $\ell_\mathrm{b}(\cdot)$ is a barrier function that can account for soft output constraints $\mathcal{Y}_\mathrm{s}$. The barrier function is defined by

$$\ell_\mathrm{b}(\hat{y}_k) \geq \alpha_\mathrm{b}\big(d(\hat{y}_k, \mathcal{Y}_\mathrm{s})\big)$$

and must satisfy $\ell_\mathrm{b}(\hat{y}_k) = 0, \forall \hat{y}_k \in \mathcal{Y}_\mathrm{s}$[12], where $\alpha_\mathrm{b}(\cdot)$ is a $\mathcal{K}$-function and $d(\cdot)$ the distance function as defined in Appendix A.1.

Remark 16 (Barrier functions)**.** *The use of barrier functions is a trade-off between constraint satisfaction and computation time. In case of hard constraints, the barrier function $\ell_\mathrm{b}(\cdot)$ is omitted (or set to zero) and the solution of the optimal control problem is computationally expensive. If, however, satisfaction of the constraints $\mathcal{X}$ is not required 100 % of the time (soft constrained case), one can remove the constraints $\mathcal{X}$ from* (3.17)*, which makes the solution of the optimal control problem much faster. In that case a barrier function can be employed, which establishes/recovers constraint satisfaction for most of the operating time, though not guaranteed as with hard constraints.*

The feasible set of (3.17) is denoted by $\mathcal{X}_N$, the optimal solution by $\hat{\mathbf{u}}^*_{k|k}$, and the resulting optimal state sequence by $\hat{\mathbf{x}}^*_{k|k}$. The first element of $\hat{\mathbf{u}}^*_{k|k}$, i.e., $\hat{u}^*_{k|k}$, is applied to the process such that we obtain $u_k = \kappa_\mathrm{MPC}(\boldsymbol{x}_k|\mathcal{D}_k) = \hat{u}^*_{k|k}$. Note that here the implicitly defined control law $\kappa_\mathrm{MPC}(\boldsymbol{x}_k|\mathcal{D}_k)$ is time-varying, as well as the value function $V^*_N(\boldsymbol{x}_k|\mathcal{D}_k) = V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}^*_{k|k}|\mathcal{D}_k)$ because they depend on the changing prediction model associated with $\mathcal{D}_k$.

The resulting rGP-MPC formulation of (3.17) is given in Algorithm 1.

Remark 17. *The control objective is to steer the state to $\boldsymbol{x}_\mathrm{ref}$ and stabilize it there. This is a forced set-point, associated with the nonzero input u_ref. To achieve the con-*

[11] If for instance $\boldsymbol{x}_k = [y_k, y_{k-1}, y_{k-2}]$, then $\mathcal{X} = \mathcal{Y}_\mathrm{h} \times \mathcal{Y}_\mathrm{h} \times \mathcal{Y}_\mathrm{h}$.

[12] This way, the barrier function does not influence the construction of the terminal cost.

trol objective, the cost function is designed such that it penalizes deviations from this known reference point $(\boldsymbol{x}_{\mathrm{ref}}, u_{\mathrm{ref}})$*. If* u_{ref} *is unknown beforehand, one option is to reformulate* (3.17) *such that a sequence of input changes* $\{\Delta\hat{u}_{k|k}, \Delta\hat{u}_{k+1|k}, \ldots, \Delta\hat{u}_{k+N-1|k}\}$ *is computed. This formulation also allows attaining zero control error [117].*

3.6 Stability of rGP-MPC

We establish stability of the rGP-MPC scheme. In particular, we adapt and extend the presented nominal and robust stability results of Section 3.2.3 to the rGP-MPC case. The biggest change originates from the changing Gaussian process prediction model $\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k)$ due to the online learning scheme. A further but smaller change originates from the fact that we do not consider stabilization of the origin but an arbitrary forced reference point $(\boldsymbol{x}_{\mathrm{ref}}, u_{\mathrm{ref}})$.

We start by reformulating the input-to-state stability condition in terms of the nonzero reference point $\boldsymbol{x}_{\mathrm{ref}}$.

Definition 6 (Input-to-state Stability)**.**
The closed-loop system $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k), e^{\mathrm{p}})$ *is input-to-state stable (ISS) with respect to the the set-point* $\boldsymbol{x}_{\mathrm{ref}}$ *if there exist a* $\mathcal{KL}$*-function* $\beta(\cdot,\cdot)$ *and a* $\mathcal{K}$*-function* $\gamma(\cdot)$ *such that*

$$\|\boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{ref}}\| \leq \beta(\|\boldsymbol{x}_0 - \boldsymbol{x}_{\mathrm{ref}}\|, k) + \gamma\Big(\max_{k\geq 0}|e^{\mathrm{p}}|\Big)$$

holds for all initial states $\boldsymbol{x}_0$*, prediction errors* e^{p}*, and for all* k*.*

Note that input-to-state stability combines nominal stability as well as uniformly bounded influence of uncertainty in a single condition. Therefore, we consider stability first for the nominal case in Section 3.6.1, i.e., when the prediction/nominal model (3.15) and the true system (3.16) are exactly the same. Afterwards, we establish robust stability in the sense of input-to-state stability.

3.6.1 Nominal Stability of rGP-MPC

In the following, let the current deviation from the reference point and the deviation at the next time step be $\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_k - \boldsymbol{x}_{\mathrm{ref}}$ and $\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_{\mathrm{ref}}$ respectively.

Assumption 10. *Assume that*

1. *the stage cost function* $\ell(\boldsymbol{x}, u)$ *is positive definite, i.e.,* $\ell(\boldsymbol{x}_{\mathrm{ref}}, u_{\mathrm{ref}}) = 0$ *and there exists a* $\mathcal{K}_\infty$*-function* $\alpha(\cdot)$ *such that* $\ell(\boldsymbol{x}, u) \geq \alpha(\|\tilde{\boldsymbol{x}}\|)$ *for all* $u \in \mathcal{U}$*,*

2. *let* d *be a positive constant such that* $\ell(\boldsymbol{x}, u) > d$ *for all* $\boldsymbol{x} \notin \mathcal{X}_{\mathrm{f}}$ *and all* $u \in \mathcal{U}$*, and*

Algorithm 1 Recursive Guassian Process Model Predictive Control

MPC Parameters: Prediction horizon N, stage cost $\ell(\cdot)$ with respective parameters, hard input constraint set $\mathcal{U}$, output constraint set $\mathcal{Y}$.
rGP Parameters: Prior mean $m(\cdot)$, covariance function $\varrho(\cdot,\cdot)$, initial hyperparameters $\boldsymbol{\theta}$, thresholds $\bar{e}^{\mathrm{p}}$ and $\bar{\sigma}^2$, maximum number of training points $\bar{n}$.

Initialization
Training data set $\mathcal{D}$.
Optimize hyperparameters $\boldsymbol{\theta}$ (2.8) with initial data set $\mathcal{D}$.
Initialize GP posterior mean function $m_+(\boldsymbol{w})$ with covariance matrix $\boldsymbol{K}$, Cholesky factor $\boldsymbol{R}$, and $\boldsymbol{\alpha}$ (Section 3.4.2).
Compute GP posterior mean gradient $\nabla m_+(\boldsymbol{w})$ (Section 3.7.1).
Compute linear GP model at $\boldsymbol{x}_{\mathrm{ref}}$ (Section 3.7).
Compute terminal cost function $V_{\mathrm{f}}(\cdot)$ (Section 3.7).

Recursion
for each time step k **do**
Solve optimal control problem (3.17) for initial condition $\boldsymbol{x}_k$ and obtain optimal input sequence $\hat{\mathbf{u}}^*_{k|k}$.
Apply first element $u_k = \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k) = \hat{u}^*_{k|k}$.
Obtain new output y_{k+1}.
Construct new GP data point $(\boldsymbol{w}_k, y_{k+1})$ with $\boldsymbol{w}_k = (\boldsymbol{x}_k, u_k)$.
Update GP:
Compute $\hat{y}_{k+1} = m_+(\boldsymbol{w}_k|\mathcal{D}_k)$ and $\sigma^2_+ = \sigma^2_+(\boldsymbol{w}_k|\mathcal{D}_k)$.
if $|y_{k+1} - \hat{y}_{k+1}| > \bar{e}^{\mathrm{p}}$ OR $\sigma^2_+ > \bar{\sigma}^2$ **then**
$\mathcal{D}'_{k+1} = \mathcal{D}_k \cup (\boldsymbol{w}_k, y_{k+1})$.
Using $\mathcal{D}'_{k+1}$, compute $\boldsymbol{K}$' and $\boldsymbol{R}$' via (A.2).
if number of training points $> M$ **then**
Remove oldest data point and downdate $\boldsymbol{K}$' and $\boldsymbol{R}$' via (A.3).
end if
Compute $\boldsymbol{\alpha}$' via (3.14).
if $V^*_N\big(\boldsymbol{x}_k|\mathcal{D}'_{k+1}\big) \leq V^*_N\big(\boldsymbol{x}_k|\mathcal{D}_k\big)$ **then**
$\mathcal{D}_{k+1} = \mathcal{D}'_{k+1}$
Make $\boldsymbol{K}$', $\boldsymbol{R}$', and $\boldsymbol{\alpha}$' effective.
else
$\mathcal{D}_{k+1} = \mathcal{D}_k$
Reverse $\boldsymbol{K}$', $\boldsymbol{R}$', and $\boldsymbol{\alpha}$'.
end if
else
$\mathcal{D}_{k+1} = \mathcal{D}_k$
end if
end for

3. *there exists a terminal control law* $\kappa_\mathrm{f}(\tilde{\boldsymbol{x}}_k)$ *and a control Lyapunov function* $V_\mathrm{f}(\tilde{\boldsymbol{x}}_k)$ *such that the conditions*

$$\alpha_1(\|\tilde{\boldsymbol{x}}_k\|) \leq V_\mathrm{f}(\tilde{\boldsymbol{x}}_k) \leq \alpha_2(\|\tilde{\boldsymbol{x}}_k\|)$$

and

$$V_\mathrm{f}(\tilde{\boldsymbol{x}}_{k+1}) \leq V_\mathrm{f}(\tilde{\boldsymbol{x}}_k) - \ell(\tilde{\boldsymbol{x}}_k + \boldsymbol{x}_\mathrm{ref}, \kappa_\mathrm{f}(\tilde{\boldsymbol{x}}_k) + u_\mathrm{ref})$$

hold for all $\tilde{\boldsymbol{x}}_k \in \mathcal{X}_\mathrm{f} = \{\tilde{\boldsymbol{x}}_k \in \mathbb{R}^{n_x} : V_\mathrm{f}(\tilde{\boldsymbol{x}}_k) \leq \nu\} \subseteq \mathcal{X}$ *with* $\nu > 0$ *and* $\tilde{\boldsymbol{x}}_{k+1} = \hat{F}(\tilde{\boldsymbol{x}}_k + \boldsymbol{x}_\mathrm{ref}, \kappa_\mathrm{f}(\tilde{\boldsymbol{x}}_k) + u_\mathrm{ref} \,|\, \mathcal{D}) - \boldsymbol{x}_\mathrm{ref}$*, and where* $\alpha_1(\cdot)$ *and* $\alpha_2(\cdot)$ *are* $\mathcal{K}_\infty$*-functions. The constant* ν *is chosen such that* $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$ *and* $\kappa_\mathrm{f}(\tilde{\boldsymbol{x}}_k) + u_\mathrm{ref} \in \mathcal{U}$ *for all* $\tilde{\boldsymbol{x}}_k \in \mathcal{X}_\mathrm{f}$*.*

Assumption 10 is a reformulation of Assumptions 4 to 8 to the present case of the rGP-MPC scheme and regulation to the reference point $(\boldsymbol{x}_\mathrm{ref}, u_\mathrm{ref})$. It ensures that the system is locally asymptotically stable on the positive invariant set $\mathcal{X}_\mathrm{f}$, while satisfying state and input constraints. It can be satisfied if we have a, at least, locally valid description of the process at the target point. This can, for instance, be a linearized version of the GP prediction model at the reference (with e.g. $\mathcal{D} = \mathcal{D}_\mathrm{ref}$), which in turn can then be used to derive a suitable terminal cost and controller. Possible options are then, for instance, the use of a linear-quadratic regulator and/or applying Lyapunov methods (see also Section 3.7).

Remark 18. *Although it is sufficient to determine the terminal components from the nominal model (see Section 3.7), one could also consider the design of a robust terminal controller and cost. For instance, using a Gaussian process model for the target region one could consider a specific probability bound given by the posterior variance and then, based on this, design a robust terminal controller.*

We now state the nominal stability result for rGP-MPC. It is an extension of Theorem 2 (which considers a MPC formulation without terminal region) that also accounts for the changing GP prediction model.

Theorem 4 (rGP-MPC nominal stability)**.** *Let* $\kappa_\mathrm{MPC}(\boldsymbol{x}_k|\mathcal{D}_k)$ *be the predictive control law derived from the optimal control problem* (3.17) *and let Assumption 10 hold. Furthermore, let* $\mathcal{D}_k$ *be the training data set at time* k*,* $\mathcal{D}_{k+1}$ *the data set that will be used at time* $k+1$*, and* $\mathcal{D}'_{k+1}$ *the updated training data set candidate resulting from Rule 1. If* $\mathcal{D}_k$ *is updated using the rule*

if $V_N^*(\boldsymbol{x}_k|\mathcal{D}'_{k+1}) \leq V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ ***then***
$\mathcal{D}_{k+1} \leftarrow \mathcal{D}'_{k+1}$
else
$\mathcal{D}_{k+1} \leftarrow \mathcal{D}_k$
end if

then $\forall \lambda \geq 1$*, there exists a region of attraction* $\Upsilon_0(\lambda) \subset \mathcal{X}_N$ *such that* $\forall \boldsymbol{x}_0 \in \Upsilon_0(\lambda)$ *the target* $\boldsymbol{x}_{\text{ref}}$ *of the nominal closed-loop system* $\boldsymbol{x}_{k+1} = \hat{F}(\boldsymbol{x}_k, \kappa_{\text{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k))$ *is asymptotically stable. The size of the set* $\Upsilon_0(\lambda)$ *increases with* λ*.*

Proof. We first prove that the value function $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ is a Lyapunov function and afterwards specify the region of attraction $\Upsilon_0(\lambda)$.

1. Let $\hat{\mathbf{x}}_{k|k}^* = \{\hat{\boldsymbol{x}}_{k|k}^*, \hat{\boldsymbol{x}}_{k+1|k}^*, \ldots, \hat{\boldsymbol{x}}_{k+N|k}^*\}$ be the predicted state sequence that results from applying the optimal input sequence $\hat{\mathbf{u}}_{k|k}^*$. Then we can write the optimal cost for initial condition $\boldsymbol{x}_k = \hat{\boldsymbol{x}}_{k|k}^*$ also as $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) = V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}^*|\mathcal{D}_k) = V_N(\hat{\mathbf{x}}_{k|k}^*, \hat{\mathbf{u}}_{k|k}^*|\mathcal{D}_k)$. Let furthermore be $\hat{\mathbf{u}}_{k+1|k}^* = \{\hat{u}_{k+1|k}^*, \ldots, \hat{u}_{k+N-1|k}^*, \kappa_{\text{f}}(\hat{\boldsymbol{x}}_{k+N|k}^* - \boldsymbol{x}_{\text{ref}}) + u_{\text{ref}}\}$ and $\hat{\mathbf{x}}_{k+1|k}^* = \{\hat{\boldsymbol{x}}_{k+1|k}^*, \ldots, \hat{\boldsymbol{x}}_{k+N+1|k}^*\}$ the respective sequences that start at $k+1$ computed at time k, where the last state is given by the terminal control law, that is to say, $\hat{\boldsymbol{x}}_{k+N+1|k}^* = \hat{F}\big(\hat{\boldsymbol{x}}_{k+N|k}^*, \kappa_{\text{f}}(\hat{\boldsymbol{x}}_{k+N|k}^* - \boldsymbol{x}_{\text{ref}}) + u_{\text{ref}}|\mathcal{D}_k\big)$.

   By Assumption 10 we have that the stage and terminal cost are positive definite. Hence, the cost function $V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k})$ is positive definite. Furthermore we also obtain, given a fixed $\mathcal{D}_{k+1}$, the decreasing property

$$V_N\big(\hat{\mathbf{x}}_{k+1|k}^*, \hat{\mathbf{u}}_{k+1|k}^*|\mathcal{D}_{k+1}\big) \leq V_N\big(\hat{\mathbf{x}}_{k|k}^*, \hat{\mathbf{u}}_{k|k}^*|\mathcal{D}_{k+1}\big) - \ell(\boldsymbol{x}_k, u_k)$$

   by Assumption 10, which is a well known result in standard MPC (for the derivation see, for instance, [160] or [156]).

   Given the update rule in Theorem 4 we have

$$V_N\big(\hat{\mathbf{x}}_{k|k}^*, \hat{\mathbf{u}}_{k|k}^*|\mathcal{D}_{k+1}\big) - \ell(\boldsymbol{x}_k, u_k) \leq V_N\big(\hat{\mathbf{x}}_{k|k}^*, \hat{\mathbf{u}}_{k|k}^*|\mathcal{D}_k\big) - \ell(\boldsymbol{x}_k, u_k) \ .$$

   Combining the previous two equations we obtain

$$V_N\big(\hat{\mathbf{x}}_{k+1|k}^*, \hat{\mathbf{u}}_{k+1|k}^*|\mathcal{D}_{k+1}\big) \leq V_N\big(\hat{\mathbf{x}}_{k|k}^*, \hat{\mathbf{u}}_{k|k}^*|\mathcal{D}_k\big) - \ell(\boldsymbol{x}_k, u_k) \ ,$$

   which is the same as

$$\begin{aligned} & V_N^*\big(\hat{\boldsymbol{x}}_{k+1}|\mathcal{D}_{k+1}\big) \leq V_N^*\big(\hat{\boldsymbol{x}}_k|\mathcal{D}_k\big) - \ell(\boldsymbol{x}_k, u_k) \\ \Leftrightarrow \ & V_N^*\big(\boldsymbol{x}_{k+1}|\mathcal{D}_{k+1}\big) \leq V_N^*\big(\boldsymbol{x}_k|\mathcal{D}_k\big) - \ell(\boldsymbol{x}_k, u_k) \end{aligned} \tag{3.18}$$

   because predicted and non-predicted values are the same in the nominal case. Thus, the value function is decreasing even if the prediction model changes due to the change from $\mathcal{D}_k$ to $\mathcal{D}_{k+1}$.

2. Regarding the region of attraction, we use Theorem 2 from [108], and show its extension to the rGP-MPC case. In particular, Theorem 2 (Theorem 3 in [108]) shows for the nominal and time-invariant case of (3.17) (i.e. constant prediction model and no model-plant mismatch) with value function $V_N^*(\boldsymbol{x}_k)$ that $\forall \lambda \geq 1$ there exists a region of attraction $\Upsilon(\lambda) \subset \mathcal{X}_N$ such that $\forall \boldsymbol{x}_0 \in$

$\Upsilon(\lambda)$ the nominal closed-loop system $\boldsymbol{x}_{k+1} = \hat{F}(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k))$ is recursively feasible and asymptotically stable. The region of attraction is characterized by $\Upsilon(\lambda) = \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k) \leq N \cdot d + \lambda \cdot \nu\}$, where ν and d are defined in Assumption 10. The size of the set $\Upsilon(\lambda)$ increases with λ.[13]

In this work, the value function $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ changes at certain time instances k whenever the data set $\mathcal{D}_k$ changes. Thus, we extend the definition of the region of attraction to

$$\Upsilon_k(\lambda) := \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) \leq N \cdot d + \lambda \cdot \nu\} \ ,$$

which then also changes with k. Due to (3.18), the optimal cost is decreasing for a particular state sequence $\mathbf{x} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \ldots\}$ with

$$Nd + \lambda\nu \geq V_N^*(\boldsymbol{x}_0|\mathcal{D}_0) \geq V_N^*(\boldsymbol{x}_1|\mathcal{D}_1) \geq \ldots \geq V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$$

and therefore $\Upsilon_k(\lambda)$ is increasing along the state sequence. Thus, if the initial state $\boldsymbol{x}_0 \in \Upsilon_0(\lambda)$, then the subsequent states $\boldsymbol{x}_k \in \Upsilon_k(\lambda)$ and the optimal control problem is recursively feasible. Hence, the target $\boldsymbol{x}_{\mathrm{ref}}$ is asymptotically stable in $\Upsilon_0(\lambda)$ for the nominal closed-loop system $\boldsymbol{x}_{k+1} = \hat{F}(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k))$.

□

At $\boldsymbol{x}_k$ (with the current output measurement y_k) the optimal control problem (3.17) is solved with the data set $\mathcal{D}_k$ and the resulting input $u_k = \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k) = \hat{u}_{k|k}^*$ is applied to the system. If the next data point $(\boldsymbol{w}_k, y_{k+1})$ is a candidate for updating the GP, the previous optimal cost is recomputed using the updated GP. If the cost does not increase, the GP update becomes effective; otherwise it is discarded. Thus, the update rule in Theorem 4 is executed additionally after update Rule 1 presented in Section 3.4.1. This is also reflected in Algorithm 1.

Remark 19. *Assume that, for instance, the update rule of Theorem 4 is satisfied to update the training data set from $k = 1$ to $k = 2$, i.e., from $\mathcal{D}_1$ to $\mathcal{D}_2$ for a particular state $\boldsymbol{x}$. This however does not automatically translate to that $\mathcal{D}_1$ can be updated to $\mathcal{D}_2$ for a different state $\boldsymbol{x}'$. Put more generally, the update rule to update $\mathcal{D}_k$ to $\mathcal{D}_{k+1}$ might not be satisfied for all states $\boldsymbol{x}$. For this reason we cannot guarantee that $\Upsilon_k(\lambda)$ increases in general for all $\boldsymbol{x}$, i.e., $\Upsilon_0(\lambda) \subseteq \Upsilon_1(\lambda) \subseteq \Upsilon_2(\lambda) \subseteq \ldots$ cannot be guaranteed. We can only guarantee that $\Upsilon_k(\lambda)$ increases along a* specific *trajectory $\mathbf{x} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \ldots\}$ and for this trajectory we have $\boldsymbol{x}_k \in \Upsilon_k(\lambda)$ $\forall k$.*

Remark 20 (Conflicting objectives)**.** *Theorem 4 establishes nominal stability despite a changing training data set $\mathcal{D}_k$. To determine the new data set candidate $\mathcal{D}_{k+1}'$ we use*

[13] Note that Theorem 3 in [108] (Theorem 2 in this thesis) is stated the other way round, i.e., for each region $\Upsilon(\lambda) \subset \mathcal{X}_N$ and for all $\boldsymbol{x}_k \in \Upsilon(\lambda)$, there exists a $\lambda \geq 1$ such that the nominal closed-loop system is asymptotically stable at $\boldsymbol{x}_{\mathrm{ref}}$.

Rule 1, whose objective is to refine the current prediction model. Note that also other rules, which utilize different selection criteria for model refinement (e.g. statistical methods, see Section 3.4.1), can be employed. One could assume that the additional update rule in Theorem 4 is not necessary because with every new data point the prediction model should become more accurate. This is, however, not necessarily the case if, for instance, the output is corrupted by noise or if outliers are present. In both cases, the apparent process behavior differs from the true behavior and it cannot be guaranteed that the prediction model becomes more accurate with every added data point, nor that the value function continues decreasing monotonically. Thus, the objective of the update rule in Theorem 4 is to make sure that safety, in the sense of stability and constraint satisfaction, is guaranteed. This is also illustrated in the examples, see, e.g., Fig. 3.11. In the same simulations we also see that data points, selected by Rule 1 and which carry valuable information, are discarded by the update rule of Theorem 4 because the decreasing value function condition, and with that stability, could not be guaranteed. In other words, the two objectives of model refinement (expressed by Rule 1) and safety (in the sense of stability, expressed by the update rule of Theorem 4) are conflicting objectives, especially in the case of corrupted measurements. ***In this work we prioritize safety, thereby sacrificing a bit of the potential of model refinement.***

On the basis of the nominal stability result for the online rGP-MPC scheme, we now establish robust stability.

3.6.2 Robust Stability of rGP-MPC

We show that the reference $\boldsymbol{x}_{\text{ref}}$ of the real process (3.9), controlled by the proposed predictive control law $\kappa_{\text{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k)$, is input-to-state stable w.r.t. the prediction error e^{p}.

Assumption 11. *We assume that the Gaussian measurement noise ϵ of the real process* (3.9a) *has bounded support, i.e., $|\epsilon| \leq \bar{\epsilon} < \infty$.*

Remark 21. *Assumption 11 allows establishing a bounded prediction error e^{p} required for robust constraint satisfaction of Theorem 5. This restricts the considered system class* (3.9a) *in Section 3.3 to those with bounded measurement noise, which includes real processes because there, the measurement noise is always bounded (e.g. due to the limitations of the involved data acquisition systems).*

Remark 22. *The concept of Gaussian process regression assumes Gaussian noise in the measurements, i.e., noise with unbounded support. Thus, employing GP regression for systems with Gaussian measurement noise with bounded support can be questioned from a probabilistic point of view. One can regain Gaussian noise with unbounded support by means of* GP warping *[175]. The smaller the bounded support, the larger the*

difference between the distributions and the larger the correcting effect of warping. Note furthermore that any resulting approximation error can be absorbed into the prediction error e^{p}.

Theorem 5 (rGP-MPC input-to-state stability)**.** *Let $\kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k)$ be the predictive controller derived from optimal control problem* (3.17) *satisfying Assumption 10, Theorem 4, and Assumption 11. If*

- *the nominal model $\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k)$ is uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$, all $u_k \in \mathcal{U}$, and all $\mathcal{D}_k$ during the prediction horizon[14], and*
- *the stage cost function $\ell(\boldsymbol{x}_k, u_k)$ and the terminal cost function $V_{\mathrm{f}}(\boldsymbol{x}_k)$ are uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$ and all $u_k \in \mathcal{U}$,*

then the target $\boldsymbol{x}_{\mathrm{ref}}$ of the closed-loop system $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k), e^{\mathrm{p}})$ is ISS w.r.t. the prediction error e^{p} with $|e^{\mathrm{p}}| \leq \mu < \infty$ in a robust invariant set $\Omega_0^r(\lambda) \subset \mathcal{X}_N$ for a sufficiently small μ. The smaller μ, the larger the set $\Omega_0^r(\lambda)$.

Proof. We first establish the set $\Omega_0^r(\lambda)$ and prove recursive feasibility. Afterwards we prove ISS.

1. Regarding the nature of $\Omega_0^r(\lambda)$, first note that in Theorem 4 we have shown that $\Upsilon_0(\lambda)$ is a region of attraction for the nominal case with $e^{\mathrm{p}} \equiv 0$. This was an extension of the results presented in [108] to the case of a changing GP prediction model. Now, in a similar manner, we review a result of [107], where ISS for MPC was shown with a constant prediction model, and then extend it to our case of a changing prediction model. Note that [107] can itself be considered as an extension of [108] because the authors also proved the ISS property for an optimal control problem without terminal region.

   Proposition 1 (C2) in [107] shows for the time-invariant case of (3.17) (i.e. for an optimal control problem without terminal region and a constant prediction model) that the closed-loop $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k), e^{\mathrm{p}})$ is robustly feasible for all $\boldsymbol{x}_k$ in a robust invariant set $\Omega^r(\lambda)$. In particular, it is proven that if $|e^{\mathrm{p}}| \leq \mu$ with a sufficiently small μ, there exists a $r > 0$ such that $\Omega^r(\lambda) := \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k) \leq r\} \subset \mathcal{X}_N$ is a compact and robust invariant set such that for all $\boldsymbol{x}_k \in \Omega^r(\lambda)$ the resulting predicted state sequence remains in $\Omega^r(\lambda)$. Therefore the state constraints $\mathcal{X}$ do not become active. Hence, for all $\boldsymbol{x}_0 \in \Omega^r(\lambda)$ the MPC scheme is recursively feasible and the constraints are robustly satisfied. Furthermore, larger values of λ lead to a larger region $\Omega^r(\lambda)$.

   Regarding rGP-MPC, first note that in Proposition 1 (C2) of [107] the following is not shown: Since an optimal control problem without terminal region is considered, according to the results in [108] (reviewed in this thesis as Theorem 2

[14] Note that this condition does not prohibit the change of the nominal model from the current time instant k to the next $k+1$.

and in Theorem 4), the region of attraction for the nominal case can be characterized by $\Upsilon(\lambda) = \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k) \leq N \cdot d + \lambda \cdot \nu\}$. Thus, in the robust case we need $\Omega^r(\lambda) \subset \Upsilon(\lambda) \subset \mathcal{X}_N$. For this to hold we require $0 < r < N \cdot d + \lambda \cdot \nu$, which establishes an upper bound on r.

Second, in the definition of $\Omega^r(\lambda)$ in [107] the value function $V_N^*(\boldsymbol{x}_k)$ is time-invariant, whereas in this work $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ depends on the changing data set $\mathcal{D}_k$. For this reason, we extend the definition of the robust invariant set to

$$\Omega_k^r(\lambda) := \{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) \leq r\} \subset \Upsilon_k(\lambda) \ ,$$

which then also changes with k. In order for $\Omega_k^r(\lambda) \subset \Upsilon_k(\lambda)$ to hold we require, as shown above, $0 < r < N \cdot d + \lambda \cdot \nu$. Like the region of attraction $\Upsilon_k(\lambda)$ of the nominal case (see the proof to Theorem 4), also $\Omega_k^r(\lambda)$ increases with λ and, in particular, with k along a specific state sequence $\mathbf{x} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots\}$ because $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ decreases due to the update rule in Theorem 4. Therefore, if the initial state $\boldsymbol{x}_0 \in \Omega_0^r(\lambda)$, then the subsequent states $\boldsymbol{x}_k \in \Omega_k^r(\lambda)$ and the state constraints do not become active. The existence of $\Omega_0^r(\lambda)$ is thereby established by Proposition 1 (C2) in [107] (as outlined above) and therefore, if $\boldsymbol{x}_0 \in \Omega_0^r(\lambda)$ then (3.17) is recursively feasible and the constraints are robustly satisfied.

2. Now we show that the closed-loop system $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \kappa_{\text{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k), e^{\text{p}})$ is input-to-state stable w.r.t. the prediction error e^{p}. To this end, we start by showing that the cost function $V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k})$ is uniformly continuous in $\boldsymbol{x}_k$. Since the nominal model $\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k)$ is uniformly continuous in $\boldsymbol{x}_k$ during the prediction horizon and due to Lemma 1 (Appendix A.1), there exists a $\mathcal{K}$-function $\sigma_x(\cdot)$ such that $\left\|\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k) - \hat{F}(\boldsymbol{z}_k, u_k|\mathcal{D}_k)\right\| \leq \sigma_x(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|)$ for all $\boldsymbol{x}_k, \boldsymbol{z}_k \in \mathcal{X}_N$, all $u_k \in \mathcal{U}$, and for a given data set $\mathcal{D}_k$. In accordance with Lemma 2 (Appendix A.1), the predicted state evolution then satisfies $\left\|\hat{\boldsymbol{x}}_{k+i|k} - \hat{\boldsymbol{z}}_{k+i|k}\right\| \leq \sigma_x^i(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|)$ for $i \in \mathcal{I}_{0:N-1}$. Furthermore, since the stage and terminal cost are uniformly continuous in $\boldsymbol{x}_k$, there exists a couple of $\mathcal{K}$-functions $\sigma_\ell(\cdot), \sigma_{V_{\text{f}}}(\cdot)$ such that $\|\ell(\boldsymbol{x}_k, u_k) - \ell(\boldsymbol{z}_k, u_k)\| \leq \sigma_\ell(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|)$ and $\|V_{\text{f}}(\boldsymbol{x}_k) - V_{\text{f}}(\boldsymbol{z}_k)\| \leq \sigma_{V_{\text{f}}}(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|)$ for all $\boldsymbol{x}_k, \boldsymbol{z}_k \in \mathcal{X}_N$ and all $u \in \mathcal{U}$. Combining these properties we obtain

$$\begin{aligned}
\left\|V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}) - V_N(\boldsymbol{z}_k, \hat{\mathbf{u}}_{k|k})\right\| &\leq \sum_{i=0}^{N-1} \left\|\ell(\hat{\boldsymbol{x}}_{k+i|k}, \hat{u}_{k+i|k}) - \ell(\hat{\boldsymbol{z}}_{k+i|k}, \hat{u}_{k+i|k})\right\| \\
&\quad + \left\|V_{\text{f}}(\hat{\boldsymbol{x}}_{k+N|k}) - V_{\text{f}}(\hat{\boldsymbol{z}}_{k+N|k})\right\| \\
&\leq \sum_{i=0}^{N-1} \sigma_\ell \circ \sigma_x^i(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|) + \sigma_{V_{\text{f}}} \circ \sigma_x^N(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|) \\
&=: \sigma_V(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|) \ ,
\end{aligned}$$

where $\circ$ denotes the concatenation of functions (e.g. $\sigma_1 \circ \sigma_2(x) = \sigma_1(\sigma_2(x))$) and $\sigma_V(\cdot)$ is a $\mathcal{K}$-function. Therefore the cost function is uniformly continuous in $\boldsymbol{x}_k$

for all $\boldsymbol{x}_k \in \mathcal{X}_N$ and all $\hat{\mathbf{u}}_{k|k}$.

As shown, for every $\boldsymbol{x}_k \in \Omega_0^r(\lambda)$ the state constraints do not become active. Thus, the optimal solution $\hat{\mathbf{u}}_{k|k}^*$ of (3.17) is feasible for every $\boldsymbol{x}_0 \in \Omega_0^r(\lambda)$ and we obtain

$$\|V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) - V_N^*(\boldsymbol{z}_k|\mathcal{D}_k)\| = \left\|V_N(\boldsymbol{x}_k, \hat{\mathbf{u}}_{k|k}^*) - V_N(\boldsymbol{z}_k, \hat{\mathbf{u}}_{k|k}^*)\right\| \leq \sigma_V(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|) \ .$$

Therefore, the value function $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ is also uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \Omega_0^r(\lambda)$ and a given data set $\mathcal{D}_k$.

Last we show that the value function is a ISS-Lyapunov function. Since $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ is a Lyapunov function for the nominal system (Theorem 4) there exists $\mathcal{K}_\infty$-functions $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot)$, such that

$$\begin{aligned} \alpha_1(\|\hat{\boldsymbol{x}}_k\|) \leq V_N^*(\hat{\boldsymbol{x}}_k|\mathcal{D}_k) \leq \alpha_2(\|\hat{\boldsymbol{x}}_k\|) \\ V_N^*(\hat{\boldsymbol{x}}_{k+1}|\mathcal{D}_{k+1}) - V_N^*(\hat{\boldsymbol{x}}_k|\mathcal{D}_k) \leq -\alpha_3(\|\hat{\boldsymbol{x}}_k\|) \ . \end{aligned}$$

Moreover, from (3.16) we have that the true process $F(\boldsymbol{x}_k, u_k, e^{\mathrm{p}})$ is affine in e^{p} and can be written as $F(\boldsymbol{x}_k, u_k, e^{\mathrm{p}}) = \hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k) + \boldsymbol{d}e^{\mathrm{p}}$. Then $F(\boldsymbol{x}_k, u_k, e^{\mathrm{p}})$ is always uniformly continuous in e^{p} because

$$\begin{aligned} &\|F(\boldsymbol{x}_k, u_k, e_1) - F(\boldsymbol{x}_k, u_k, e_2)\| \\ &= \left\|\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k) + \boldsymbol{d}e_1 - \left(\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k) + \boldsymbol{d}e_2\right)\right\| \\ &= \|\boldsymbol{d}(e_1 - e_2)\| \end{aligned}$$

is Lipschitz continuous with Lipschitz constant $L = 1$ and therefore also uniformly continuous. Then, there exists a $\mathcal{K}$-function $\sigma_e(\cdot)$ such that $\|F(\boldsymbol{x}_k, u_k, e_1) - F(\boldsymbol{x}_k, u_k, e_2)\| \leq \sigma_e(|e_1 - e_2|)$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$, all $u_k \in \mathcal{U}$, and all $|e_1|, |e_2| \leq \mu$. From these facts, it can be inferred that

$$\begin{aligned} &V_N^*(\boldsymbol{x}_{k+1}|\mathcal{D}_{k+1}) - V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) \\ &\quad = V_N^*\left(F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), e^{\mathrm{p}})|\mathcal{D}_{k+1}\right) - V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) \\ &\quad = V_N^*\left(F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), e^{\mathrm{p}})|\mathcal{D}_{k+1}\right) - V_N^*\left(F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), 0)|\mathcal{D}_{k+1}\right) \\ &\qquad + V_N^*\left(F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), 0)|\mathcal{D}_{k+1}\right) - V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) \\ &\quad \leq \left\|V_N^*\left(F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), e^{\mathrm{p}})|\mathcal{D}_{k+1}\right) - V_N^*\left(F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), 0)|\mathcal{D}_{k+1}\right)\right\| \\ &\qquad + V_N^*\left(\hat{\boldsymbol{x}}_{k+1}|\mathcal{D}_{k+1}\right) - V_N^*(\boldsymbol{x}_k|\mathcal{D}_k) \\ &\quad \leq \sigma_V\left(\|F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), e^{\mathrm{p}}) - F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k), 0)\|\right) - \alpha_3(\|\boldsymbol{x}_k\|) \\ &\quad \leq \sigma_V \circ \sigma_e(|e^{\mathrm{p}}|) - \alpha_3(\|\boldsymbol{x}_k\|) \ . \end{aligned}$$

Therefore, according to Definition 17 in Appendix A.2.2 the value function $V_N^*(\boldsymbol{x}_k|\mathcal{D}_k)$ is a ISS-Lyapunov function and with that the closed-loop system

$\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k, \kappa_{\mathrm{MPC}}(\boldsymbol{x}_k|\mathcal{D}_k), e^{\mathrm{p}})$ is ISS w.r.t. e^{p} for all $\boldsymbol{x}_0 \in \Omega_0^r(\lambda)$.

□

Remark 23 (Differences in soft and hard output constraints)**.** *In the case of soft constraints $\mathcal{Y}_{\mathrm{s}}$, the proposed controller ensures robust stability and constraint satisfaction for all initial states that lie in the feasible region $\Upsilon_0(\lambda)$ of the optimal control problem. In the case of hard constraints $\mathcal{Y}_{\mathrm{h}}$, the proposed controller ensures robust stability and constraint satisfaction for all initial states in a robust invariant set $\Omega_0^r(\lambda) \subset \Upsilon_0(\lambda)$ where the constraints are not active. Thus, from a practical point of view, if in the soft constraints case the initial state $\boldsymbol{x}_k$ leads to a feasible solution, we have $\boldsymbol{x}_k \in \Upsilon_0(\lambda)$ and the above guarantees hold. If in the hard constraints case the initial state $\boldsymbol{x}_k$ leads to a feasible solution, then we also have $\boldsymbol{x}_k \in \Upsilon_0(\lambda)$. However, in that case, one cannot be sure if $\boldsymbol{x}_k \in \Omega_0^r(\lambda)$ is satisfied. If $\boldsymbol{x}_k \notin \Omega_0^r(\lambda)$, then feasibility might be lost at one point. Thus, for safety critical applications the set $\Omega_0^r(\lambda)$ would required to be known in order to check $\boldsymbol{x}_k \in \Omega_0^r(\lambda)$, which is challenging because $\Omega_0^r(\lambda)$ (as well as $\Upsilon_0(\lambda)$) can in general not be computed but has to be estimated via simulations [160]. This issue could be circumvented if the hard constraints were tightened [107], thereby enlarging $\Omega_0^r(\lambda)$.*

Remark 24. *Notice that the ISS property is based on the uniform continuity of the optimal cost function and this does not depend on the size of the error signal. Hence, even if $|e^{\mathrm{p}}|$ is larger than μ for a short period of time in which we assume that the feasibility of the optimal control problem is not lost, i.e., $\boldsymbol{x}_k$ remains in $\Upsilon_k(\lambda)$ and ends in $\Omega_k^r(\lambda)$, then the closed-loop ISS property and constraint satisfaction will still hold.*

Remark 25 (Generalization)**.** *Theorems 4 and 5 are independent of the input dimension and do not require the considered NARX case. They also hold for state space systems where each state is modeled by an individual GP. Then, the process is also affine in the prediction error (a vector in that case). Thus, Theorems 4 and 5 also include the multi-input multi-output case. In addition, as long as the presented assumptions are satisfied, in particular, the update rule in Theorem 4, the stability results also hold for the case of online hyperparameter optimization and even further, for general prediction models $\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k)$ that are updated online, i.e., the stability guarantees are not confined to the use of Gaussian process prediction models.*

A necessary condition of Theorem 5 is that the nominal model $\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k)$ is uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}_N$, all $u_k \in \mathcal{U}$, and all $\mathcal{D}_k$ during the prediction horizon. In the case of Gaussian processes this can be guaranteed by the following proposition.

Proposition 3 (GP uniform continuity)**.** *The nominal model* (3.15) *is uniformly continuous in $\boldsymbol{x}_k$ if $\hat{f}(\boldsymbol{x}_k, u_k) = m_+(\boldsymbol{w}_k|\mathcal{D})$ with $\boldsymbol{w}_k = (\boldsymbol{x}_k, u_k)$ is uniformly continuous*

in $\boldsymbol{x}_k$. Since the prior mean $m(\boldsymbol{w}_k)$ is added to the posterior mean $m_+(\boldsymbol{w}_k|\mathcal{D})$, the prior mean has to be uniformly continuous in $\boldsymbol{x}_k$.[15] *One way to ensure that $m_+(\boldsymbol{w}_k|\mathcal{D})$ is uniformly continuous in $\boldsymbol{x}_k$, is to employ continuously differentiable kernels*[16] *(e.g. the squared exponential covariance function, the Matérn class covariance function with appropriate hyperparameters, or the rational quadratic covariance function). In that case, the process is mean square differentiable [1, 158], i.e., the posterior mean function is differentiable and therefore also uniformly continuous.*

Remark 26. *Although not required for Theorem 5, note that uniform continuity of the process $F(\boldsymbol{x}_k, u_k, e^{\mathrm{p}}) = \hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k) + \boldsymbol{d}e^{\mathrm{p}}$ in $\boldsymbol{x}_k$ is ensured if $\hat{F}(\boldsymbol{x}_k, u_k|\mathcal{D}_k)$ is uniformly continuous in $\boldsymbol{x}_k$, which can be established via Proposition 3.*

Remark 27. *In the majority of publications that utilize a Gaussian process prediction model within model predictive control, the employed kernel is the squared exponential covariance function. Thus, the Gaussian process prediction model is usually uniformly continuous. Furthermore, the uniform continuity assumption of the stage and terminal cost function is usually also satisfied. Then, all conditions of Theorem 5 are satisfied, which presents one possible explanation why the model predictive control schemes that employ Gaussian process prediction models usually seem to be robustly stable, though in many cases not specifically designed to be so.*

3.6.2.1 Resulting Prediction Errors

We finish this section with a discussion on the prediction error $e^{\mathrm{p}} = y_k - \hat{y}_k = f(\boldsymbol{x}_k, u_k) + \epsilon - m_+(\boldsymbol{w}_k|\mathcal{D}_k)$. According to Theorem 5, the smaller the error bound $|e^{\mathrm{p}}| \leq \mu$, the better because the positive invariant set $\Omega_0^r(\lambda)$ increases. Since the noise ϵ is in practice bounded by a finite $\bar{\epsilon}$ (Assumption 11), the error bound μ is finite if the true process $f(\boldsymbol{x}_k, u_k)$ and the GP posterior mean $m_+(\boldsymbol{w}_k|\mathcal{D}_k)$ are bounded. Boundedness of the true process is certainly a reasonable assumption in many practical relevant cases. Boundedness of the GP posterior mean, however, is more subtle. From a theoretical point of view, such a bound exists under certain conditions.

The posterior mean (with zero prior mean $m(\boldsymbol{w}) = 0$) can also be expressed via $m_+(\boldsymbol{w}^*|\mathcal{D}) = \sum_{i=1}^n \alpha_i k(\boldsymbol{w}_i, \boldsymbol{w}^*)$, with $\boldsymbol{w}_i \in \boldsymbol{W}$, as a linear combination of n kernel functions that determines a *reproducing kernel Hilbert space* (RKHS, [158]). As shown in [178], a bound in the RKHS exists if *universal* kernels are employed. One such kernel is, for instance, the squared exponential covariance function (2.7) for which the existence of a bound has been established in [150]. The authors of [40] presented a similar result when modeling the impulse response via a spline kernel. This result has also been used in [155]. Furthermore, the works [52, 177] provide ways to explicitly compute the bound, though only with high probability.

[15] The prior mean is usually specified by the user and often set to zero. Thus, uniform continuity of $m(\boldsymbol{w})$ is usually not an issue.

[16] Note that limiting the kernel to such functions might limit the expressiveness of the Gaussian process.

In practice, however, $m_+(\boldsymbol{w}^*|\mathcal{D})$ will generally be bounded assuming that the employed GP prior is well chosen and sufficiently informative training data $\mathcal{D}$ is used. Thus, the actual bound depends on the designer's choices regarding the specific employed GP model and the involved tuning parameters. Among these, in particular the thresholds for the prediction error and posterior variance for the presented rGP approach.

3.7 Terminal Components

The terminal cost function $V_\mathrm{f}(\cdot)$ of (3.17) is elementary for stability and can be determined in many ways, as long as Assumption 10 is satisfied. In the following, we outline a possible approach that is based on the analytic linearization of the Gaussian process prediction model.

3.7.1 GP Posterior Mean Gradient

In order to analytically linearize the Gaussian process prediction model, we require the gradient of the posterior mean function $m_+(\boldsymbol{w}|\mathcal{D})$ with respect to the regressor vector $\boldsymbol{w}$, which we derive in the following.

First note that the regressor training data set $\boldsymbol{W} = [\boldsymbol{w}^1 \cdots \boldsymbol{w}^n]^\mathsf{T}$ consists of n regressor vectors and one regressor vector $\boldsymbol{w} = [w_1 \cdots w_{n_w}]^\mathsf{T}$ consists of n_w scalar regressors. We define the posterior mean gradient with respect to $\boldsymbol{w}$ as

$$\nabla m_+(\boldsymbol{w}|\mathcal{D}) := \frac{\partial m_+(\boldsymbol{w}|\mathcal{D})}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial m_+(\boldsymbol{w}|\mathcal{D})}{\partial w_1} \\ \vdots \\ \frac{\partial m_+(\boldsymbol{w}|\mathcal{D})}{\partial w_{n_w}} \end{bmatrix} \in \mathbb{R}^{n_w} ,$$

which can be determined as follows. From (2.6a) we obtain

$$\begin{aligned} \nabla m_+(\boldsymbol{w}|\mathcal{D}) &= \frac{\partial m_+(\boldsymbol{w}|\mathcal{D})}{\partial \boldsymbol{w}} \\ &= \frac{\partial m(\boldsymbol{w})}{\partial \boldsymbol{w}} + \frac{\partial \varrho(\boldsymbol{w}, \boldsymbol{W})}{\partial \boldsymbol{w}} \boldsymbol{K}^{-1}(\boldsymbol{z} - m(\boldsymbol{W})) \end{aligned} \tag{3.19}$$

with

$$\frac{\partial m(\boldsymbol{w})}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial m(\boldsymbol{w})}{\partial w_1} & \cdots & \frac{\partial m(\boldsymbol{w})}{\partial w_{n_w}} \end{bmatrix}^\mathsf{T}$$

and

$$\frac{\partial \varrho(\boldsymbol{w}, \boldsymbol{W})}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} \begin{bmatrix} \varrho(\boldsymbol{w}, \boldsymbol{w}^1) & \cdots & \varrho(\boldsymbol{w}, \boldsymbol{w}^n) \end{bmatrix} = \begin{bmatrix} \frac{\partial \varrho(\boldsymbol{w}, \boldsymbol{w}^1)}{\partial \boldsymbol{w}} & \cdots & \frac{\partial \varrho(\boldsymbol{w}, \boldsymbol{w}^n)}{\partial \boldsymbol{w}} \end{bmatrix} ,$$

and where the derivative $\frac{\partial \varrho(\boldsymbol{w}, \boldsymbol{w}')}{\partial \boldsymbol{w}} \in \mathbb{R}^{n_w}$ depends on the employed kernel $\varrho(\cdot, \cdot)$. For

the squared exponential covariance function (2.7) we obtain

$$\begin{aligned}\frac{\partial \varrho(\boldsymbol{w},\boldsymbol{w}')}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}}\left(\sigma_\mathrm{f}^2 \exp\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}')^\mathsf{T}\boldsymbol{\Lambda}(\boldsymbol{w}-\boldsymbol{w}')\right)\right) \\ &= \underbrace{\sigma_\mathrm{f}^2 \exp\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}')^\mathsf{T}\boldsymbol{\Lambda}(\boldsymbol{w}-\boldsymbol{w}')\right)}_{\varrho(\boldsymbol{w},\boldsymbol{w}')} \cdot \frac{\partial}{\partial \boldsymbol{w}}\left(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}')^\mathsf{T}\boldsymbol{\Lambda}(\boldsymbol{w}-\boldsymbol{w}')\right) \\ &= \varrho(\boldsymbol{w},\boldsymbol{w}')(-\boldsymbol{\Lambda})(\boldsymbol{w}-\boldsymbol{w}') \\ &= \varrho(\boldsymbol{w},\boldsymbol{w}')\boldsymbol{\Lambda}(\boldsymbol{w}'-\boldsymbol{w}) \ .\end{aligned} \tag{3.20}$$

The penultimate step results from the identity $\frac{\partial \boldsymbol{x}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{x}} = (\boldsymbol{A}+\boldsymbol{A}^\mathsf{T})\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ for symmetric matrices $\boldsymbol{A} = \boldsymbol{A}^\mathsf{T}$.

At last, assuming a constant prior mean $m(\boldsymbol{w})$, i.e., $\frac{\partial m(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0$, and inserting (3.20) into (3.19) yields the sought gradient of the posterior mean with respect to the regressor vector

$$\nabla m_+(\boldsymbol{w}|\mathcal{D}) = \underbrace{\begin{bmatrix}\varrho(\boldsymbol{w},\boldsymbol{w}^1)\boldsymbol{\Lambda}(\boldsymbol{w}^1-\boldsymbol{w}) & \cdots & \varrho(\boldsymbol{w},\boldsymbol{w}^n)\boldsymbol{\Lambda}(\boldsymbol{w}^n-\boldsymbol{w})\end{bmatrix}}_{[n_w \times n]} \underbrace{\boldsymbol{K}^{-1}(\boldsymbol{z}-m(\boldsymbol{W}))}_{[n\times 1]} \ .$$

3.7.2 Calculation of Terminal Components

We choose the terminal controller as $\kappa_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{k}^\mathsf{T}(\boldsymbol{x}-\boldsymbol{x}_\mathrm{ref}) + u_\mathrm{ref}$ and the terminal cost function as $V_\mathrm{f}(\tilde{\boldsymbol{x}}_k) = \|\boldsymbol{x}_k - \boldsymbol{x}_\mathrm{ref}\|_{\boldsymbol{P}}^2 = (\boldsymbol{x}_k-\boldsymbol{x}_\mathrm{ref})^\mathsf{T}\boldsymbol{P}(\boldsymbol{x}_k-\boldsymbol{x}_\mathrm{ref})$, where $\boldsymbol{k} \in \mathbb{R}^{n_x}$ and $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_x}$ with $\boldsymbol{P} = \boldsymbol{P}^\mathsf{T} > 0$. Vector $\boldsymbol{k}$ and matrix $\boldsymbol{P}$ are computed using the linearization of the GP prediction model (3.15), based on a training data set $\mathcal{D}_\mathrm{ref}$ obtained near the reference $\boldsymbol{x}_\mathrm{ref}$.

The linearization of the nominal NARX model $\boldsymbol{x}_{k+1} = \hat{F}(\boldsymbol{x}_k, u_k)$ with state $\boldsymbol{x}_k = [y_k \ y_{k-1} \ \cdots \ y_{k-m_y} \ u_{k-1} \ \cdots \ u_{k-m_u}]^\mathsf{T}$ takes the following form:

$$\begin{bmatrix} y_{k+1} \\ y_k \\ y_{k-1} \\ \vdots \\ y_{k-m_y+1} \end{bmatrix} = \underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m_y} & a_{1(m_y+1)} \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-m_y} \end{bmatrix} + \underbrace{\begin{bmatrix} b_1 & b_2 & b_3 & \cdots & b_{m_u+1} \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{\boldsymbol{B}} \begin{bmatrix} u_k \\ u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-m_u} \end{bmatrix} \tag{3.21}$$

As the next output is computed using the GP with $y_{k+1} = m_+(\boldsymbol{w}_k|\mathcal{D})$, the parameters in $\boldsymbol{A}$ and $\boldsymbol{B}$ can be determined using the GP posterior mean gradient derived in Section 3.7.1. The linear model (3.21) has to capture the system behavior at the reference point $\boldsymbol{x}_\mathrm{ref}$. Therefore we compute $[a_{11}, \ldots, a_{1(m_y+1)}, b_1, b_2, \ldots, b_{m_u+1}] = \nabla m_+(\boldsymbol{w}_\mathrm{ref}|\mathcal{D}_\mathrm{ref})^\mathsf{T}$ with $\boldsymbol{w}_\mathrm{ref} = (\boldsymbol{x}_\mathrm{ref}, u_\mathrm{ref})$ and a training data set $\mathcal{D}_\mathrm{ref}$ obtained near the

reference.

We furthermore define the feedback vector as $\boldsymbol{k}^\mathsf{T} = \boldsymbol{s}^\mathsf{T}\boldsymbol{P}$ with $\boldsymbol{s} \in \mathbb{R}^{n_x}$ and set $\boldsymbol{G} = \boldsymbol{P}^{-1}$. In addition, we define the state constraint set $\mathcal{X}$ as the combination of n_x multiple instances of $\mathcal{Y}_\mathrm{h}$, i.e., $\mathcal{X} = \mathcal{Y}_\mathrm{h} \times \cdots \times \mathcal{Y}_\mathrm{h}$ and reformulate $\mathcal{X}$ and $\mathcal{U}$ as polyhedral sets of the form $\mathcal{X} = \left\{\boldsymbol{y} \in \mathbb{R}^{m_y+1} : \boldsymbol{q}_i^\mathsf{T}\boldsymbol{y} \leq r_i,\ i = 1, \ldots, n_\mathcal{X}\right\}$ and $\mathcal{U} = \left\{\boldsymbol{u} \in \mathbb{R}^{m_u+1} : \boldsymbol{v}_l^\mathsf{T}\boldsymbol{u} \leq t_l,\ l = 1, \ldots, n_\mathcal{U}\right\}$, where $n_\mathcal{X}$ and $n_\mathcal{U}$ are the respective numbers of inequalities. Then, we compute $\boldsymbol{s}$ and $\boldsymbol{G}$ (and therewith $\boldsymbol{P}$) offline by solving the semidefinite optimization problem

$$\max_{\boldsymbol{G},\boldsymbol{s}} \ \log\left(\det(\boldsymbol{G})\right) \tag{3.22a}$$

$$\text{s.t. } \boldsymbol{G} = \boldsymbol{G}^\mathsf{T} > 0 \tag{3.22b}$$

$$\begin{bmatrix} \boldsymbol{G} & \left(\boldsymbol{A}\boldsymbol{G} + \boldsymbol{b}\boldsymbol{s}^\mathsf{T}\right)^\mathsf{T} \\ \left(\boldsymbol{A}\boldsymbol{G} + \boldsymbol{b}\boldsymbol{s}^\mathsf{T}\right) & \boldsymbol{G} \end{bmatrix} \geq 0 \tag{3.22c}$$

$$\begin{bmatrix} \boldsymbol{G} & (\boldsymbol{G}\boldsymbol{q}_i) \\ (\boldsymbol{G}\boldsymbol{q}_i)^\mathsf{T} & r_i^2 \end{bmatrix} \geq 0, \quad \forall i \in \{1, \ldots, n_\mathcal{X}\} \tag{3.22d}$$

$$\begin{bmatrix} \boldsymbol{G} & (\boldsymbol{s}^\mathsf{T}\boldsymbol{v}_l) \\ (\boldsymbol{s}^\mathsf{T}\boldsymbol{v}_l)^\mathsf{T} & t_l^2 \end{bmatrix} \geq 0, \quad \forall l \in \{1, \ldots, n_\mathcal{U}\}\ . \tag{3.22e}$$

The optimization problem (3.22) results from using the Schur complement in combination with the discrete time Lyapunov equation and the support function concept of closed convex sets. The derivation of (3.22) is provided in detail in Appendix A.4. The resulting $\boldsymbol{s}$ and $\boldsymbol{P}$ are such that the closed-loop linearized system is asymptotically stable in $\mathcal{X}_\mathrm{f} = \left\{\boldsymbol{x}_k \in \mathbb{R}^{n_x} : V_\mathrm{f}(\tilde{\boldsymbol{x}}_k) = \|\boldsymbol{x}_k - \boldsymbol{x}_\mathrm{ref}\|_{\boldsymbol{P}}^2 \leq 1\right\} \subseteq \mathcal{X}$ and $\boldsymbol{k}\mathcal{X}_\mathrm{f} \subseteq \mathcal{U}$.

Remark 28. *If only soft output constraints $\mathcal{Y}_\mathrm{s}$ are considered, Eq.* (3.22d) *that accounts for the hard state constraints can be removed. If furthermore no input constraints $\mathcal{U}$ are considered, also* (3.22e) *can be removed.*

Remark 29. *As is well known, the quadratic Lyapunov function can be used for the nonlinear system in a certain neighborhood of the equilibrium, given certain assumptions hold. The terminal region definition $\mathcal{X}_\mathrm{f} = \left\{\boldsymbol{x} \in \mathbb{R}^{n_x} : V_\mathrm{f}(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_\mathrm{ref}\|_{\boldsymbol{P}}^2 \leq \nu\right\}$, parameterized with ν, could be used to characterize this neighborhood. Then one would need to take the nonlinear remainder term into account to calculate a particular value for ν, which would require the solution of a global optimization problem. Such a problem could be solved by using scenarios or a Monte Carlo approach. However, since the optimal control problem* (3.17) *does not need the terminal region constraint, ν is not required.*

3.8 Simulations

In this section, we provide simulation results for the presented rGP-MPC scheme and compare it with other MPC controllers. To this end, we consider a continuous stirred-tank reactor as simulation case study and present the training data set generation and the resulting terminal components for the MPC, based on the linearized GP posterior mean function. The closed-loop simulations illustrate and validate the theoretical properties, as well as that the proposed controller works as expected. This includes the influence of different initial training data sets, the evolving GP thresholds (Rule 1), the update rule of Theorem 4, the effect of a limited number of training data points, the reduction in computational time, and the region of attraction.

3.8.1 Continuous Stirred-tank Reactor

We consider the continuous stirred-tank reactor (CSTR), where substrate A is converted into product B [169]. The following set of differential equations describes the reactor dynamics:

$$\dot{C}_A(t) = \frac{q_0}{V}\big(C_{A\mathrm{f}} - C_A(t)\big) - k_0 \exp\left(\frac{-E}{RT(t)}\right) C_A(t) \tag{3.23a}$$

$$\dot{T}(t) = \frac{q_0}{V}\big(T_{\mathrm{f}} - T(t)\big) - \frac{\Delta H_{\mathrm{r}} k_0}{\rho C_{\mathrm{p}}} \exp\left(\frac{-E}{RT(t)}\right) C_A(t) + \frac{UA}{V\rho C_p}\big(T_{\mathrm{c}}(t) - T(t)\big) \tag{3.23b}$$

$$\dot{T}_{\mathrm{c}}(t) = \frac{T_{\mathrm{r}}(t) - T_{\mathrm{c}}(t)}{\tau} \tag{3.23c}$$

The coolant temperature reference T_{r} (K) is the input and the concentration C_A (mol/l) the output, i.e., $u = T_{\mathrm{r}}$ and $y = C_A$. The tank and coolant temperatures are T and T_{c}, respectively. The model parameters are given in Tab. 3.1.

3.8.2 Training Data Sets

A raw data set $\mathcal{D}_{\mathrm{raw}}$ (depicted in Fig. 3.3) is generated in simulations using the plant model (3.23). The data points $(z_i, \boldsymbol{w}_i)$ consist of values of $(y_{k+1}, y_k, \ldots, y_{k-m_y}, u_k, \ldots, u_{k-m_u})$, where $z = y_{k+1}$ is going to be the GP output and $\boldsymbol{w} = (y_k, \ldots, y_{k-m_y}, u_k, \ldots, u_{k-m_u})$ its corresponding regressor. Based on this data, we generate the three training data sets $\mathcal{D}_0$, $\mathcal{D}_{\mathrm{ref}}$, and $\mathcal{D}_{\mathrm{comb}}$. The set $\mathcal{D}_0$ is a local subset around the initial point $y_0 = C_A = 0.6\,\mathrm{mol/l}$. The associated input is $u_0 = T_{\mathrm{r}} = 353.5\,\mathrm{K}$. The set $\mathcal{D}_{\mathrm{ref}}$ is a local subset around the target reference point $y_{\mathrm{ref}} = C_A = 0.439\,\mathrm{mol/l}$ with associated input $u_{\mathrm{ref}} = T_{\mathrm{r}} = 356\,\mathrm{K}$. The set $\mathcal{D}_{\mathrm{comb}} = \mathcal{D}_0 \cup \mathcal{D}_{\mathrm{ref}}$ is the union of the two sets.

The sets $\mathcal{D}_0$ and $\mathcal{D}_{\mathrm{ref}}$ are generated by selecting first all points $z = y_{k+1}$ (and their corresponding regressor $\boldsymbol{w}$) that are located within a local neighborhood of the respective set-points and second, by reducing the number of points via exclusion of

Table 3.1: CSTR Parameters

Parameters	Explanation	Value
q_0	Reactive input flow	10 l/min
V	Liquid volume in the tank	150 l
k_0	Frequency constant	$6 \cdot 10^{10}$ 1/min
E/R	Arrhenius constant	9750 K
ΔH_r	Reaction enthalpy	10000 J/mol
UA	Heat transfer coefficient	70000 J/(min K)
ρ	Density	1100 g/l
C_p	Specific heat	0.3 J/(g K)
τ	Time constant	1.5 min
$C_{A\mathrm{f}}$	C_A in the input flow	1 mol/l
T_f	Input flow temperature	370 K

those that add only little information. For a given data point $(z_i, \boldsymbol{w}_i)$, all following points $(z_j, \boldsymbol{w}_j), j > i$, are removed for which $\|\boldsymbol{w}_i - \boldsymbol{w}_j\| < \bar{w}$ with a chosen threshold $\bar{w}$. As a result, the sets are less dense but still contain enough informative data points. The thresholds for $\mathcal{D}_0$ and $\mathcal{D}_\mathrm{ref}$ are chosen such that both sets contain approximately 40 data points.

Note that we used arbitrary inputs to sufficiently excite the system. In practice, one could use tailored excitations to avoid implementation challenges.

Remark 30. *All input and output values are given in the original units of the system* (3.23)*. However, it is beneficial for the modeling process with the GP to normalize the input-output data to the interval* $[0, 1]$.

3.8.3 GP Prediction Models

For the Gaussian process prior we employ a constant mean function $m(\boldsymbol{w}) = c$. Since the underlying process equations are smooth and to obtain the universal approximation property (see Section 3.6.2) we employ the squared exponential covariance function (2.7) with regressor $\boldsymbol{w} = [y_k \; y_{k-1} \; y_{k-2} \; u_k]^\mathsf{T}$. According to (3.10), the NARX state is then $\boldsymbol{x}_k = [y_k \; y_{k-1} \; y_{k-2}]^\mathsf{T}$. The hyperparameters are $\boldsymbol{\theta} = \{c, l_1, l_2, l_3, l_4, \sigma_\mathrm{f}^2\}$ and are computed offline via maximization of (2.8) for each of the three data sets $\mathcal{D}_0$, $\mathcal{D}_\mathrm{ref}$, and $\mathcal{D}_\mathrm{comb}$. We obtain three sets of hyperparameters respectively (Tab. 3.2) and with that three different GP prediction models that use the same prior but different training data sets and hyperparameters. The cross validation results of these different GP models are shown in Fig. 3.4, where we select test points throughout the regions of the respective training data sets. Test points are chosen such that they are not part of $\mathcal{D}_0$, $\mathcal{D}_\mathrm{ref}$, or $\mathcal{D}_\mathrm{comb}$. As can be seen, appropriate GP predictions are achieved with prediction error

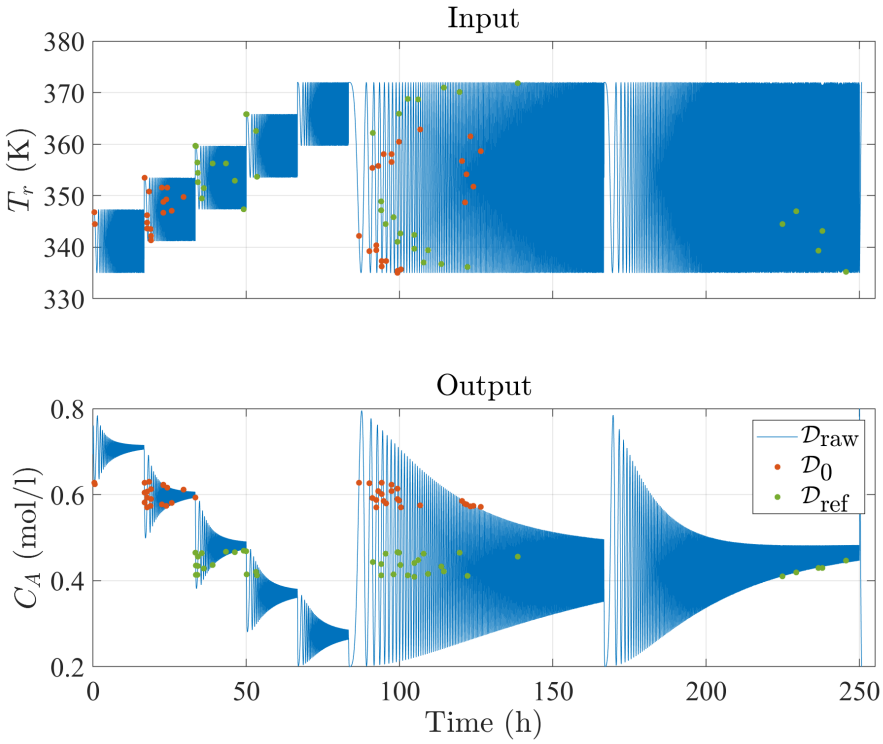


Figure 3.3: Training data sets: The raw data set $\mathcal{D}_{\text{raw}}$ was generated by chirp signals on the input. The sets $\mathcal{D}_0$ and $\mathcal{D}_{\text{ref}}$ are local neighborhoods of the initial point u_0 and the reference point y_{ref} and their associated inputs.

$e^{\text{p}} < \bar{e}^{\text{p}} = 0.02\,\text{mol/l}$ and posterior standard deviation $\sigma_+ < \bar{\sigma}^2 = 5 \cdot 10^{-3}\,\text{mol/l}$ for all three GPs.

Table 3.2: Hyperparameters

	c	l_1	l_2	l_3	l_4	σ_{f}^2
$\mathcal{D}_0$	0.64	0.07	0.29	0.14	9.93	0.06
$\mathcal{D}_{\text{ref}}$	0.36	0.20	11.7	0.64	5.07	0.13
$\mathcal{D}_{\text{comb}}$	0.43	0.42	2.09	1.01	2.83	0.26

3.8.4 rGP-MPC Optimal Control Problem

The continuous-time model (3.23) is discretized with Euler's method using a sampling time of $T_{\text{s}} = 0.5\,\text{min}$. The input constraints are $\mathcal{U} = \{335\,\text{K} \leq T_{\text{r}} \leq 372\,\text{K}\}$, the (hard) output constraints $\mathcal{Y}_{\text{h}} = \{0.35\,\text{mol/L} \leq C_A \leq 0.65\,\text{mol/l}\}$. We add measurement noise $\epsilon \sim \mathcal{N}(0, \sigma_{\text{n}}^2)$ to the output data with $\sigma_{\text{n}}^2 = 0.003^2$, which we furthermore bound[17]

[17] According to the considered system class and Assumption 11 we add Gaussian noise with bounded support. However, due to the large bound of four standard deviations, the difference is so small that the following simulation results are equal to the case of unbounded noise.

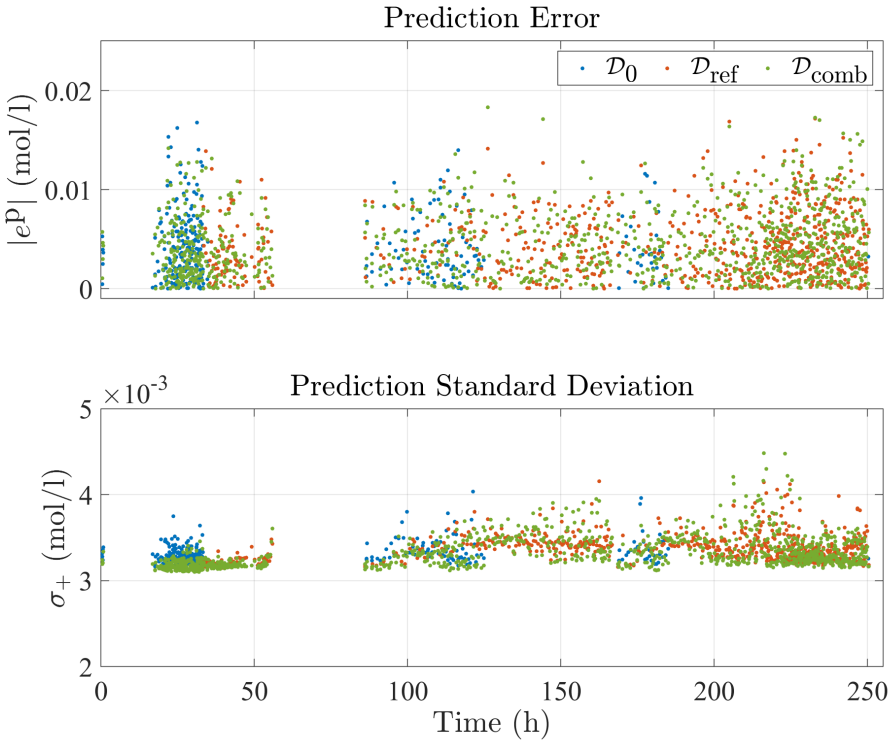


Figure 3.4: Cross validation results: Top, the prediction error e^{p} (3.13) is depicted. Bottom, the posterior standard deviation $\sigma_+(\boldsymbol{w}) = \sqrt{\sigma_+^2(\boldsymbol{w})}$.

by $\pm 4\sigma_{\text{n}}$. This, together with the universal approximation property of the squared exponential covariance function (also discussed in Section 3.6.2.1), yields a prediction error e^{p} that is bounded, thereby fulfilling Assumption 11.

The employed quadratic stage cost is given by

$$\ell_{\text{s}}(\boldsymbol{x}_k, u_k) = \|\boldsymbol{x}_k - \boldsymbol{x}_{\text{ref}}\|_{\boldsymbol{Q}}^2 + \|u_k - u_{\text{ref}}\|_R^2$$

with $\boldsymbol{Q} = \text{diag}(100, 0, 0)$, and $R = 5$.

The linearized GP model at $\boldsymbol{x}_{\text{ref}}$, together with the associated terminal controller and cost function, are computed as explained in Section 3.7 with the training data set $\mathcal{D}_{\text{ref}}$. The resulting linear model becomes

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 0.162 & 0.005 & -0.012 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} -0.034 \\ 0 \\ 0 \end{bmatrix} u_k \ .$$

The feedback vector $\boldsymbol{k}$ and the terminal cost matrix $\boldsymbol{P}$ are computed as presented in Section 3.7 and we obtain

$$\boldsymbol{k}^{\mathsf{T}} = \begin{bmatrix} 1.745 & 0.082 & -0.001 \end{bmatrix} \text{ and } \boldsymbol{P} = \begin{bmatrix} 16.38 & -0.556 & -0.066 \\ -0.556 & 16.32 & -0.554 \\ -0.066 & -0.554 & 16.30 \end{bmatrix} .$$

The prediction horizon is set to $N = 5$ and the resulting optimal control problem is solved in MATLAB using `fmincon` on a standard desktop computer.

3.8.5 Simulation Results

First, we simulate the set-point change from (u_0, y_0) to $(u_{\text{ref}}, y_{\text{ref}})$ and compare the closed-loop results of the rGP-MPC, a batch GP approach (bGP-MPC) that uses a fixed training data set, and an output feedback MPC scheme (oMPC) that uses the model equations (3.23) and acts as a performance bound. We evaluate the performance for the three cases, where $\mathcal{D}_0$, $\mathcal{D}_{\text{ref}}$, and $\mathcal{D}_{\text{comb}}$ are used as initial training data sets. The bGP and rGP are initialized with the same initial training data and hyperparameters but the rGP updates its training data set during operation. We set $\bar{e}^{\text{p}} = \bar{\sigma}^2 = 0$ such that every data point is considered as a candidate for inclusion[18] with no upper limit on the number of data points, i.e., $\bar{n} = \infty$. Hence, no points are removed. Due to the stochastic nature of the noise component, we simulate each case $N_{\text{sim}} = 50$ times. The results are depicted in Fig. 3.5 to Fig. 3.7. To quantify the performance we employ the measure

$$\bar{V} = \frac{1}{N_{\text{sim}}} \sum_{j=0}^{N_{\text{sim}}} \sum_{k=0}^{N_{\text{step}}} \ell\left(\boldsymbol{x}_k^j, u_k^j\right) , \tag{3.24}$$

which averages the stage costs of the resulting state and input sequences over all time steps $k \in \{0, 1, \ldots, N_{\text{step}}\}$, as well as the individual simulations $j \in \{1, 2, \ldots, N_{\text{sim}}\}$. The resulting $\bar{V}$ values are presented in Table 3.3.

Table 3.3: MPC Performance $\bar{V}$ computed by (3.24).

	$\mathcal{D}_0$	$\mathcal{D}_{\text{ref}}$	$\mathcal{D}_{\text{comb}}$
oMPC	59.5	59.5	59.5
bGP-MPC	71.3	95.3	66.2
rGP-MPC	64.5	63.6	66.7

As expected, the output MPC scheme that uses the true model performs best and always the same (see Table 3.1) because it does not depend on any training data points. The rGP outperforms the bGP in the $\mathcal{D}_0$ and $\mathcal{D}_{\text{ref}}$ cases due to the additional information gained during operation. The performance difference is especially large for $\mathcal{D}_{\text{ref}}$, where the bGP, throughout the whole operation, has only data points at the reference at its disposal but not at the initial condition. The rGP performs significantly better due to the added data points at the beginning of operation. In the $\mathcal{D}_{\text{comb}}$ case, the rGP and bGP performance is almost the same for the employed training data points.

[18] Not every data point is added due to the update rule of Theorem 4.

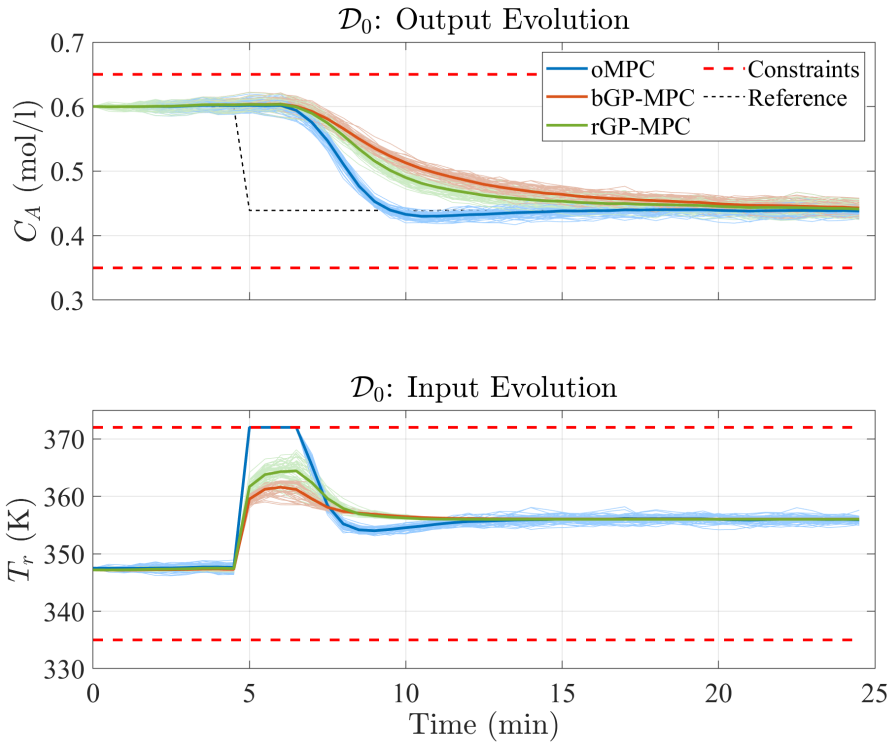


Figure 3.5: Comparison of the three MPC schemes for the case of initial training data $\mathcal{D}_0$. Thin lines represent individual simulations, thick lines represent mean values.

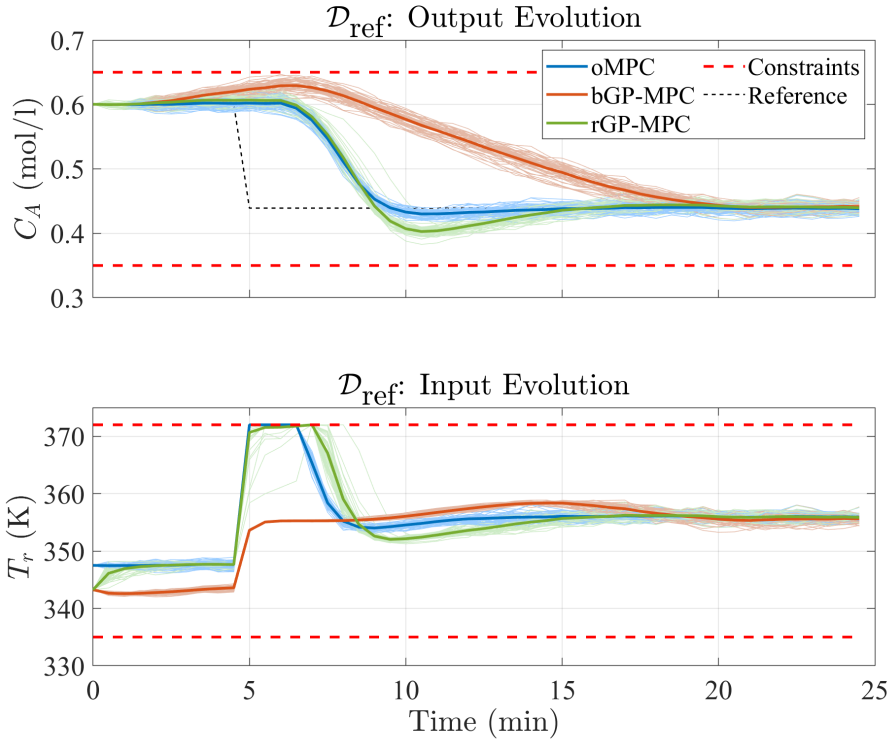


Figure 3.6: Comparison of the three MPC schemes for the case of initial training data $\mathcal{D}_{\text{ref}}$. Thin lines represent individual simulations, thick lines represent mean values.

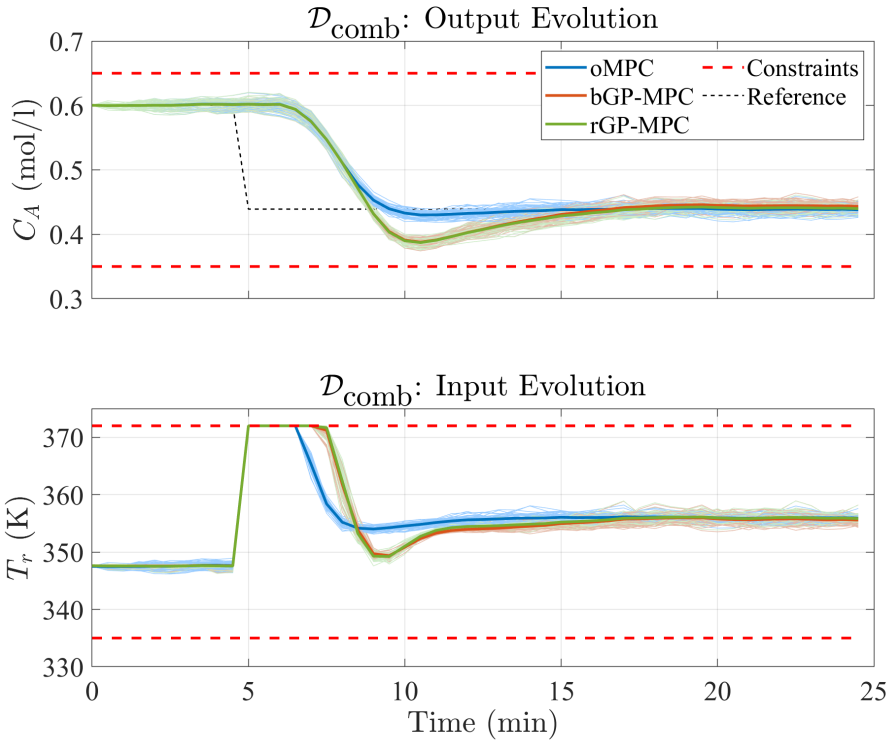


Figure 3.7: Comparison of the three MPC schemes for the case of initial training data $\mathcal{D}_{\text{comb}}$. Thin lines represent individual simulations, thick lines represent mean values.

Remark 31. *The previous simulation results suggest that one should in general prefer the $\mathcal{D}_{\text{ref}}$ case over the other cases, which is convenient for the model predictive control scheme because knowledge at the reference is required anyway to determine the terminal cost and controller (see Section 3.7). Furthermore, this also suggests a practical rule for offline hyperparameter determination, namely that the hyperparameters should be optimized for a data set that contains the target reference.*

In the second set of simulations, we investigate the influence of different thresholds used in Rule 1, i.e., different values for the maximum prediction error $\bar{e}^{\text{p}}$ and the maximum prediction variance $\bar{\sigma}^2$. To this end, we start with Fig. 3.8 that combines the rGP results of the previous figures for the three training data cases, together with the now plotted evolution of the prediction error e^{p} and the prediction variance σ_+^2. In particular the prediction variance illustrates nicely the difference between the three cases. In the case of $\mathcal{D}_0$, the variance is small at the beginning and increases around $t = 8\,\text{min}$ when the system leaves the neighborhood of the initial condition and moves towards the reference. The same holds, but the other way round, for the case with $\mathcal{D}_{\text{ref}}$, where the initial ($t < 3\,\text{min}$) large error and variance is caused by their computation before the first data points are added to the training set. The prediction error bound μ is 0.033, 0.021, and 0.024 for the cases $\mathcal{D}_0$, $\mathcal{D}_{\text{ref}}$, and $\mathcal{D}_{\text{comb}}$ respectively.

Fig. 3.9 and Fig. 3.10 show results for different threshold values, where we focus on the simulation case with $\mathcal{D}_{\text{ref}}$. The results illustrate that instead of adding all data

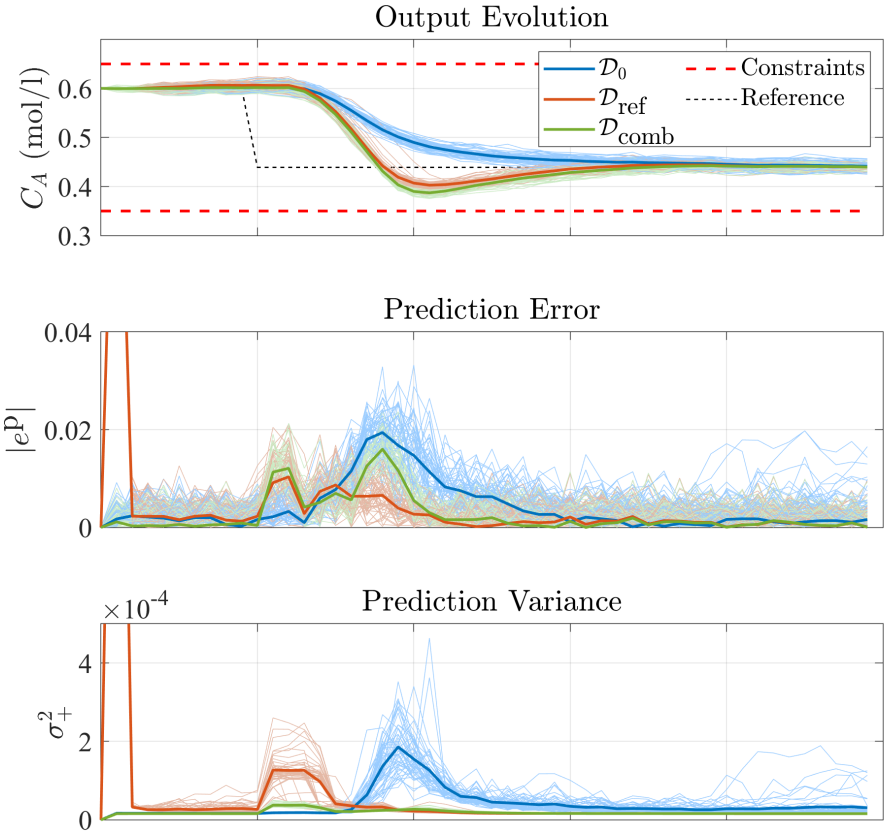


Figure 3.8: Simulation results with the rGP-MPC for the different training data cases together with the absolute value of prediction error $|e^{\text{p}}|$ and the prediction variance σ_+^2.

points, almost the same closed-loop performance can be achieved by adding only a fraction of them. Hence, this shows not only that online learning can be achieved but also that it allows working with significantly smaller training data sets, which in turn result in lower computational costs.

After evaluating the influence of the parameters of Rule 1, we illustrate the influence of the update rule in Theorem 4 that guarantees a decrease of the value function. To this end, we continue with the $\mathcal{D}_{\text{ref}}$ case and additionally insert outliers into the output measurements in the course of the simulations. The effect of the update rule is shown in Fig. 3.11. With it, the results are almost the same as before, except for the distortions due to the outliers, which however are compensated shortly after. All simulation outcomes are very similar in that case. Without the update rule, the resulting mean output sequence is different but not necessarily worse (smaller rise time, similar settling time, no overshoot) than the mean output sequence with the update rule. Some of the individual simulation outcomes perform even better, which is an indication that data points with valuable information are indeed discarded by the update rule as was also

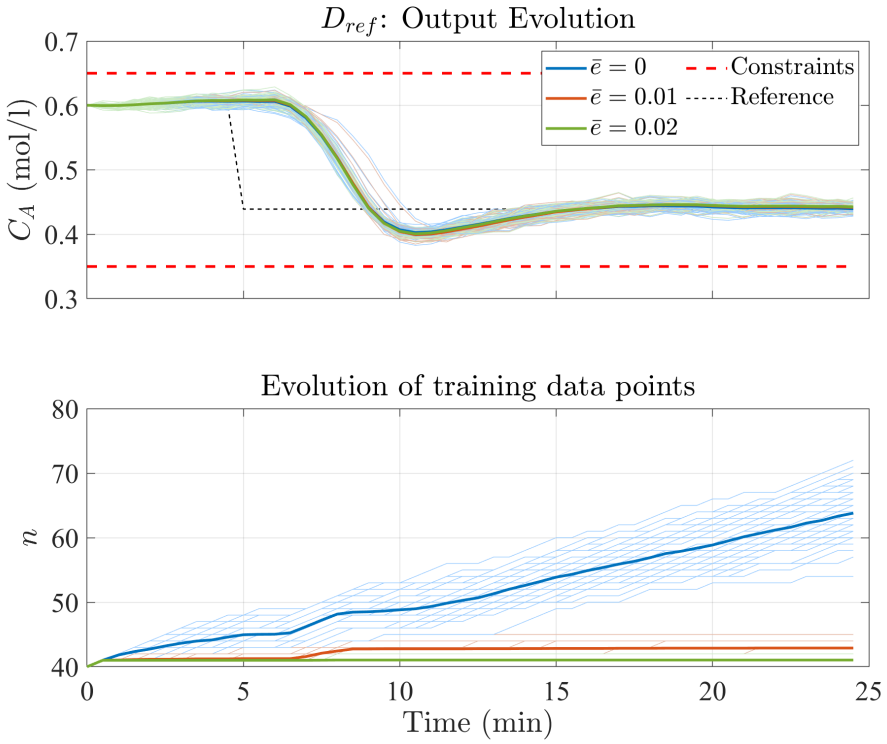


Figure 3.9: Influence of $\bar{e}^{\mathrm{p}}$ on the rGP-MPC with initial training data $\mathcal{D}_{\mathrm{ref}}$. With $\bar{e}^{\mathrm{p}} = 0$, every encountered data point is considered to be added to the training data set. The variance threshold $\bar{\sigma}^2$ was set to a large value to not affect the result.

pointed out in Remark 20. On the other hand, the variability among the individual simulations is much larger. Several of the simulated output evolutions converge slower to the target, some do not converge at all until the end of the simulation. This is a direct result of the corresponding input sequences computed by the optimizer. In between 5 min and 11 min, the deviation of the mean input sequence from the optimal input sequence of the performance bound (oMPC, see Fig. 3.5 to Fig. 3.7) is larger than in the case with the update rule. Furthermore, the individual input sequence outcomes vary considerably, even hitting the lower constraints. Due to the inclusion of every encountered data point candidate, the prediction model changes in some cases in an unfavorable way during the respective simulations, which leads to the depicted results. Note that qualitatively the same results (including not converging output sequences) are obtained, even without outliers. For instance, between the reference change at 5 min and the first outlier at 7 min, we observe that the input sequences already deviate considerably from the case with an active update rule, i.e., the outlier is not the cause but usual noisy data points. This illustrates the importance of the update rule in Theorem 4, not only for theoretical guarantees but also in terms of practical application.

Next, we consider the case that the number of training data points is limited by $\bar{n}$. For the case of $\mathcal{D}_{\mathrm{ref}}$ we set $\bar{n} = 40$, which is the number of initially available training

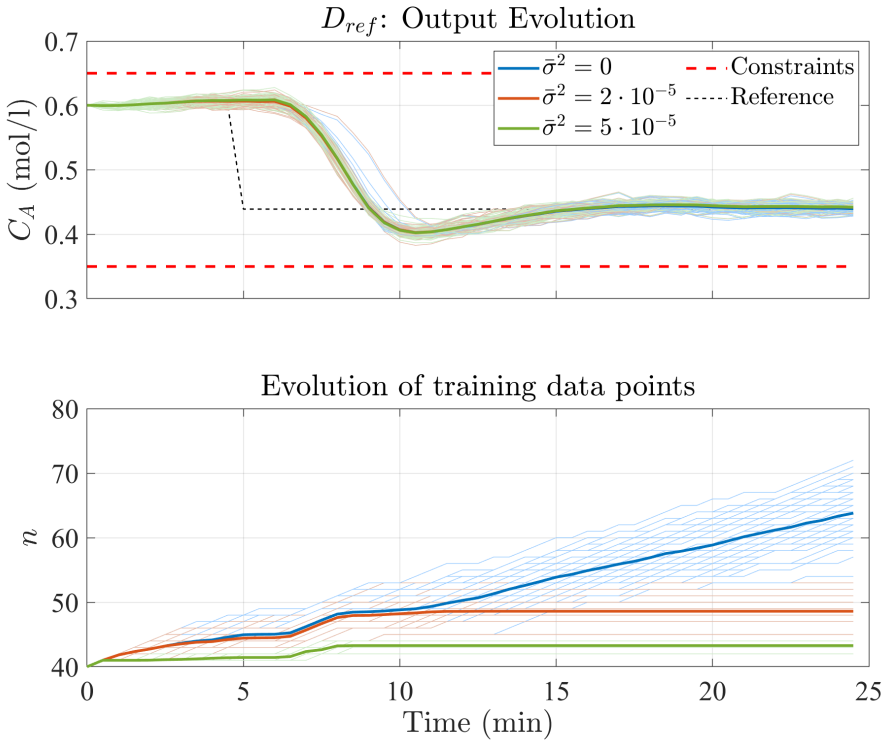


Figure 3.10: Influence of $\bar{\sigma}^2$ on the rGP-MPC with initial training data $\mathcal{D}_{\text{ref}}$. With $\bar{\sigma}^2 = 0$, every encountered data point is considered to be added to the training data set. The prediction error threshold $\bar{e}^{\text{p}}$ was set to a large value to not affect the result.

points, i.e., the training data set cannot increase but old data points are exchanged with newer more informative ones. To this end, whenever a new point is added, the oldest data point is removed. In Fig. 3.12, we compare the bGP (the initial training data set is not updated at all), the rGP with $\bar{n} = \infty, \bar{e}^{\text{p}} = \bar{\sigma}^2 = 0$ (every encountered data point is considered to be added), and the rGP with $\bar{n} = 40, \bar{e}^{\text{p}} = 0.01, \bar{\sigma}^2 = 2{\cdot}10^{-5}$ (data points are only exchanged). The bGP result is the same as in Fig. 3.6 and represents the worst case because the training data set is not updated at all. The $\bar{n} = \infty$ case on the other hand represents the performance bound for this specific case because it includes the maximum of the incoming data points and does not remove any. As can be seen, the reaction of the limited case is a bit slower than the performance bound case but the resulting settling times are almost identical. Thus, with a training data set of only 40 points, where the points are exchanged during operation, almost the same performance can be achieved for the considered example as if every encountered point was included in the training data set $\mathcal{D}$.

Besides the computational cost reduction due to the possibility to work with smaller training data sets, we also illustrate the computational reduction due to the recursive update of the Cholesky factor. In Fig. 3.13, we continue with the $\mathcal{D}_{\text{ref}}$ case, where we add every incoming point to the training data set and compare the computation

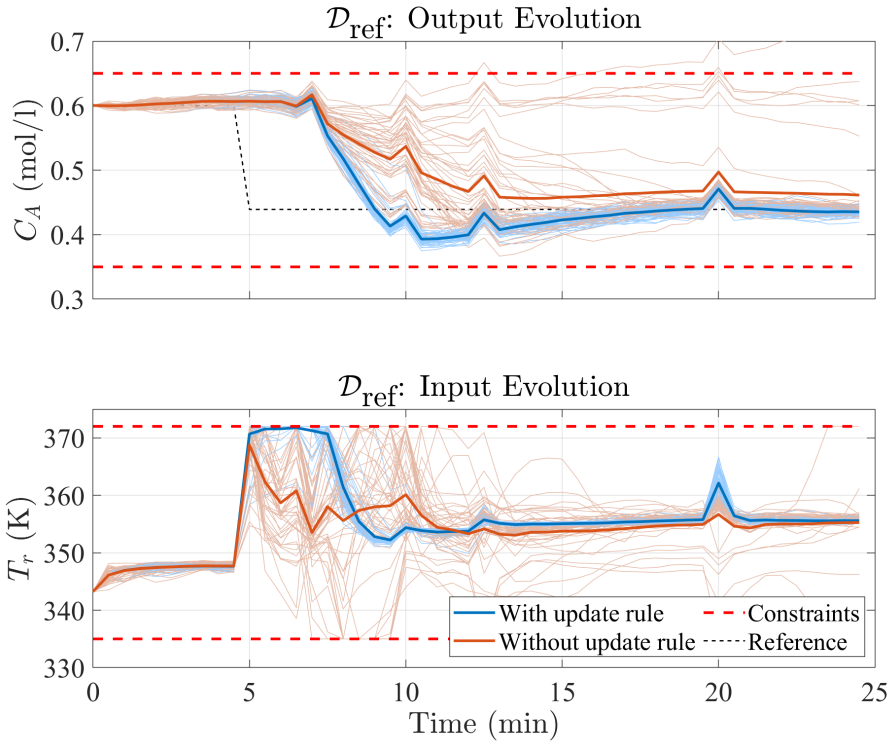


Figure 3.11: Influence of the update rule in Theorem 4, which permits inclusion of data point candidates only if they result in a decreasing value function. Outliers are generated at 7 min, 10 min, 12.5 min, and 20 min.

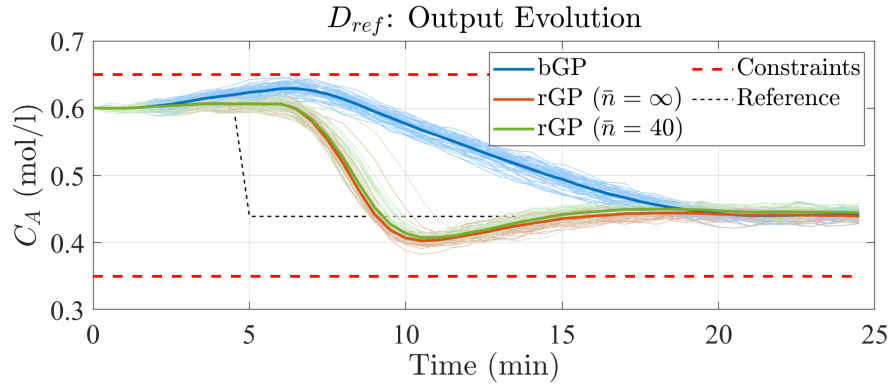


Figure 3.12: Influence of a limited number of training data points on the rGP-MPC with initial training data $\mathcal{D}_{\text{ref}}$.

times of the full and the recursive update of the Cholesky factor. The results show that the larger the training data set becomes, the larger the absolute and relative computational reduction. At $t = 24\,\text{min}$ the full recomputation of the Cholesky factor increases the computation times significantly. Investigations point to the reason lying in the generation of the covariance matrix and the inner workings of Matlab's `chol` function to compute the Cholesky decomposition.

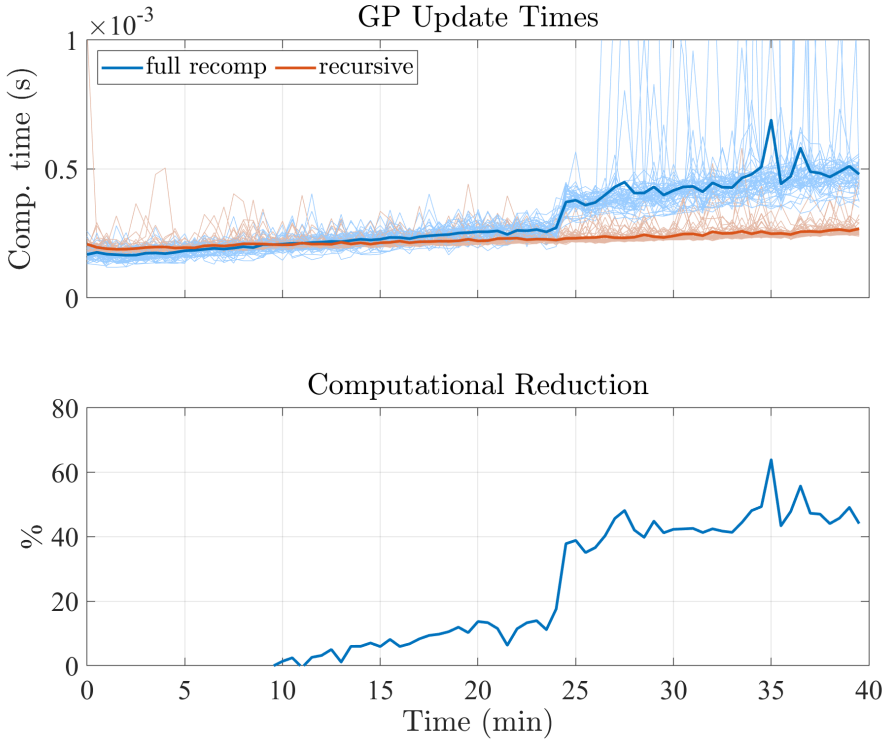


Figure 3.13: Comparison of computation times of the full recalculation of the Cholesky factor and the recursive update. The computational reduction that goes along with the recursive update increases with the amount of training data points.

At last, we present simulations of the robust invariant set $\Omega_0^r(\lambda)$, also denoted region of attraction (ROA), and how it changes for different maximum prediction errors μ. We continue with the $\mathcal{D}_{\text{ref}}$ case with $\bar{e}^{\text{p}} = \bar{\sigma}^2 = 0$ such that every data point is considered as a candidate for inclusion and $\bar{n} = \infty$ so that no points are removed from $\mathcal{D}_k$. We select 32 different initial conditions $\boldsymbol{x}_0 = [y_0\ y_0\ y_0]^\mathsf{T}$ and repeat each of the associated simulations 30 times. These simulations are furthermore repeated for different μ values, which are obtained by varying the measurement noise from $\sigma_{\text{n}}^2 = 0.003^2$ to $\sigma_{\text{n}}^2 = 0.012^2$, where μ is then the largest error of all simulation runs and time steps. The result in Fig. 3.14 yields a clear tendency. The larger μ, the smaller $\Omega_0^r(\lambda)$.

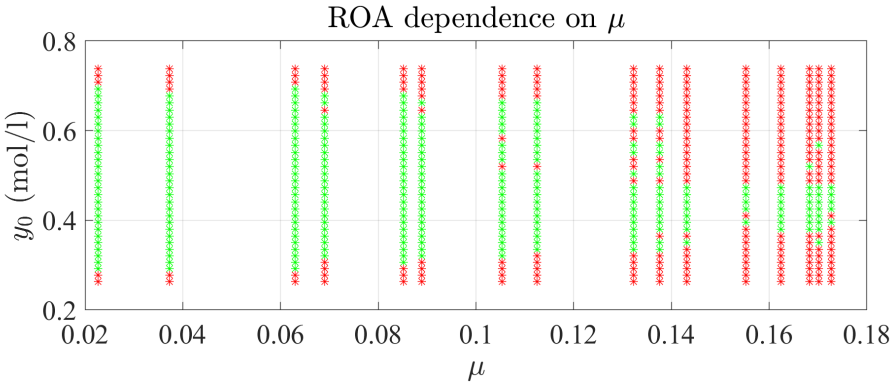


Figure 3.14: Change of the region of attraction Ω^0 for different μ. Red stars denote infeasible initial conditions, green stars feasible initial conditions. An initial condition is marked as infeasible if at least one simulation resulted in a constraint violation.

3.9 Conclusions

Gaussian processes are increasingly used as prediction models in model predictive control. We presented an output feedback model predictive control formulation that uses a Gaussian process-based nonlinear model for prediction, denoted as rGP-MPC. The approach is applicable to a wide variety of processes, including those where the state cannot be measured or is difficult to be estimated. In order to gear the control approach towards the possible application for fast processes, we reduced the computational costs by reducing and eventually limiting the maximum number of training data points. To counteract a limited amount of training data (and with that limited process knowledge) and account for a possibly changing process and process environment, the approach includes online learning by means of updating the training data set during operation. To this end, the concept of evolving Gaussian processes was adapted together with a recursive update of the Cholesky decomposition to minimize the computational costs associated with updating the covariance matrix inverse.

We showed that the presented rGP-MPC scheme is nominally, as well as inherently robustly (input-to-state) stable with respect to the prediction error, despite a changing prediction model. To this end, we presented a structured approach to determine the necessary MPC terminal components, based on a GP prediction model at the target point. Notably, the theoretical guarantees hold not only for rGP-MPC. They are applicable for general model predictive control schemes that use a Gaussian process, or even other machine learning methods, as prediction model. This possibly explains why many of the schemes that combine model predictive control and Gaussian processes in the literature present robust stability properties, though not specifically designed for it.

Simulations verfied that it is in general possible to start with limited a priori process knowledge and to refine the model during operation. One important finding is that it is particularly beneficial to start with a model that captures at least the behavior at the target reference, which is fortunately an intrinsic necessity for all MPC schemes that use a terminal region, cost, and controller to guarantee recursive feasibility and stability. In the case of fixed GP hyperparameters during online operation, a further consequence is that the hyperparameters should be optimized offline for a data set that captures the target reference. The simulations underline that rGP-MPC yields good closed-loop performance with few training data points, thereby efficiently reducing the computational load. This presents itself as a possible option for very fast processes, where hyperparameter optimization is not an option but some kind of online learning is desirable.

Current research is seeking to validate and confirm the presented results for application cases that exhibit a more nonlinear system behavior and faster dynamics. One such example is wind energy generation via tethered kites. Besides the practical rele-

vance for renewable energy generation, this application case is also interesting from a theoretical point of view because the rGP-MPC scheme has to be extended from set-point changes to time-varying reference tracking. Interesting questions in this context are, for instance, what conditions has the initial training data set to satisfy to achieve acceptable tracking results, and how to automatically compute safe thresholds for the data inclusion approach?

Similar to time-varying reference tracking, another question to investigate would be how the approach performs for time-varying processes. A possible hypothesis to test could be to combine the squared exponential covariance function with a non-stationary one to account for time variance in the process model.

At last, future work has to aim at implementing the presented approach in laboratory experiments, either with a fully Gaussian process-based prediction model or via a combination of a deterministic base model and the Gaussian process model.

4 Gaussian Process supported Control of Scanning Quantum Dot Microscopy

In this chapter, we present a control framework for scanning quantum dot microscopy, a novel microscopy technique that generates images of electrostatic potentials of nanostructures. The designated framework uses a two-degree-of-freedom control concept, including a Gaussian process to learn online, static but unknown maps related to the scanned nanostructure. The Gaussian process model is used to calculate predictions that are utilized as a feedforward signal in the two-degree-of-freedom control framework.

The outline of this chapter is as follows. After a first introduction into the topic (Section 4.1), we present the details of scanning quantum dot microscopy, including the physical working principle, the original image generation process, as well as a model derived for simulation purposes (Section 4.2). The proposed two-degree-of-freedom control framework is detailed in Section 4.3, which includes different developed options for the feedback and the feedforward part. The performance of the control scheme and its different parts is investigated via simulations in Section 4.4 and experimental results are presented in Section 4.5. The chapter is concluded in Section 4.6.

4.1 Introduction

Research in physical chemistry and nanotechnology is driven by a wide field of possible future applications such as, for instance, molecular manipulation that would enable the assembly of molecular machines or nanoscopic electric circuits by single molecule placement [56, 134]. Doing so requires a fundamental understanding of the specific characteristics of the basic building blocks of matter, namely atoms and molecules. Some of these characteristics are the electric properties that govern in many cases the functionality of nanoscale objects and systems. This is particularly important for new materials and devices associated with nanoscale electronics, such as semiconductors. The investigation of electrostatics at the nanoscale becomes therefore more and more important and is a vivid field of ongoing research (see, e.g., [59, 78, 127, 137, 196]).

Scanning quantum dot microscopy (SQDM), introduced in [67, 195] is a novel microscopy technique that allows to measure and generate 2D images of the electrostatic potentials (Fig. 4.1) of surface nanostructures (single atoms or molecules, or nanoscopic structures built from several atoms/molecules) with sub-nanometer resolution at large imaging distances. It furthermore allows to separately image the electrostatic poten-

tial and the surface topography. It was recently shown in [196] that SQDM does not only allow to generate qualitative but also quantitative measurements of electrostatic potentials. The work [196] also demonstrated large-scale imaging (Fig. 4.1), resolving both small (single atoms and molecules) and large structures (an island composed of several hundreds of molecules) in the same image, whereas in the previous works [67, 195] only isolated atoms and molecules could be imaged independently of each other. Given this progress scanning quantum dot microscopy has become a widely applicable microscopy technique for the area of nanotechnology.

One of the key components is a tailored control framework, as presented in this chapter. Its main contribution is a significant increase of the scanning speed and therewith a decrease of the time required for image generation. This in turn enables to scan larger samples than before, as for example shown in Fig. 4.1 [196].

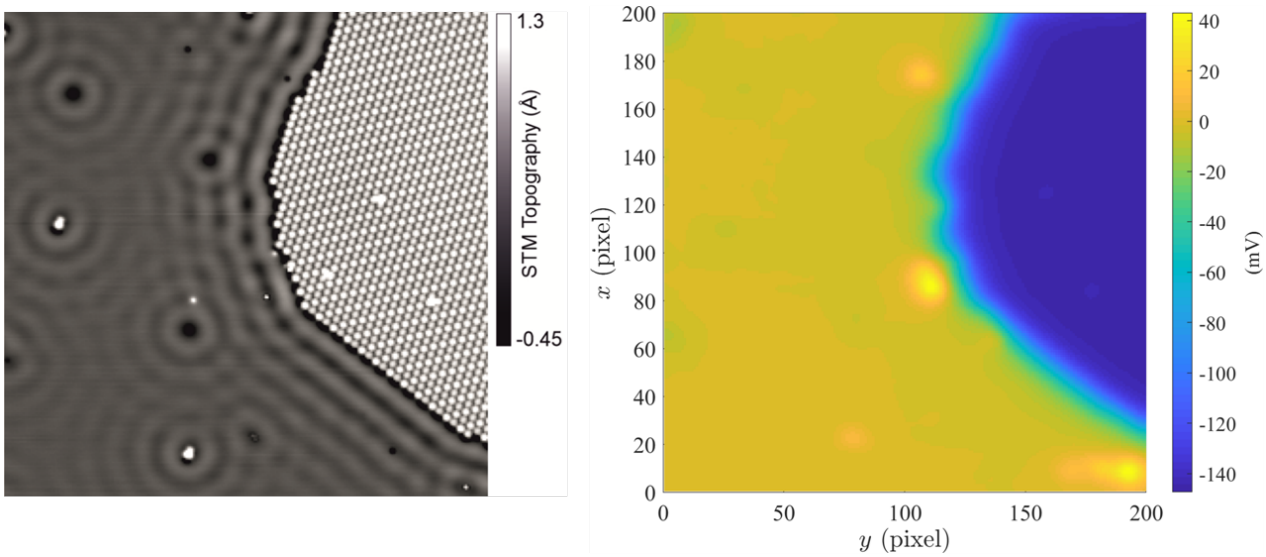


Figure 4.1: Left: Scanning tunneling microscope image of a sample [196] that shows a silver Ag(111) substrate with a large island located on the right, consisting of PTCDA (Perylenetetracarboxylic dianhydride) molecules and further individual features distributed on the whole sample (see [196] for more information). The image size is $600 \times 600\,\text{Å}^2$. Right: 2D electrostatic potential image generated with SQDM of the same sample. The SQDM image consists of 200×200 pixels.

Contributions

The contributions of this chapter are as follows.

- We present the **first control framework for scanning quantum dot microscopy**, which is a key enabling component that turns SQDM into a well applicable microscopy technique.
- The designated framework is a **two-degree-of-freedom control** scheme that consists of a feedback and a feedforward part.

  - Feedback part: Two different controllers tailored to SQDM are developed; an extremum seeking controller and a slope tracking controller.
  - Feedforward part: Two different feedforward signal generators are developed; an approach that uses the last line scan as a prediction for the next line and an approach that utilizes a Gaussian process that online learns unknown static maps, which arise in the process of SQDM and can be leveraged to generate suitable feedforward signals.

- The control framework allows to **continuously scan** the sample (opposed to the initial method of generating SQDM images, based on spectroscopy grids [195]) at different scan speeds, which can be adapted during operation. This allows order-of-magnitude **faster image generation** and eliminates the need for spectroscopy. This puts it in line with other microscopy techniques like scanning tunneling microscopy [22] and atomic force microscopy [6].

- The resulting increased scan speeds enable to generate **larger images and with higher resolution** than before.

- The control framework furthermore allows scanning images with **highly varying electrostatic potentials**. Before, highly varying electrostatic potentials resulted in an even slower image generation process due to the employed spectroscopy grids.

The extremum seeking feedback controller and the Gaussian process-based feedforward were published in [122]. Slope tracking controller and the last line feedforward are under preparation for publication and were used to generate the results published in [196].

4.2 Scanning Quantum Dot Microscopy

In this section, we explain the working principle of scanning quantum dot microscopy (Section 4.2.1), present the original image generation process (Section 4.2.2), and provide a model that will be required for simulations (Section 4.2.3).

4.2.1 Working Principle

Scanning quantum dot microscopy is able to measure electric surface potentials and allows distinguishing between topographical effects and those generated by electric charges. It utilizes a frequency modulated non-contact atomic force microscope (NC-AFM [6]), operating in ultra-high vacuum and at a temperature of 5 K. The atomically sharp tip (Fig. 4.2) is mounted on a tuning fork (qPlus sensor [62]) that oscillates with a frequency $f = f_0 + \Delta f$ of around 30 kHz, where f_0 is the free resonance frequency

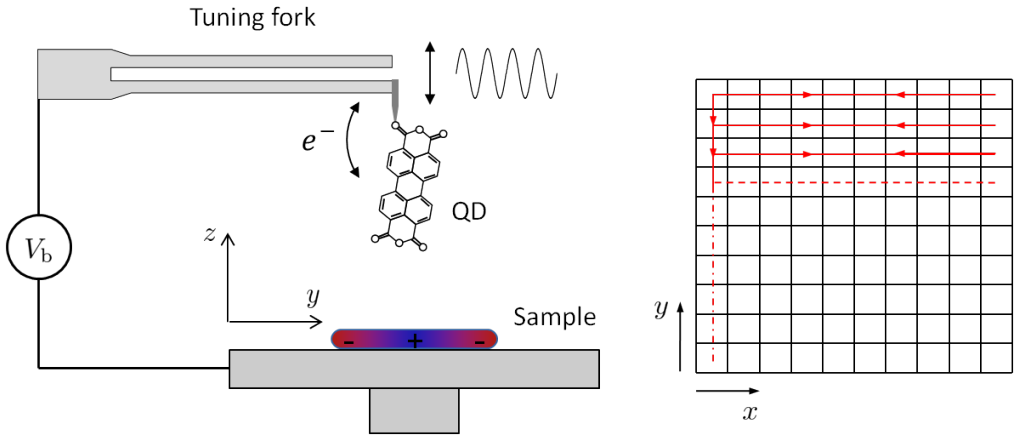


Figure 4.2: Schematic of scanning quantum dot microscopy (left). The tuning fork tip of a frequency modulated non-contact atomic force microscope is decorated with a quantum dot (QD), a bias voltage source V_b is connected between the tip and the sample. Depending on the electrostatic potential of the nanostructure on the sample surface and the bias voltage, a single electron (e^-) can tunnel back and forth between the tip and the quantum dot. The tip together with the quantum dot is moved back and forth in a line by line raster scanning pattern (right).

and Δf is the frequency change that is caused by a vertical force gradient acting on the tip. Typically, this force is the result of the tip-sample interaction.

The AFM tip is decorated [56, 58, 183, 193, 194] with a quantum dot (QD)[1], a nano-sized object whose energy levels can take only discrete values. Changes of the electrostatic potential Φ_s of the surface can change the QD's charge state via gating as an electron tunnels from the tip into the QD or the other way round (charging or discharging), which leads to an abrupt change in the tip-sample force. These tip-sample force changes lead to a shift in the measured oscillation frequency of the NC-AFM, effectively transducing the information about the electrostatic potential of, e.g. a complex nanostructure, into the measurable quantity Δf. Monitoring the charging events of the quantum dot while scanning the sample is the basic working principle of scanning quantum dot microscopy.

In order to detect the charging events, a bias voltage source V_b is connected to the sample while the tip is grounded (Fig. 4.2). The associated electrostatic potential Φ_b that is generated by V_b is superimposed on the intrinsic electrostatic surface potential of the sample Φ_s. At the position of the QD/the microscope tip $\boldsymbol{p} = [x\ y\ z]^\mathsf{T}$, the *effective* electrostatic potential is $\Phi^*(\boldsymbol{p}) = \Phi_s(\boldsymbol{p}) + \Phi_b(\boldsymbol{p})$. Accordingly, a change in V_b then leads to a change of the effective electric potential $\Phi^*(\boldsymbol{p})$. Therefore, by varying V_b, charging or discharging of the QD can be induced at will.

The quantum dot's charging events lead to a change in the tip-sample force, whose gradient is proportional to Δf for small amplitudes of the AFM tip oscillation [61].

[1]Currently a PTCDA (Perylenetetracarboxylic dianhydride) molecule serves as quantum dot.

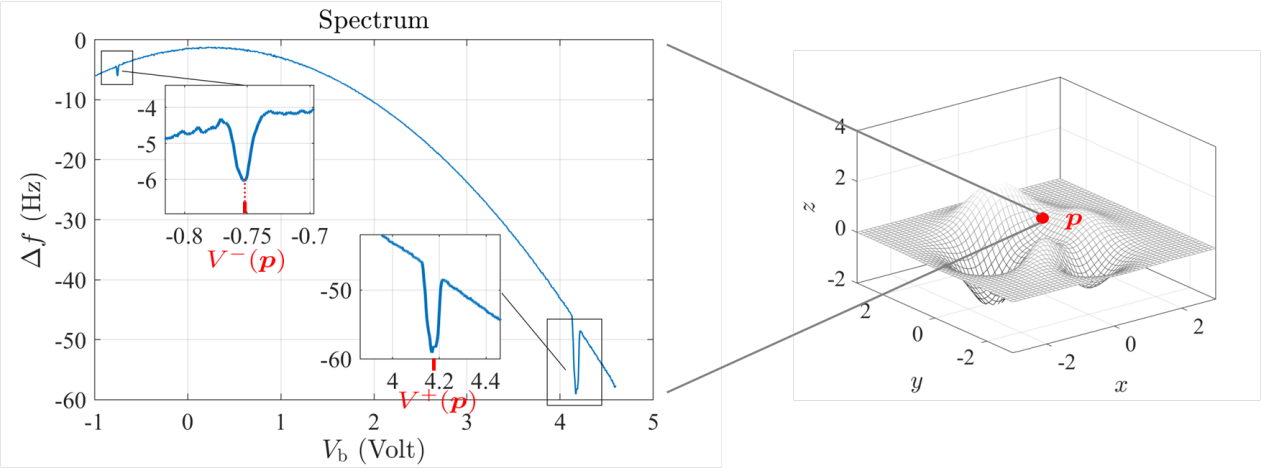


Figure 4.3: The spectrum $\Delta f(V_\mathrm{b}, \boldsymbol{p})$ (left image) describes how the tuning fork oscillation frequency changes with the bias voltage V_b and the position $\boldsymbol{p}$ of the microscope tip over an exemplary sample topography (right image). The overall parabolic shape of the spectrum is a consequence of the tip-sample capacitance [70]. The SQDM specific dips result from the charging events of the quantum dot. The voltage values where the two dips reach their minimum are indicated by $V^-(\boldsymbol{p})$ and $V^+(\boldsymbol{p})$.

The changes of the tip-sample force generated by the charging events appear in the so-called *spectrum* as features that we denote as *dips* (Fig. 4.3). Within the voltage interval of these two dips, one electron tunnels back and forth between the QD and the microscope tip. Thus, the dips lead to different V_b intervals with different charge states of the quantum dot. We denote the voltage values at which the dips reach their minimum with $V^-(\boldsymbol{p})$ and $V^+(\boldsymbol{p})$, or short $V^\mp(\boldsymbol{p})$. These values characterize the dips' positions within the spectrum and depend on the microscope tip position $\boldsymbol{p}$ because the electrostatic potential Φ_s varies in space and therefore the charging events occur at different V_b values for different tip positions (see Fig. 4.4). Hence, the spectrum, denoted by $\Delta f(V_\mathrm{b}, \boldsymbol{p})$, depends on both the bias voltage and the tip position and therewith on the surface and it's properties.

By means of

$$\Phi^*(\boldsymbol{p}) = \frac{V_0^- \cdot \Delta V(\boldsymbol{p})}{\Delta V_0} - V^-(\boldsymbol{p}) \ , \tag{4.1}$$

the effective surface potential $\Phi^*(\boldsymbol{p})$ at the position of the tip can be calculated, where $\Delta V(\boldsymbol{p}) = V^+(\boldsymbol{p}) - V^-(\boldsymbol{p})$ and $V_0^-, \Delta V_0$ are calibration values usually measured at the start of an image scan. The actual surface potential Φ_s can be recovered from $\Phi^*(\boldsymbol{p})$ through deconvolution in post-processing. A deconvoluted version of the SQDM image in Fig. 4.1 can be found in [196]. In the rest of this chapter we will deal with Φ^* unless otherwise indicated. For more details see [197]. The $V^\mp(\boldsymbol{p})$ maps together with $\Phi^*(\boldsymbol{p})$ of the sample shown in Fig. 4.1 are illustrated in Fig. 4.5.

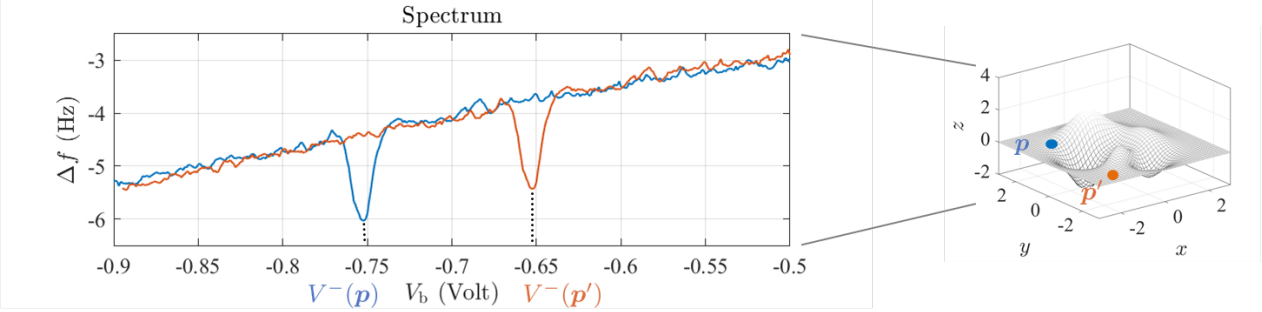


Figure 4.4: Exemplary cutout (left image) of the spectrum $\Delta f(V_\mathrm{b}, \boldsymbol{p})$ illustrating the movement of the negative dip along the parabola at different microscope tip positions $\boldsymbol{p}$ and $\boldsymbol{p}'$ (right image). Each plotted line corresponds to the spectrum at the respective tip position.

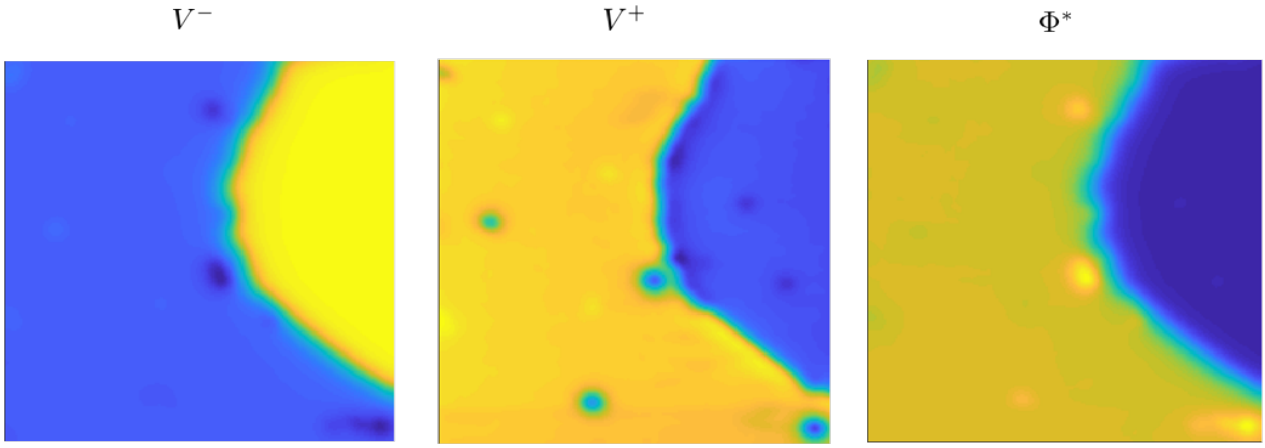


Figure 4.5: From left to right: the $V^-(\boldsymbol{p})$ map, the $V^+(\boldsymbol{p})$ map, and the resulting effective electrostatic potential $\Phi^*(\boldsymbol{p})$ of Fig. 4.1.

4.2.2 Original Image Generation Process

The image generation process previously used [67, 195] performed on the basis of pixel discretization (Fig. 4.2), where the tip with the quantum dot is moved from pixel to pixel. At the first pixel $\boldsymbol{p}_0$, a complete spectrum (like Fig. 4.3) is measured and the $V_0^{\mp}$ values, i.e., the positions of the dips, at this pixel are determined. Then, the tip is moved to the next pixels to determine the corresponding $V^{\mp}(\boldsymbol{p})$ values. To this end, the intervals in which the $V^{\mp}(\boldsymbol{p})$ values will change while scanning the sample are assumed to be known a priori. Then, the tip is moved from pixel to pixel and the bias voltage V_b is swept accordingly within these two intervals (e.g. a voltage range of 0.2 V instead of 6 V for the complete spectrum). This results in the measurement of local dip spectra. After obtaining the local dip spectra for all pixels, the $V^{\mp}(\boldsymbol{p})$ values are determined for each pixel and used in (4.1) to generate $\Phi^*(\boldsymbol{p})$.

The main limitation of this image generation process is the required large measurement time. For instance, measuring the local dip spectra takes about 3 s for each dip and pixel for a certain V_b interval size. Hence, the determination of the complete

$V^{\mp}(\boldsymbol{p})$ maps in Fig. 4.5 would require 66.7 h (in comparison, the data shown in Fig. 4.5 was generated by the proposed control approach in this thesis, which results in significantly smaller measurement times). This severely limits the applicability of scanning quantum dot microscopy. In particular,

- the microscope is blocked for several hours for the generation of one image,
- the longer the measurement time, the higher the probability of failures, and
- effects like (e.g. piezo and temperature) drift increase and deteriorate image quality.

Furthermore, obtaining $\Phi^*(\boldsymbol{p})$ images from a grid of spectra limits scanning quantum dot microscopy in several ways:

- Generation of fast and rough images for a first impression is not possible.
- Generation of images like the one presented in Fig. 4.1 and Fig. 4.5 are practically impossible.
- The measurement times of the local spectra increase if the aforementioned voltage intervals, wherein the dips move, increase. This can occur, for instance, for other substrate-sample combinations with stronger electrostatic variations and thus, reduce the applicability even further.

4.2.3 Model

To evaluate and develop the control framework, a model and simulation environment was developed, which involves a model of scanning quantum dot microscopy, together with reference data maps of $V^{-}(\boldsymbol{p})$ and $V^{+}(\boldsymbol{p})$.

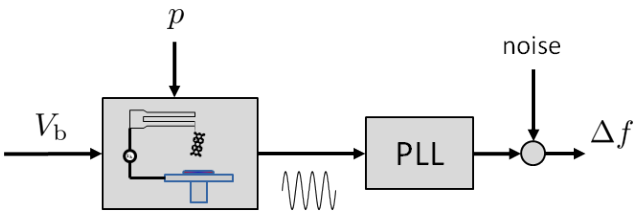


Figure 4.6: SQDM block diagram.

Fig. 4.6 illustrates the SQDM system. In the experiment, the current bias voltage V_b and tip position $\boldsymbol{p}$ result in an oscillation frequency f of the tuning fork different from the free resonance frequency f_0. Hence, V_b and $\boldsymbol{p}$ are the inputs to the system and the frequency change Δf is the output. The bias voltage V_b is the control input to the system and can be chosen freely, whereas the tip position $\boldsymbol{p}$ changes according to the line by line raster scanning pattern as shown in Fig. 4.2.

The relatively small frequency change Δf (only up to a few Hz, see Fig. 4.4), compared to 30 kHz of the free resonance frequency f_0, is determined from the oscillation signal using a phase-locked loop (PLL), which can be modeled as a first order system (right block in Fig. 4.6)

$$G_{\mathrm{PLL}}(s) = \frac{\omega_{\mathrm{PLL}}}{s + \omega_{\mathrm{PLL}}}$$

with bandwidth ω_{PLL}. The output of the PLL is furthermore corrupted by white Gaussian noise.

Regarding the left block in Fig. 4.6, when V_{b} or $\boldsymbol{p}$ changes, the oscillation frequency changes almost instantaneously. Therefore, this block can be modeled as a memoryless function. In fact, this function is the spectrum of Fig. 4.3. To simulate SQDM and the corresponding image generation process, the spectrum $\Delta f(V_{\mathrm{b}}, \boldsymbol{p})$ has to be available as an analytic function that depends on the bias voltage and the tip position. Since the spectrum is the superposition of a parabola (see Fig. 4.3 and [70]) and the two dips, which we model as Gaussian curves[2], the ansatz for the spectrum is

$$\Delta f(V_{\mathrm{b}}, \boldsymbol{p}) = \Delta f_{\mathrm{para}}(V_{\mathrm{b}}) + \Delta f^{-}(V_{\mathrm{b}}, \boldsymbol{p}) + \Delta f^{+}(V_{\mathrm{b}}, \boldsymbol{p}) \tag{4.2}$$

with the parabola function

$$\Delta f_{\mathrm{para}}(V_{\mathrm{b}}) = c_1 V_{\mathrm{b}}^2 + c_2 V_{\mathrm{b}} + c_3 \tag{4.3}$$

and the Gaussian curves

$$\Delta f^{-}(V_{\mathrm{b}}, \boldsymbol{p}) = d^{-} \cdot \exp\left(-\left(\frac{V_{\mathrm{b}} - V^{-}(\boldsymbol{p})}{w^{-}}\right)^2\right) \tag{4.4a}$$

$$\Delta f^{+}(V_{\mathrm{b}}, \boldsymbol{p}) = d^{+} \cdot \exp\left(-g\left(\frac{V_{\mathrm{b}} - V^{+}(\boldsymbol{p})}{w^{+}}\right)\right) \tag{4.4b}$$

for the dips, where $d^{\mp}, V^{\mp}(\boldsymbol{p}), w^{\mp}$ are the respective depth, position, and width of the dips. The depth and width of the dips are nearly constant while scanning the sample, whereas the dips' position $V^{\mp}(\boldsymbol{p})$ depend on the tip position $\boldsymbol{p}$ and has to be provided via reference data. The function $g(\cdot)$ in (4.4b) is considered as a polynomial $g(x) = a_1 x^2 + a_2 x^4 + a_3 x^6$ to account for the slightly deformed shape of the positive dip. The parameters in (4.3) and (4.4) are then fitted using experimental data. The resulting fit in Fig. 4.7 and Table 4.1 shows that the proposed ansatz is well suited to model the experimentally acquired Δf spectrum.

To simulate the changing tip position, the experimental reference data of Fig. 4.5 for $V^{\mp}(\boldsymbol{p})$ for every pixel is fed to (4.4). Thus, at each new pixel in the simulation, the dips are shifted according to the experimental reference. This model is used for

[2]Note that in the actual experiment the shape of the dips is somewhere between a Gaussian curve and a compressed half-circle that depends on the chosen tip oscillation amplitude, the value of $V^{\mp}(\boldsymbol{p})$, and the width of the electronic level of the quantum dot. See also [86].

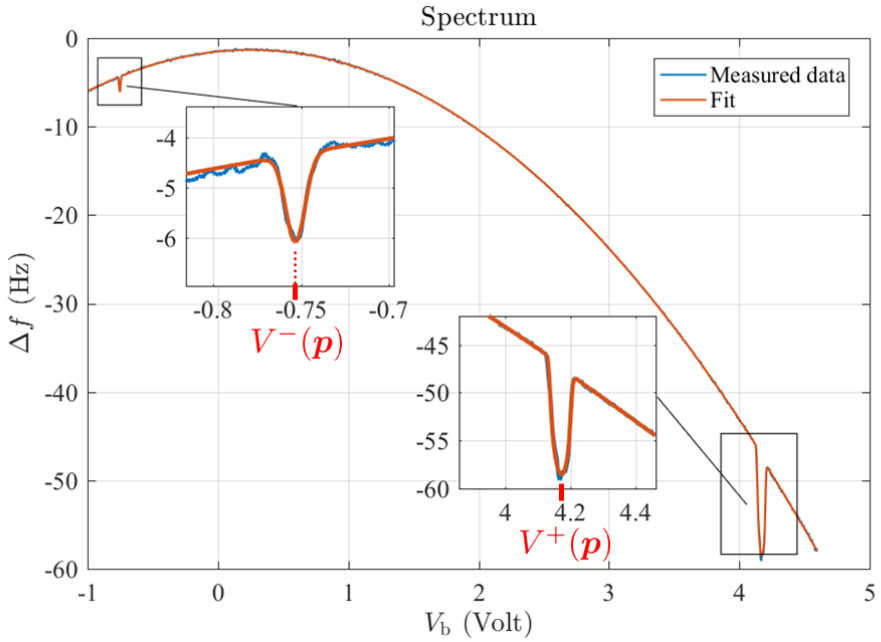


Figure 4.7: Spectrum of Fig. 4.3 with the fitted function (4.2).

Table 4.1: Δf Spectrum Fit Parameters

$c_1 = -1.3$	$a_1 = 0.70$	$d^- = -1.1$	$d^+ = -4.6$
$c_2 = 0.56$	$a_2 = -0.61$	$V^- = -1.3$	$V^+ = 4.3$
$c_3 = -0.76$	$a_3 = 1.64$	$w^- = 0.022$	$w^+ = 0.087$

the simulation results presented in Section 4.4.

4.3 Two-degree-of-freedom Control Framework

Two-degree-of-freedom (2DOF) control is an approach that consists of two parts, a feedback part/controller and a feedforward part/controller [9, 80], illustrated in Fig. 4.8. The objective of the feedback controller is to regulate the error between a reference and the controlled variable to zero. The objective of the feedforward part is to "help" the feedback part by increasing the regulation performance. To this end, the feedforward part often contains a model approximation of the controlled process, which allows computing open-loop inputs that assist the regulation when, for instance, the reference changes.

Two-degree-of-freedom control approaches are well known in scanning probe techniques like scanning tunneling microscopy (STM) or atomic force microscopy (AFM). There, the objective is to control the piezo stages that govern the movement of the microscope tip in x, y, and z-direction to achieve fast scanning. The z-piezo is controlled according to the topography feedback signal (e.g. tunneling current in STM or cantilever deflection in contact mode AFM [165, 166]), whereas the piezos in x and

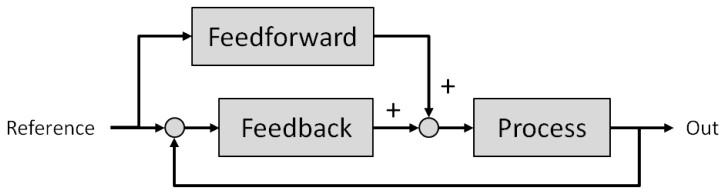


Figure 4.8: General two-degree-of-freedom control structure.

y-direction are controlled such that the tip follows specific reference trajectories in the (x, y)-plane (that implement the raster scanning pattern) as close and as fast as possible. For the main scanning direction, this is usually a triangular signal [149]. The controllers are often based on models of the respective piezo stages. The feedforward part of the 2DOF controllers is therefore also usually model-based and techniques like H_∞, l_1-optimal, model inversion, or iterative learning control are employed. For overviews on this topic see [37, 44, 71, 100, 149, 204].

Although scanning quantum dot microscopy is based on AFM, the considered control problem is different from the one in AFM. First, notice that electrostatic potential images are generated via SQDM at a constant height z. Thus, control of the z-piezo is not necessary. Furthermore, the controllers of commercially available NC-AFMs (used for SQDM) are, up to now, sufficient to deal with the occurring scan speeds. One of the reasons for this is the fact that the bottleneck in achieving high image generation rates with SQDM is not the task of tracking reference trajectories for the microscope tip but tracking unknown and changing electric parameters (the $V^{\mp}$ values) via the quantum dot.

As detailed in Section 4.2.1, the effective electrostatic potential of the sample at the position of the tip $\Phi^*(\boldsymbol{p})$ changes in the three dimensional space and to compute it we need, according to (4.1), measurements of the negative and positive dip positions $V^{\mp}(\boldsymbol{p})$ that depend themselves on the position of the microscope tip. Therefore, from a control point of view, SQDM with the changing $V^{\mp}(\boldsymbol{p})$ values can be regarded as a parameter varying system and the objective of this chapter is to design a control framework that automatically determines and tracks the unknown parameters $V^{\mp}(\boldsymbol{p})$ while scanning the sample. The better a potential controller can track the $V^{\mp}(\boldsymbol{p})$ values, the faster the sample can be scanned.

As outlined, $V^{\mp}(\boldsymbol{p})$ cannot be measured directly and a model of the respective dynamics is unavailable because it depends, besides the scan speed, directly on the electrostatic potential, which itself is unknown. Thus, a potential control approach should be majorly model-independent. It furthermore has to adapt V_b indirectly, based on a quantity that can be measured, in this case, the frequency change Δf. Hence, we are looking for a control law of the form $V_\mathrm{b} = \kappa(\Delta f)$.

In addition, we want to exploit the repetitive nature of the line by line raster scan pattern to improve control performance. To this end, we present in the following a two-degree-of-freedom control approach, consisting of a feedback and a feedforward part

(Fig. 4.9). For both parts we develop two different approaches, leading to different versions of the 2DOF controller.

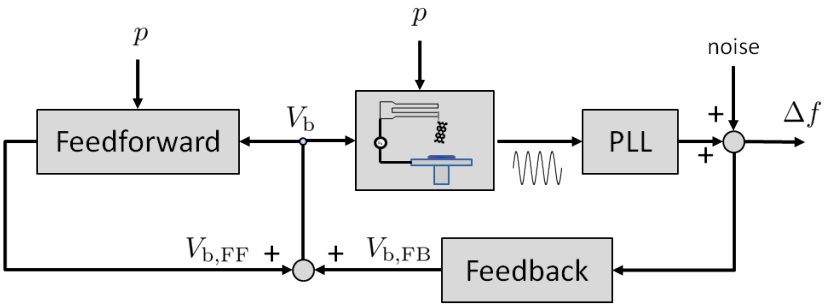


Figure 4.9: Proposed 2DOF control structure: The bias voltage V_b is the sum of the feedback and feedforward output. The feedforward part is influenced by the current tip position $\boldsymbol{p}$.

The first feedback controller is an extremum seeking controller and the second controller is called slope tracking controller. The central idea to both controllers is, instead of measuring the dip spectrum (time consuming, contains a lot of unnecessary data), to track one specific reference point in each dip. One major difference between the extremum seeking and the slope tracking controller is this reference point. While the extremum seeking controller tracks directly $V^{\mp}(\boldsymbol{p})$, the slope tracking controller tracks a point on the dip's slope (thus the name).

The objective of the feedforward part is to provide a prediction for the course of $V^{\mp}(\boldsymbol{p})$ for the next line. With a good prediction, the feedback controller is already close to the true value of the reference point, and the better the prediction, the less the feedback controller has to correct. This leads to a more accurate tracking of $V^{\mp}(\boldsymbol{p})$, which in turn allows to increase the scanning speed. In addition, if the V_b value leaves the interval of the respective dip, it yields a Δf value that is located on the parabola part of the spectrum (Fig. 4.3). This can lead to the V_b value further diverging from the dip. As we will discuss in the next subsection, the feedforward signal is also critical to prevent this situation (or at least significantly lower the probability of this happening). The presented feedforward approaches both leverage the repetitive nature of scanning quantum dot microscopy. In particular we present (i), an approach that merely uses the last scanned line as a prediction for the next and (ii), an approach that utilizes a Gaussian process to generate a model of $V^{\mp}(\boldsymbol{p})$ online and from that a prediction for the next line to be scanned. The individual 2DOF parts will be discussed in more detail in the remainder of this section.

4.3.1 Feedback Control

We develop two different feedback controllers, namely an extremum seeking controller (Section 4.3.1.1) and a so-called slope tracking controller (Section 4.3.1.2). Afterwards, we investigate the influence of the shape and size of the dips on the feedback controllers'

performance (Section 4.3.1.3). This already yields some specific recommendations for practitioners of scanning quantum dot microscopy.

4.3.1.1 Extremum Seeking Controller

The fact that the dips' positions $V^{\mp}(\boldsymbol{p})$ characterize the minima of the local convex functions $\Delta f^{\mp}$ satisfying

$$V^{-}(\boldsymbol{p}) = \underset{V_\mathrm{b}}{\operatorname{argmin}}\, \Delta f^{-}(V_\mathrm{b}, \boldsymbol{p}) \tag{4.5a}$$

$$V^{+}(\boldsymbol{p}) = \underset{V_\mathrm{b}}{\operatorname{argmin}}\, \Delta f^{+}(V_\mathrm{b}, \boldsymbol{p}), \tag{4.5b}$$

can be exploited for the determination of $V^{\mp}(\boldsymbol{p})$. Since Δf is measured online by the phase-locked loop, the minima of $\Delta f^{\mp}$ can be determined continuously by appropriately adapting the input V_b in (4.5a) and (4.5b) during the scanning process.

This can be achieved by employing methods of extremum seeking control [10, 207]. As the name indicates, these methods are designed to find the extremum, i.e., a minimum or a maximum, of the output/a function of a given system. The core principle includes a seeking element that continuously samples the output signal at the current operating point to obtain some kind of direction or gradient information. If a measure of the gradient is available, the current operating point is changed accordingly in the negative gradient direction. If the optimum is reached, the gradient is zero and the operating point is not changed anymore. Thus, extremum seeking approaches are closely related to optimization and are therefore also sometimes called *real-time optimization* methods [10]. In principle, an extremum seeking controller can be realized in different ways, though they have to satisfy a variety of additional properties, such as, for instance, low computational load or the ability to deal with constraints.

Works on extremum seeking date back to as early as 1922 [101] or 1951 [47]. Though many other works were published in the second half of the 20th century, e.g. [72], it wasn't until 2000 that a rigorous stability analysis of extremum seeking control was developed [96]. Since then, interest has sparked again and found its realizations in applications such as anti-lock-breaking systems [205, 206], maximum-power-point-tracking [29, 105], source seeking [38, 206], beam control in particle accelerators [164], or the control of plasma in a Tokamak reactor [34]. Extensions for discrete time systems [36], for multivariable systems [162], systems with partial model information [3, 72], and further results on stability [164, 182] exist. Generalizations of the extremum seeking control scheme such as a unifying framework [139] and with optimization approaches, such as Newton like extremum seeking [60, 133], stochastic [109, 110], and a non-gradient approach [140] have been developed too. For extensive lists of further publications see [164, 181, 207].

We employ an adapted version of the approach of [96] as shown in Fig. 4.10, where we deal with a local convex function $h(u)$ for which there exists a minimum at $u = u^*$.

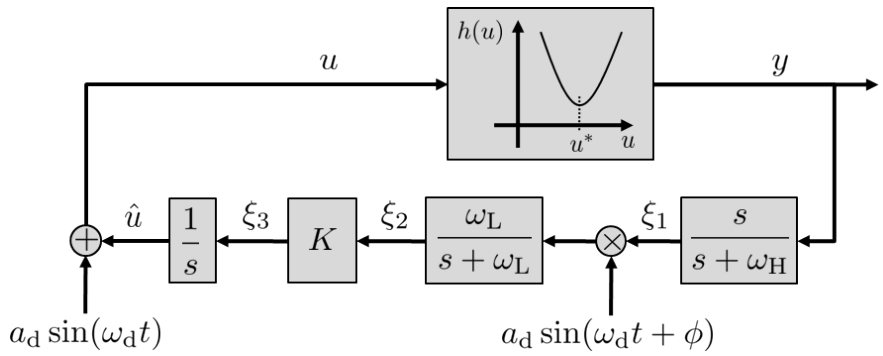


Figure 4.10: Block diagram of the extremum seeking control approach.

At this minimum we naturally have $\frac{\mathrm{d}h}{\mathrm{d}u}\big|_{u^*} = 0$. To search for u^*, a dither signal $d(t) = a_d \sin(\omega_d t)$ is used to perturb the current operating point $\hat{u}$, i.e., $u(t) = \hat{u} + d(t)$. The resulting signal $y(t) = h(u(t))$ is passed through a high pass filter $G_{\mathrm{H}}(s) = \frac{s}{s+\omega_{\mathrm{H}}}$ to eliminate any constant offsets. The filtered signal $\xi_1(t)$ is then multiplied by the phase shifted dither signal[3] and low pass filtered with $G_{\mathrm{L}}(s) = \frac{\omega_{\mathrm{L}}}{s+\omega_{\mathrm{L}}}$. This leads to $\xi_2(t)$ and it can be shown (see Appendix A.5) that

$$\lim_{t\to\infty} \xi_2(t) = \frac{a_d^2}{2} \frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}}$$

at the current operating point $\hat{u}$. Choosing $K := {}^{-2}/_{a_d^2}$ in Fig. 4.10, one obtains

$$\lim_{t\to\infty} \xi_3(t) = -\frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} ,$$

i.e. $\xi_3(t)$ tends to the negative gradient of $h(\hat{u})$. Hence, $\xi_3(t)$ can be used subsequently to implement a gradient descent approach, realized by the integration of $\xi_3(t)$, turning $\hat{u}$ into an estimation of the minimizer u^*.

In SQDM we have $h(u) = \Delta f(V_{\mathrm{b}})$, which is locally convex in the dips Δf^- and Δf^+. Then, the extremum seeking controller computes the derivative $\Delta f' = \frac{\mathrm{d}\Delta f}{\mathrm{d}V_{\mathrm{b}}}$ by modulating the dither signal $d(t)$ onto the V_{b} signal. Since $\Delta f' = 0$ characterizes the dips' minima, it also characterizes exactly the value of $V^{\mp}(\boldsymbol{p})$. Thus, if a potential controller regulates $\Delta f'$ to zero, it automatically yields $V_{\mathrm{b}} = V^{\mp}(\boldsymbol{p})$.[4] Therefore, $\Delta f' = 0$ can be interpreted as the control reference and the gradient descent is achieved by using an integral controller

$$V_{\mathrm{b,FB}}(t) = K_{\mathrm{ESC}} \int_0^t e_d(\tau)\, \mathrm{d}\tau$$

that minimizes the error $e_d(t) = \Delta f'_{\mathrm{ref}} - \Delta f'(t)$ with $\Delta f'_{\mathrm{ref}} = 0$ and $K_{\mathrm{ESC}} > 0$. The

[3]The phase of $\xi_1(t)$ is shifted by the high pass filter by ϕ. In order to be in phase, the dither signal that is multiplied with $\xi_1(t)$ is shifted also by ϕ.

[4]Note that only one dip can be tracked at a time.

voltage applied to the AFM tip is

$$\begin{aligned} V_{\mathrm{b,mod}}(t) &= V_{\mathrm{b,FB}}(t) + V_{\mathrm{b,FF}}(t) + d(t) \\ &= V_{\mathrm{b}}(t) + d(t) \ , \end{aligned}$$

where $V_{\mathrm{b,FF}}$ is the feedforward signal computed as detailed in Section 4.3.2. Note that the signal $V_{\mathrm{b}} = V_{\mathrm{b,FB}} + V_{\mathrm{b,FF}}$ without the dither signal is used for image generation. The corresponding block diagram is depicted in Fig. 4.11.

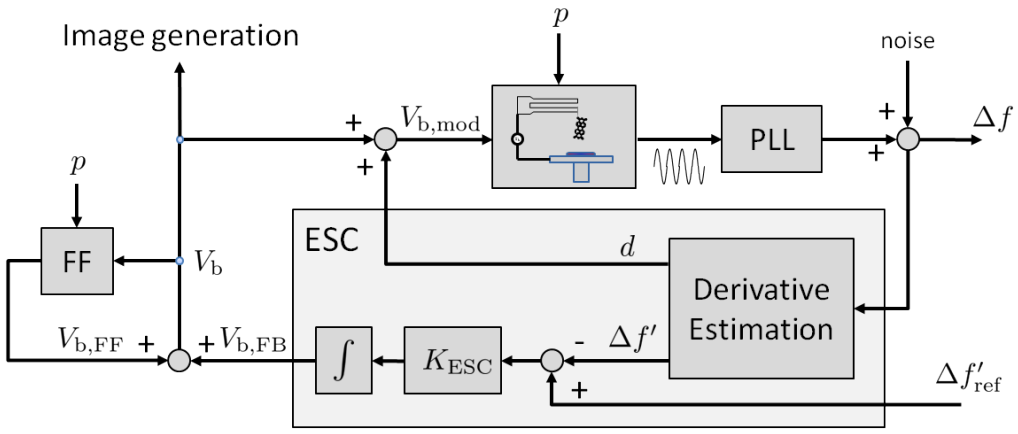


Figure 4.11: Block diagram of the closed-loop with the ESC.

The extremum seeking controller parameters that need to be chosen are the dither signal amplitude a_d and frequency ω_d, the low and high pass filter cut-off frequencies ω_{L} and ω_{H}, and the control gain K_{ESC}. In the following, we present some general guidelines for choosing these parameters, based on the characteristics of the respective dip and the phase-locked loop.

Dither Signal The interval in which the applied bias voltage varies locally is $[V_{\mathrm{b}} - a_d, V_{\mathrm{b}} + a_d]$. The gradient of the dip $\frac{\mathrm{d}\Delta f}{\mathrm{d}V_{\mathrm{b}}}$ is then approximated within this interval, i.e., an average gradient around the current bias voltage V_{b} is computed. A natural upper bound for the dither signal amplitude a_d is therefore the dip width $w^{\mp}$ itself. On the other hand, the smaller a_d, the more accurate the gradient at the current V_{b}. Thus, we get

$$a_d \leq w^{\mp} \ ,$$

where a_d should be chosen as small as possible. However, the Δf measurements for the gradient computation are corrupted by noise, which poses a lower limit on a_d.

Regarding the choice of the dither signal frequency ω_d: the higher the frequency, the faster the gradient estimate converges but also the higher the variance of the estimate. Additionally, if ω_d is much larger than the system's bandwidth, the dither signal is damped and shifted significantly and in consequence, the gradient computation deteriorates. Hence, an upper bound depends on the maximum bandwidth of the

system dynamics. In SQDM, the limiting element is the phase-locked loop with its bandwidth ω_{PLL}. We have found

$$2\omega_{\mathrm{PLL}} \leq \omega_d \leq 10\omega_{\mathrm{PLL}}$$

to work well, where larger values decrease convergence time but increase variance.

Filter Cut-Off Frequencies The objective of the high pass filter is to remove the constant offset of the Δf signal. This can be achieved sufficiently fast for

$$\omega_{\mathrm{H}} \geq 0.5\omega_d \ .$$

The objective of the low pass filter is to smoothen the signal ξ_2. The smaller ω_{L} the stronger the smoothing. The stronger the smoothing, the less oscillatory the gradient approximation becomes but also the slower the convergence. Hence, choosing ω_{L} is a trade-off between convergence speed and variance, similar to ω_d. We have found that

$$0.1\omega_d \leq \omega_{\mathrm{L}} \leq 0.5\omega_d$$

works well. Accordingly, ω_{L} can be chosen within this interval, depending on the objective of fast convergence (fast scanning) or small variance.

Phase Shift The phase-locked loop and the high pass filter introduce an additional phase shift

$$\phi = \arg\Big(G_{\mathrm{PLL}}(\mathrm{j}\omega_d)G_{\mathrm{H}}(\mathrm{j}\omega_d)\Big)$$

w.r.t. the dither signal $d(t)$. This can be accounted for by adding the same phase shift ϕ to the dither signal that is multiplied with the high pass outcoming signal (see Fig. 4.10).

Control Gain To facilitate gain tuning, we define K_{ESC} via

$$K_{\mathrm{ESC}} = \frac{k_{\mathrm{ESC}}}{|G_{\mathrm{PLL}}(\mathrm{j}\omega_d)G_{\mathrm{H}}(\mathrm{j}\omega_d)|} \ , \tag{4.6}$$

where $k_{\mathrm{ESC}} > 0$ is the new tunable extremum seeking controller gain. This redefinition automatically compensates the amplitude change of $\xi_3(t)$ introduced by the PLL and the high pass filter. This way the PLL and the high pass filter can be changed without influencing the amplitude of $\xi_3(t)$. Note that the low pass filter $G_{\mathrm{L}}(\mathrm{j}\omega_d)$ is not included in (4.6) because then one loses the possibility of adjusting the convergence of $\xi_3(t)$ independently of that of $\hat{u}(t)$.

The larger k_{ESC}, the faster the convergence to the minimum but also the more oscillatory the estimated minimizer $\hat{u}(t) = V_{\mathrm{b}}(t)$. Hence, the larger k_{ESC}, the faster we can scan the sample but at the cost of less accurate tracking. In particular, oscillations

due to high k_{ESC} values eventually appear in the final image as noise. Furthermore, if the oscillations in V_{b} become too large, the dip might be lost. For instance, when the negative dip is tracked the oscillations might cause V_{b} to leave the dip towards the left side of the dip. In that case, the gradient descent then leads the controller to further decrease the V_{b} value down the parabola, making it impossible to recover the dip.

4.3.1.2 Slope Tracking Controller

The second feedback controller is based on the idea to track a point on the dips' slope (Fig. 4.12), instead of tracking the dips' minima. The resulting Δf value is then set as the reference value Δf_{ref} for the whole sample and the deviations $e_f(t) = \Delta f_{\mathrm{ref}} - \Delta f(t)$ are used directly as an error to the integral controller

$$V_{\mathrm{b,FB}}(t) = K_{\mathrm{STC}} \int_0^t e_f(\tau)\,\mathrm{d}\tau \tag{4.7}$$

that adapts V_{b} accordingly. The resulting voltage applied to the AFM tip is

$$V_{\mathrm{b}}(t) = V_{\mathrm{b,FB}}(t) + V_{\mathrm{b,FF}}(t)$$

with $V_{\mathrm{b,FF}}$ computed as detailed in Section 4.3.2. The resulting block diagram is depicted in Fig. 4.13.

The slope tracking controller parameters that need to be chosen are the exact position of the reference point Δf_{ref} and the control gain K_{STC}.

Reference Point The choice of the slope tracking controller reference point Δf_{ref} is very important, in particular, because most of the Δf values of a dip appear three times in the local spectrum around the respective dip (see Fig. 4.12). We choose Δf_{ref} to lie on the inner slope of the dips (i.e. on the right slope of the negative dip and on the left slope of the positive dip) and detail the reasons in the following.

First, note that the inner slope is always larger than the outer slope (compare Fig. 4.3 and Fig. 4.12), which suggests better control performance on the inner slope. Second, if the reference point is located on the inner slope, according to (4.7) the controller is able to drive V_{b} back to Δf_{ref} even if V_{b} has left the dips towards the vertex of the parabola (the sign of e_f does not change until the vertex), although convergence to Δf_{ref} would not be as fast as within the dip due to the relatively flat slope of the parabola. This property is lost if the reference point is chosen to lie on the outer slope and V_{b} had left the dips towards the part of the parabola where values decrease indefinitely. In that case, the controller would drive the bias voltage to even larger absolute values until the corresponding Δf_{ref} value on the parabola is reached (crossing of the dashed black line and the blue parabola on the left side in Fig. 4.12). In general, the slope tracking controller won't be able to recover the dip in that case.

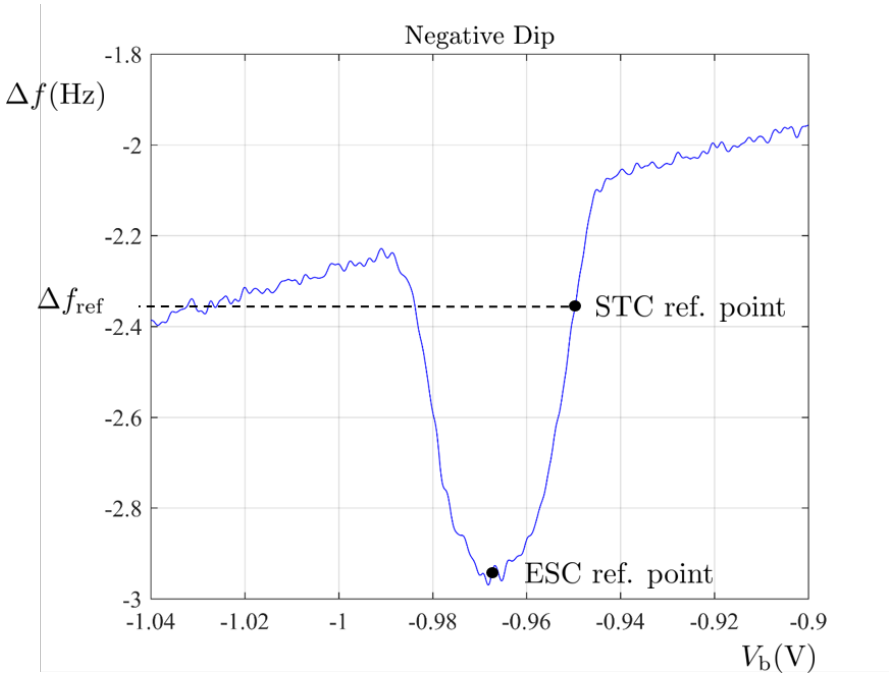


Figure 4.12: Measured spectrum of the negative dip with controller reference points for extremum seeking and slope tracking controller. The dashed black line indicates that the Δf_{ref} point of the slope tracking controller has three crossings with the blue spectrum. Hence, three possible equilibria of the closed-loop system exist for the slope tracking controller. The one on the dip's inner slope (on the right-hand side for the negative dip) is the point we want to track. In case of the positive dip, everything is mirrored on the vertical axis.

Regarding the exact position, Δf_{ref} should lie relatively far away from the dip minimum at $V^{\mp}(\boldsymbol{p})$ because this is another critical point for the slope tracking controller. If V_{b} moves over this point (e.g. $V_{\text{b}} < V^{-}(\boldsymbol{p})$ for the negative dip) the control error e_f becomes smaller instead of larger, which automatically has a deteriorating effect on the control performance and increases the probability that the dip is left towards the part of the parabola where values decrease indefinitely.

Good starting points for the reference point are $\Delta f_{\text{ref}} := \Delta f(V_0^{-} + w^{-})$ for the negative dip and $\Delta f_{\text{ref}} := \Delta f(V_0^{+} - w^{+})$ for the positive dip.

Controller Gain As the reference point is chosen to lie on the inner slope of the two dips and the corresponding slopes' gradients have different sign, we need $K_{\text{STC}} > 0$ for the negative dip and $K_{\text{STC}} < 0$ for the positive dip. As for the extremum seeking controller, the larger the absolute value of K_{STC} the faster the convergence to Δf_{ref} but also the more oscillatory. Thus, the larger the absolute value, the faster the sample can be scanned but at the cost of less accurate tracking.

Systematic Error The slope tracking controller does not require the computation of the derivative and is therefore, in principle, faster, i.e., the resulting control error

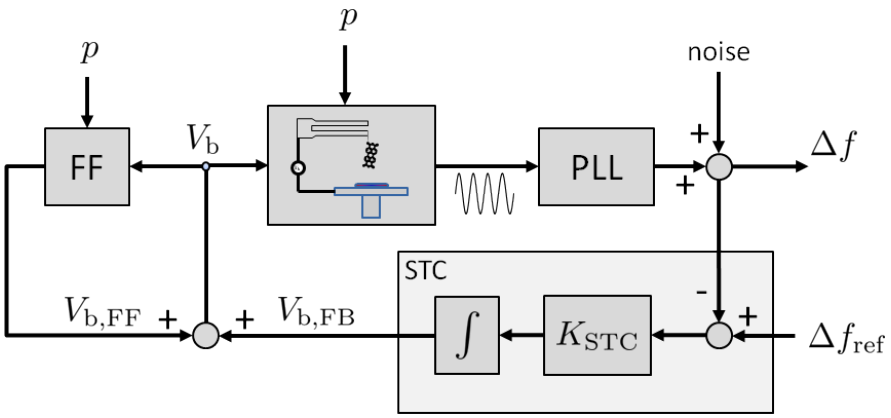


Figure 4.13: Block diagram of the closed-loop with the slope tracking controller.

decays faster. However, it introduces a systematic error $e_\mathrm{STC} := V^{\mp}(\boldsymbol{p}) - V_\mathrm{b}(\Delta f_\mathrm{ref})$ due to the difference between $V^{\mp}(\boldsymbol{p})$ and the V_b value at the slope tracking controller reference Δf_ref (approximately[5] 20 mV in Fig. 4.12). Additionally, this error is not constant and changes while scanning because when the dip changes its position it slides the parabola up- or downwards. For further clarification imagine that the negative dip moves vertically upwards (no horizontal movement). In that case, $V^{-}(\boldsymbol{p})$ does not change but $V_\mathrm{b}(\Delta f_\mathrm{ref})$ changes to more negative values, moving closer to $V^{-}(\boldsymbol{p})$. Hence, e_STC decreases as the dip moves vertically upwards and decreases as the dip moves downwards. This effect also occurs when the dip moves along the parabola because there is always a vertical motion component. Furthermore, the effect is larger for the positive dip because it is typically located at steeper parts of the parabola where the vertical motion is more pronounced.

4.3.1.3 Influence of Dip Shape and Size

We have established a connection between some of the parameters of the feedback controllers, the dips, and the phase-locked loop. Here we also investigate how the performance of the feedback controllers is influenced by the dips' shape and size, which depend on the dips' depth $d^{\mp}$ and width $w^{\mp}$. To this end, and for the moment, we do not consider the tracking but only the set-point regulation problem, i.e., we do not consider the scanning process but let the microscope tip rest at one position in space. Then, the electrostatic potential does not change and the dips' positions are constant.

We illustrate the influence of the dips' depth $d^{\mp}$ via simulations for the negative dip (Fig. 4.14) and positive dip (Fig. 4.15) and observe that the deeper the dip, the better for control as the settling times decrease with larger dip depths. We observe that, as explained in Section 4.3.1.2, the slope tracking controller is generally a bit faster than the extremum seeking controller because it does not need to estimate the gradient.

[5]Since the positive dip is wider, this error is larger for the positive dip.

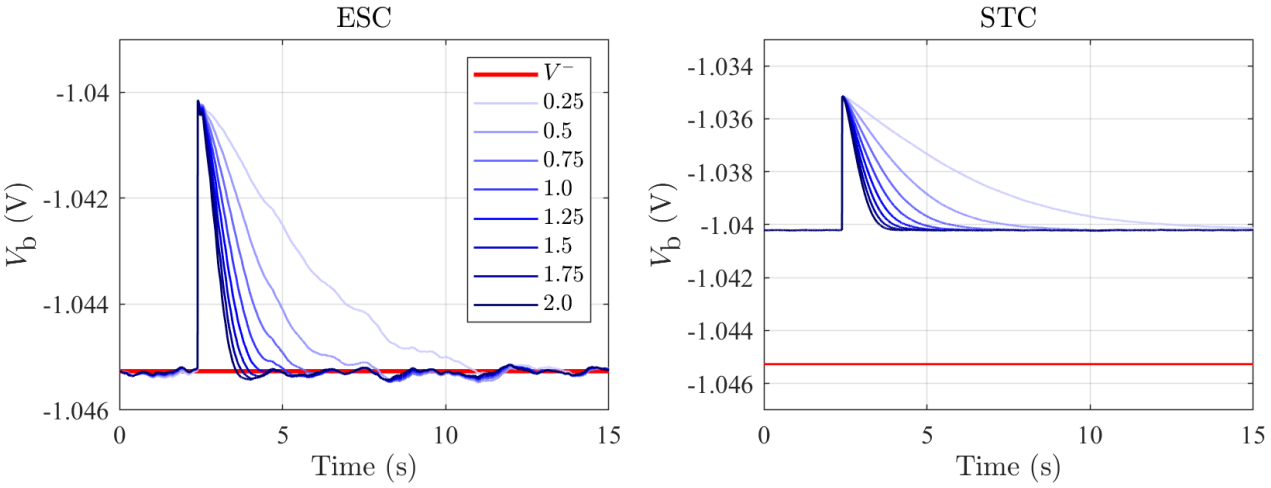


Figure 4.14: Influence of the dip depth: Extremum seeking (left) and slope tracking (right) simulations with $k_{\mathrm{ESC}} = 10^{-5}$ and $K_{\mathrm{STC}} = 0.005$. The simulations were performed for the negative dip and for different dip depths $m \cdot d^-$ with the scaling factors m as indicated in the legend. At $t = 2.5\,\mathrm{s}$ the current bias voltage V_b was shifted by d^- to the right of the dip (i.e. to less negative values) such that the controllers had to regain their reference point.

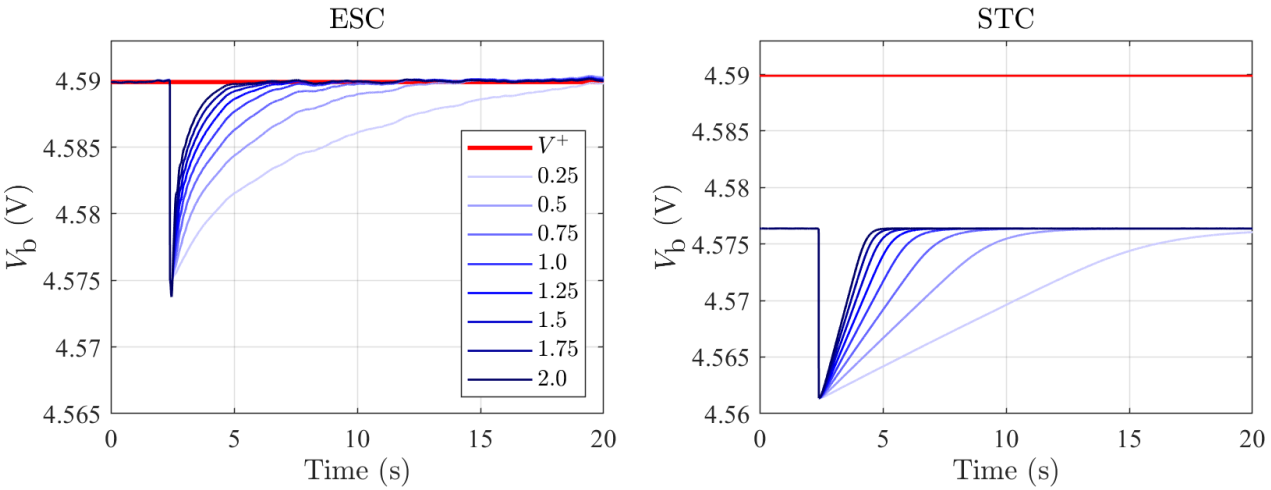


Figure 4.15: Influence of the dip depth: Extremum seeking (left) and slope tracking (right) simulations with $k_{\mathrm{ESC}} = 10^{-5}$ and $K_{\mathrm{STC}} = -0.001$. The simulations were performed for the positive dip and for different dip depths $m \cdot d^+$ with the scaling factors m as indicated in the legend. At $t = 2.5\,\mathrm{s}$ the current bias voltage V_b was shifted by d^+ to the left of the dip (i.e. to less positive values) such that the controllers had to regain their reference point.

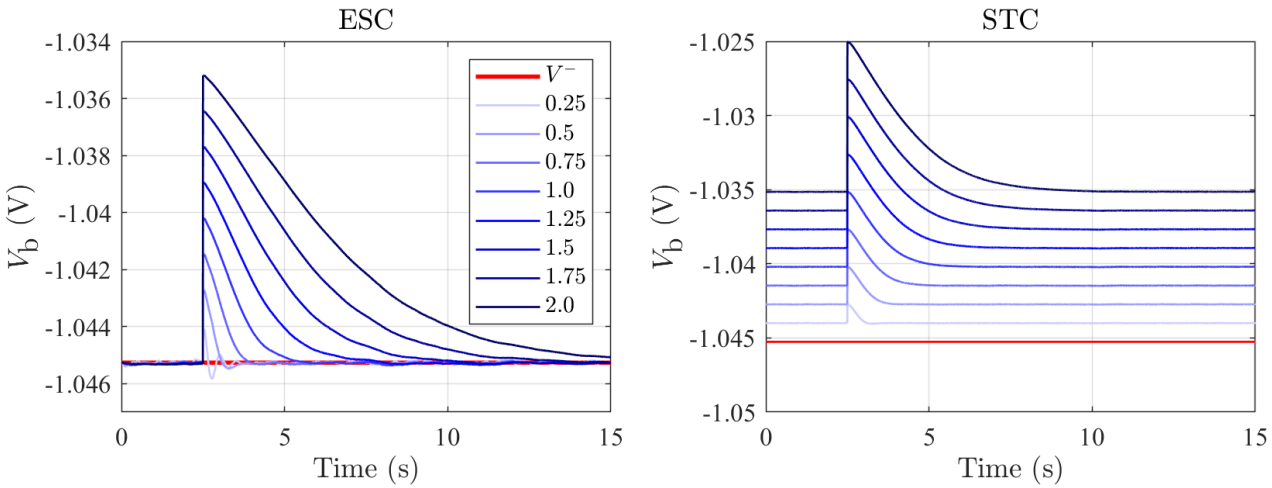


Figure 4.16: Influence of the dip width: Extremum seeking (left) and slope tracking (right) simulations with $k_{\mathrm{ESC}} = 10^{-5}$ and $K_{\mathrm{STC}} = 0.005$. The simulations were performed for the negative dip and for different dip widths $m \cdot w^-$ with the scaling factors m as indicated in the legend. At $t = 2.5\,\mathrm{s}$ the current bias voltage V_b was shifted by d^- to the right of the dip (i.e. to less negative values) such that the controllers had to regain their reference point.

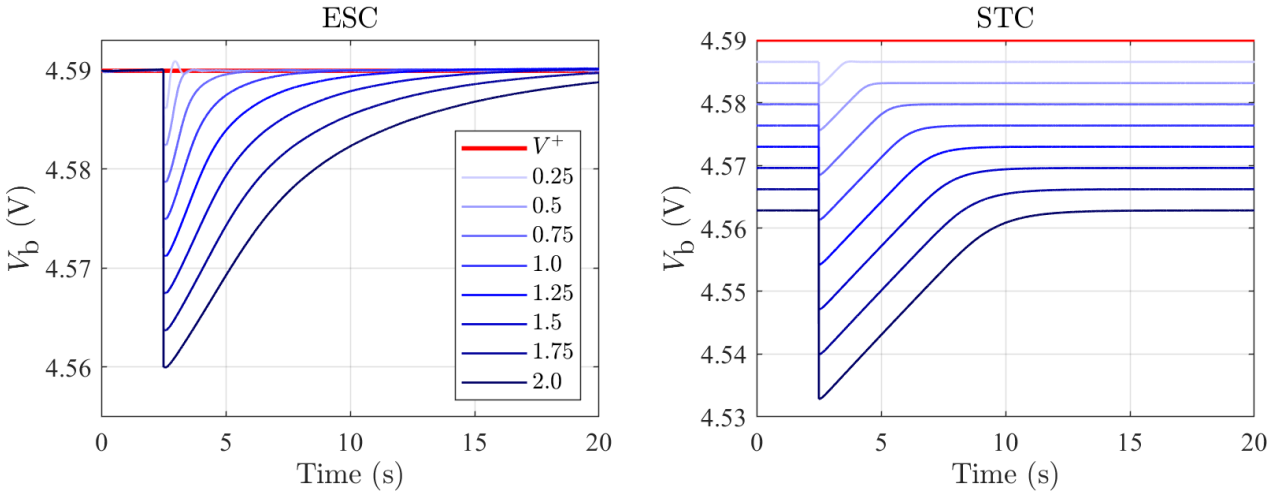


Figure 4.17: Influence of the dip width: Extremum seeking (left) and slope tracking (right) simulations with $k_{\mathrm{ESC}} = 10^{-5}$ and $K_{\mathrm{STC}} = -0.001$. The simulations were performed for the positive dip and for different dip widths $m \cdot w^+$ with the scaling factors m as indicated in the legend. At $t = 2.5\,\mathrm{s}$ the current bias voltage V_b was shifted by d^+ to the left of the dip (i.e. to less positive values) such that the controllers had to regain their reference point.

The drawback is the slope tracking controller's systematic error e_{STC} (a constant offset from the true $V^{\mp}$ value).

In Fig. 4.16 and Fig. 4.17, we illustrate the influence of the dips' width $w^{\mp}$ via simulations for the negative and positive dip, respectively, and observe that the narrower the dip, the better for control as the settling times decrease. In this case, the effect appears to be a bit stronger for the extremum seeking than for the slope tracking controller. Regarding the slope tracking controller, due to the different dip widths and therefore different reference points Δf_{ref}, the error e_{STC} is different for each of the simulations.

Thus, we conclude that sharp (deep and narrow) dips are beneficial for control performance and that practitioners of SQDM should aim for sharp dips whenever possible. This can be achieved, for instance, with small oscillation amplitudes of the microscope's tuning fork [86]. However, such amplitudes increase the noise at the same time, which ultimately imposes a lower bound on the oscillation amplitude. Thus, the selection of the oscillation amplitude is a trade-off.

4.3.2 Feedforward Control

The main objective is to track the dips' positions $V^{\mp}(\boldsymbol{p})$, which is achieved by formulating and solving surrogate control problems. In case of the extremum seeking controller, $V^{\mp}(\boldsymbol{p})$ is tracked by regulating V_{b} such that the gradient reference $\Delta f'_{\mathrm{ref}} = 0$ is attained. In case of the slope tracking controller, $V^{\mp}(\boldsymbol{p})$ is approximately tracked by regulating V_{b} such that a reference Δf_{ref} on the dips' inner slope is attained. Now, while scanning the sample the tip position $\boldsymbol{p}$ changes and with it, the dips move along the parabola. This changes the respective controlled variables, i.e., $\Delta f'$ (extremum seeking) or Δf (slope tracking) and thus, the influence of a changing tip position $\boldsymbol{p}$ can be considered as a disturbance to the extremum seeking and slope tracking controller. In principle, it is possible that the dips change their position faster than the employed controller can adapt the bias voltage V_{b}, which eventually leads to the controller "losing the dip". This is caused by a combination of a rapid change of the electrostatic potential, a high scan speed, and the controller dynamics. In that case, the scanning process has often to be aborted and restarted from the beginning.

The risk of losing the dip can be substantially reduced by the generation of an appropriate feedforward signal $V_{\mathrm{b,FF}}(\boldsymbol{p})$ that provides a prediction of the disturbance as a function of the tip position $\boldsymbol{p}$. With a suitable disturbance prediction, the initial value for each controller at each tip position stays always within the dips' interval and is closer to $\Delta f'_{\mathrm{ref}}$ (extremum seeking) or Δf_{ref} (slope tracking) than without the feedforward signal. In that way, the feedforward has not only the potential for increased performance, and with that eventually higher scan speeds, but is even essential for correct operation of SQDM.

In the following, we present two approaches to generate such a feedforward signal.

The first approach (Section 4.3.2.1) takes the last scanned line, filters it, and applies it as the feedforward signal. The second approach (Section 4.3.2.2) employs a Gaussian process to model the respective $V^{\mp}(\boldsymbol{p})$ maps online and generate an adequate feedforward signal prediction from these models. Whereas the first approach is conceptually simple, the second approach yields a feedforward signal that is closer to the true values, though at the expense of increased complexity and computational costs.

Depending if the feedforward signals are adjusted online or only after each scanned row/or a series of time points, the presented feedforward approaches are related to the concept of iterative learning control [28, 200], a tracking control approach for repetitively operated systems. It uses information from previous iterations to reduce the tracking error in the next iteration [74]. Iterative learning control has been applied to scanning probe microscopy techniques like AFM [99], where it was used, e.g., to improve the accuracy of the microscope tip trajectory and thereby increase the scanning speed [51]. In SQDM however, we do not need the feedforward signal to improve the trip trajectory but to predict the unknown parameters $V^{\mp}(\boldsymbol{p})$ that depend on the sample and tip position.

4.3.2.1 Last Line Feedforward Signal

The electrostatic potential Φ_s is smooth, as well as the maps $V^{\mp}(\boldsymbol{p})$. Thus, the $V^{\mp}(\boldsymbol{p})$ values do not change too much from one line to the next and a straightforward choice is therefore to use the V_b values of the previously scanned line to generate a prediction for the next line. Fig. 4.18 shows a block diagram of the feedforward signal generator that stores the last scanned line in a buffer and filters the data (to cancel measurement noise and obtain a smooth output signal) before it is output as the feedforward signal for the next line. The individual elements and features are detailed in the following list.

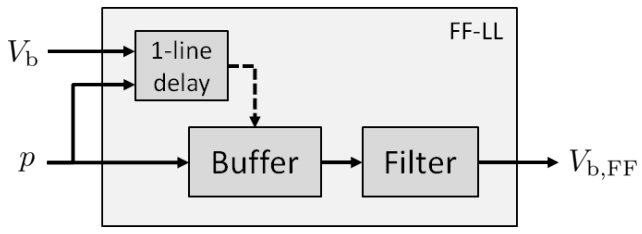


Figure 4.18: Block diagram of the last line feedforward. Current V_b values are indexed with corresponding $\boldsymbol{p}$ values, delayed by one line, stored in a buffer, and output through a mean filter at the next line as the feedforward signal $V_\mathrm{b,FF}$.

- **Buffer:** The $V_\mathrm{b}(\boldsymbol{p})$ values with $\boldsymbol{p} = (x, y)$ of the currently scanned line y (see the raster scan pattern in Fig. 4.2) are stored in a buffer alongside the indexing x values within the line[6]. At the same time, already stored $V_\mathrm{b}(x, y-1)$ values of

[6]Note that x is the fast scan direction.

the previously scanned line $y-1$ are used as a basis for the feedforward signal $V_{\mathrm{b,FF}}(x,y)$ of the current line y.

- **Filter:** The measured and buffered V_b values are corrupted by noise and small ripples (caused by the controllers) that have a deteriorating effect on the control performance, especially in regions where the electrostatic potential is relatively flat. Therefore, while scanning the current line y, the previously measured $V_\mathrm{b}(x,y-1)$ values are smoothed to generate the feedforward signal $V_{\mathrm{b,FF}}(x,y)$ using the mean filter

$$V_{\mathrm{b,FF}}(x,y) = \frac{1}{n}\sum_{i=1}^{n} V_\mathrm{b}\Big(x-\frac{n}{2}+i, y-1\Big)$$

  with filter window length n.

- **Scan speed adaptation:** The previous line $y-1$ is indexed using the measured x-position values. In the current line scan, the active x-position is determined and used for picking the right reference value. This allows for varying scan speed within a line.

4.3.2.2 Gaussian Process Feedforward Signal

As can be seen from Fig. 4.19, the $V^{\mp}(\boldsymbol{p})$ values can vary rapidly in certain regions of the sample. In these regions, using the last scanned line as a prediction for the next line may not be sufficient. For this reason, we propose another approach that generates a model of the $V^{\mp}(\boldsymbol{p})$ maps, which are continuously updated during operation and generate after each line scan a prediction for the upcoming line.

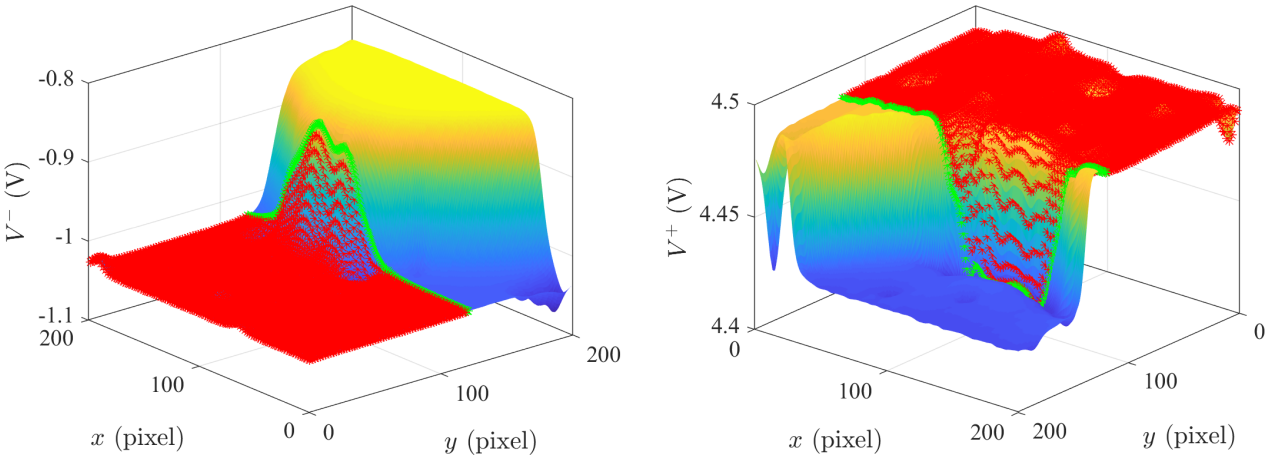


Figure 4.19: 3D representation of the $V^{\mp}(\boldsymbol{p})$ maps of Fig. 4.5. The $V^{\mp}(\boldsymbol{p})$ values are drawn onto the vertical axis to illustrate rapid changes in voltage values. Red points indicate lines that have already been scanned. The green lines indicate the next line that has to be scanned and for which a prediction is required. Note that each map corresponds to a scan at a constant height z.

The dip position maps $V^{\mp}(\boldsymbol{p}) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ are general static and smooth maps, which are a priori unknown and contain different features and large variations as illustrated in Fig. 4.19. In order to generate suitable models, we propose to use Gaussian process regression (see learning of static maps, Section 2.2.1), which then provide the required feedforward signal. Using a suitable kernel (e.g. the squared exponential covariance kernel), a Gaussian process can model any possible smooth map and is not bound to special functions, such as linear functions or polynomials of specific orders. For each of the $V^{\mp}(\boldsymbol{p})$ maps we employ one GP that we denote by $\mathcal{GP}_{V^-}$ and $\mathcal{GP}_{V^+}$.

The input/regressor to both $\mathcal{GP}_{V^-}$ and $\mathcal{GP}_{V^+}$ is the tip position $\boldsymbol{p}$ and the output are the respective $V^{\mp}(\boldsymbol{p})$ values. The associated training data sets are $\mathcal{D}^+ = \{\boldsymbol{P}, \boldsymbol{V}^+\}$ and $\mathcal{D}^- = \{\boldsymbol{P}, \boldsymbol{V}^-\}$ with all collected tip position pairs $\boldsymbol{P} = \{(x_i, y_i)\}$ and dip positions $\boldsymbol{V}^+ = \{V_i^+\}$, $\boldsymbol{V}^- = \{V_i^-\}$. As we are not interested in a point prediction but a line prediction we define the line input to the GPs as

$$\bar{\boldsymbol{p}} = \begin{bmatrix} x_1 & x_2 & \cdots & x_{n_l} \\ y & y & \cdots & y \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{n_l \times 2} ,$$

where n_l is the number of pixels in a line and the y value indicates the line that we seek to predict. Then, the predictions/feedforward signals are generated by the posterior mean function (2.6a) of $\mathcal{GP}_{V^-}$ and $\mathcal{GP}_{V^+}$, which become

$$V^+(\bar{\boldsymbol{p}}) = m_+(\bar{\boldsymbol{p}}|\mathcal{D}^+) = m(\bar{\boldsymbol{p}}) + \varrho(\bar{\boldsymbol{p}}, \boldsymbol{P})\boldsymbol{K}^{-1}(\boldsymbol{V}^+ - m(\boldsymbol{P})) \tag{4.8a}$$

$$V^-(\bar{\boldsymbol{p}}) = m_+(\bar{\boldsymbol{p}}|\mathcal{D}^-) = m(\bar{\boldsymbol{p}}) + \varrho(\bar{\boldsymbol{p}}, \boldsymbol{P})\boldsymbol{K}^{-1}(\boldsymbol{V}^- - m(\boldsymbol{P})) \ . \tag{4.8b}$$

A simple block diagram representation of the Gaussian process-based feedforward is shown in Fig. 4.20.

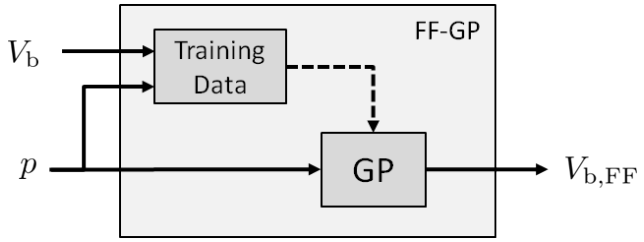


Figure 4.20: Block diagram of the Gaussian process feedforward. Current V_b, together with corresponding $\boldsymbol{p}$ values, are collected as training data for the Gaussian process.

4.4 Evaluation of the 2DOF Control Approaches in Simulations

We evaluate the two-degree-of-freedom control approaches in simulations both qualitatively and quantitatively. We compare the extremum seeking and slope tracking

feedback controllers and illustrate the principle effect and the necessity of a feedforward signal using the last line in Section 4.4.1 and finish with the improvement of the GP-based feedforward signal in Section 4.4.2.

For the simulations we employ the SQDM model of Section 4.2.3 implemented in MATLAB/Simulink and use a fixed-step `ode3` (Bogacki-Shampine) solver with a sampling time of $T_\mathrm{s} = 5\,\mathrm{ms}$. The scanning time for each of the two dip maps $V^-(\boldsymbol{p})$ and $V^+(\boldsymbol{p})$ is $T_\mathrm{scan} = 2\,\mathrm{h}$. To realistically mimic the experiment, each line is furthermore scanned back and forth. This results in approximately 18 s for each single line scan, which in turn equals a scanning speed of approximately 33.3 Å/s with the given area of $600 \times 600\,\mathrm{Å}^2$.

Experimentally acquired and fitted dip spectra (Fig. 4.21) and the $V^\mp(\boldsymbol{p})$ maps of Fig. 4.5 (from [196]) are used as reference data. The cutoff frequency of the phase-locked loop was determined by fitting a step response and was computed as $\omega_\mathrm{PLL} = 10\,\mathrm{s}^{-1}$. The normally distributed white noise has a standard deviation of $\sigma_\mathrm{n}^2 = 0.03\,\mathrm{Hz}$. The parameters of the two controllers are listed in Table 4.2.

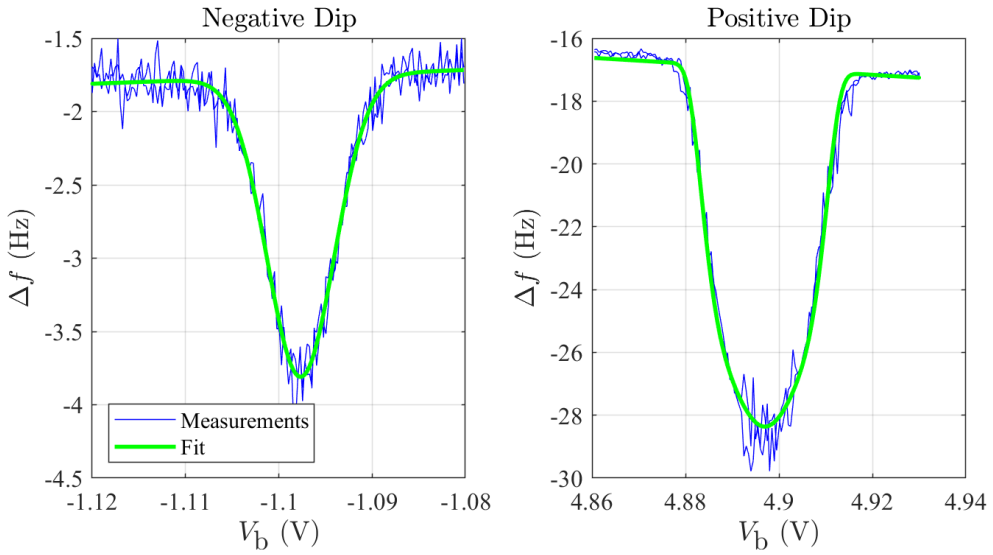


Figure 4.21: Experimentally acquired negative dip (left) and positive dip (right) with model fit. The dips were scanned twice, forward and backward.

4.4.1 2DOF Control with Last Line Feedforward

We now use the experimentally acquired reference data (Fig. 4.5 and Fig. 4.21) to compare the extremum seeking and slope tracking controllers with and without feedforward using the last scanned line. We start with an exemplary time evolution of the involved signals and illustrate the influence of the feedforward. Afterwards we turn our attention to the whole image generation and discuss the final images qualitatively and quantitatively.

Table 4.2: Controller Parameters (if not given otherwise)

Negative Dip	Positive Dip
Extremum seeking controller	Extremum seeking controller
$a_d = 1\,\mathrm{mV}$	$a_d = 1\,\mathrm{mV}$
$\omega_d = 4\omega_{\mathrm{PLL}}$	$\omega_d = 4\omega_{\mathrm{PLL}}$
$\omega_{\mathrm{L}} = 0.2\omega_d$	$\omega_{\mathrm{L}} = 0.2\omega_d$
$\omega_{\mathrm{H}} = 3\omega_d$	$\omega_{\mathrm{H}} = 3\omega_d$
$k_{\mathrm{ESC}} = -5 \cdot 10^{-5}$	$k_{\mathrm{ESC}} = -4 \cdot 10^{-5}$
Slope tracking controller	Slope tracking controller
$\Delta f_{\mathrm{ref}} = \Delta f(V_0^- + 1 \cdot w^-)$	$\Delta f_{\mathrm{ref}} = \Delta f(V_0^+ - 0.9 \cdot w^+)$
$K_{\mathrm{STC}} = 0.04$	$K_{\mathrm{STC}} = -0.003$

4.4.1.1 Time Evolution

In Fig. 4.22 and Fig. 4.23, exemplary time evolutions of the V_{b} signal for the extremum seeking and slope tracking controllers without and with the last line feedforward of Section 4.3.2.1 are shown.

In case of tracking the negative dip $V^-(\boldsymbol{p})$ (Fig. 4.22), at the beginning of the depicted evolution interval, all four controller instances can adequately track the dip, though the slope tracking controller with a constant error because it tracks a point on the dips' slope. As the variations in the electrostatic potential increase, both controllers without the feedforward are unable to fully follow the reference. The extremum seeking controller is unable to fully follow the V^- variations (beginning around $t = 160\,\mathrm{s}$) but still stays close by, whereas the slope tracking controller loses the dip completely around $t = 340\,\mathrm{s}$ because the bias voltage leaves the dip towards the lower part of the parabola (more negative values in this case) as discussed in Section 4.3.1.2. Something similar happens in the case of the positive dip (Fig. 4.23). Again, the extremum seeking controller is unable to fully follow the V^+ variations in the time intervals $[280, 300]\,s$ and $[355, 370]\,s$ but stays close, whereas the slope tracking controller without feedforward completely loses the dip (around $t = 553\,\mathrm{s}$) because the bias voltage leaves the dip towards the lower part of the parabola (more positive values in this case). As one would expect, all these cases occur when the $V^{\mp}$ values are changing rapidly, i.e., the changes are too fast for the feedback controllers to follow.

On the other hand, when the extremum seeking and slope tracking controllers are operating together with the last line feedforward signal, the $V^{\mp}$ are successfully tracked and the resulting errors are smaller. Thus, we see that the probability of both controllers successfully tracking the dips is significantly higher with feedforward.

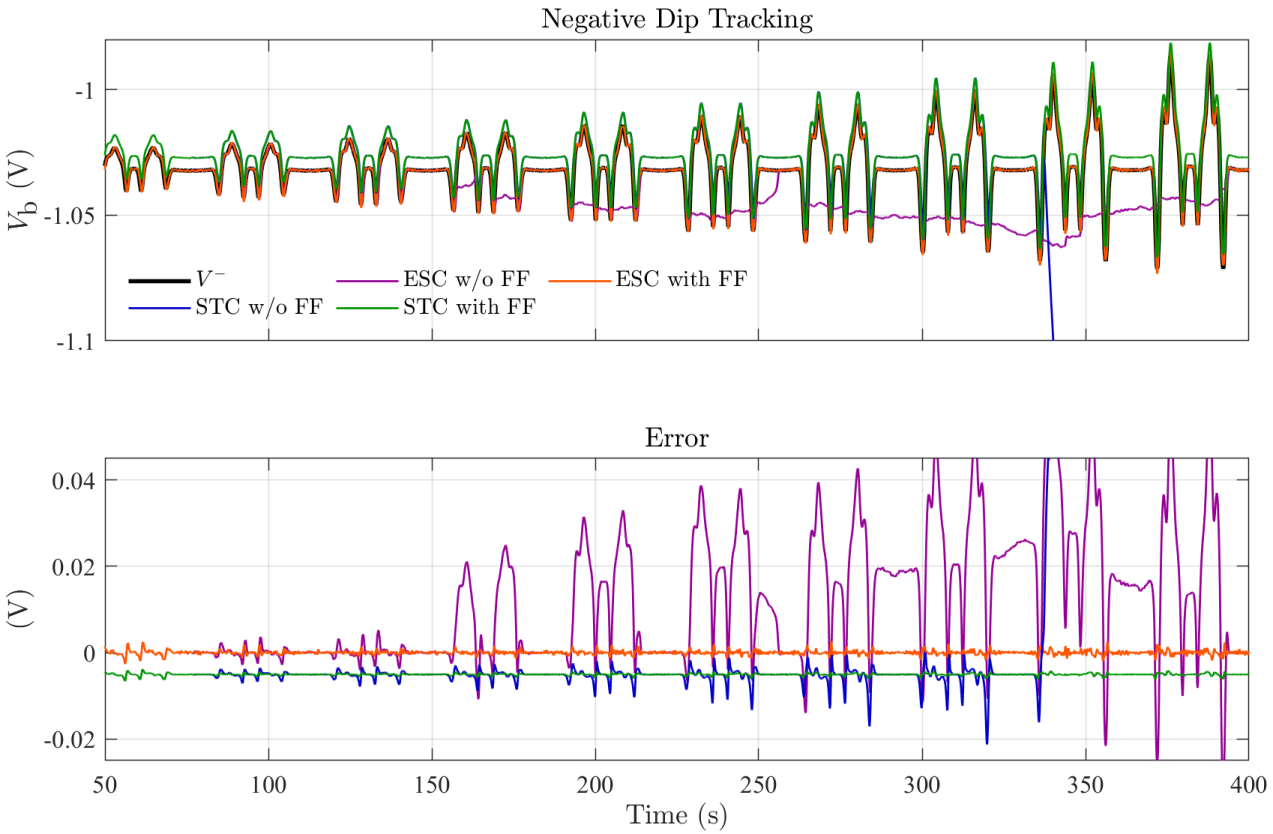


Figure 4.22: The top figure shows the reference evolution (V^-) together with the results of the extremum seeking and slope tracking controllers (ESC and STC) with and without feedforward (FF). The bottom figure shows the error. Note that each image line is scanned forward and backward.

4.4.1.2 Overall Imaging Quality

Now we turn to the complete scans of the negative and positive dip maps $V^{\mp}(\boldsymbol{p})$ and the resulting electrostatic potential $\Phi^*(\boldsymbol{p})$ computed by (4.1). Note that, as it becomes apparent in the following, the implementation of the scanning process in the simulations is done column by column (rather than line by line as outlined so far) to simplify the implementation.[7] Thus, predictions are also generated column-wise. This does, however, not change the principle observations.

Fig. 4.24a and Fig. 4.24b show the resulting maps of the extremum seeking and slope tracking controller without the last line feedforward signal. Also here we observe that both dips cannot be tracked throughout the complete image scan, which results in corrupted final electrostatic potential images.

On the other hand, Fig. 4.24c and Fig. 4.24d show the resulting maps of the extremum seeking and slope tracking controller with the last line feedforward, which are now successfully tracked, though with some oscillations in the $V^+(\boldsymbol{p})$ map. Due to the way Φ^* is computed (4.1) and the larger variation of $V^-(\boldsymbol{p})$ ($\approx 200\,\mathrm{mV}$ vs. $\approx 100\,\mathrm{mV}$

[7] A line by line scan could be realized by rotating the image before processing, do a column-wise processing, and rotating it back afterwards. The results would be the same.

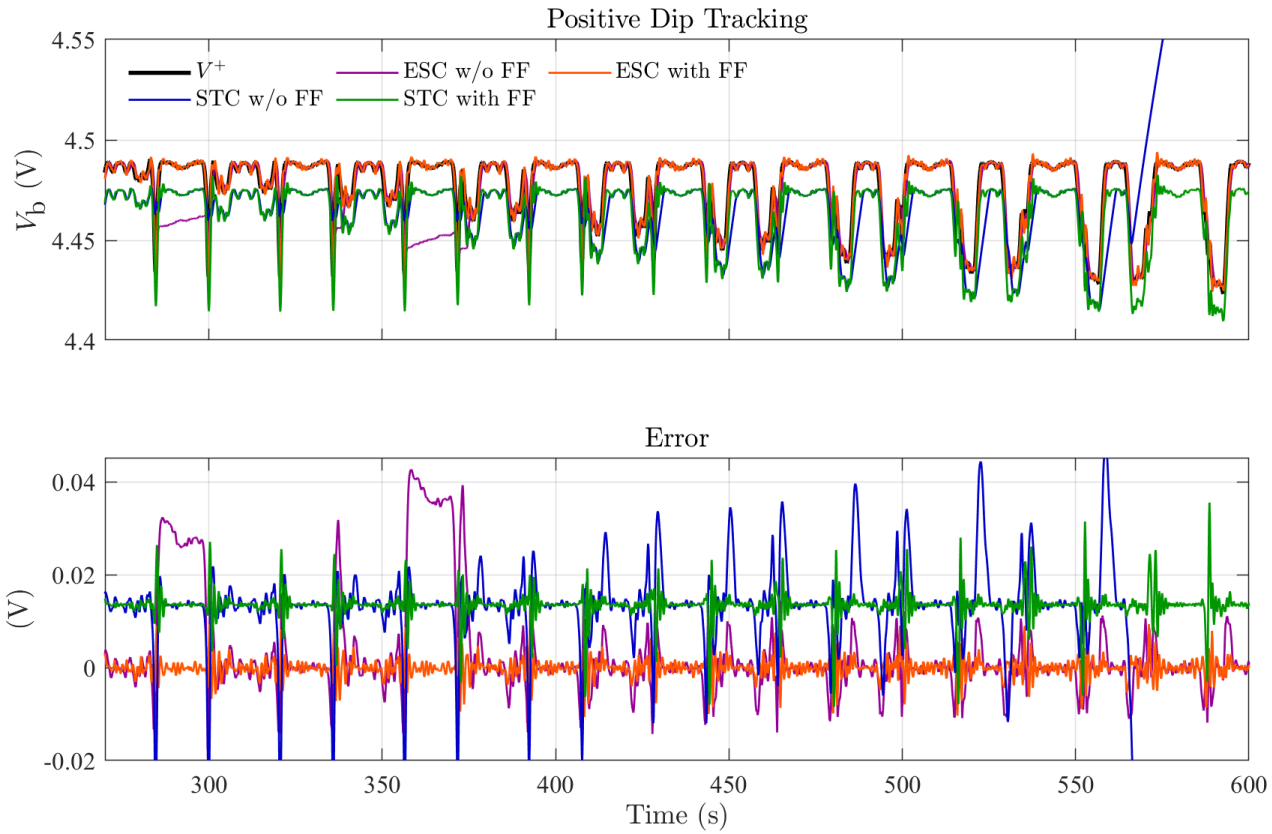


Figure 4.23: The top figure shows the reference evolution (V^+) together with the results of the extremum seeking and slope tracking controllers (ESC and STC) with and without feedforward (FF). The bottom figure shows the error. Note that each image line is scanned forward and backward.

for $V^+(\boldsymbol{p})$), the oscillations of $V^+(\boldsymbol{p})$ are attenuated in the final Φ^* images.

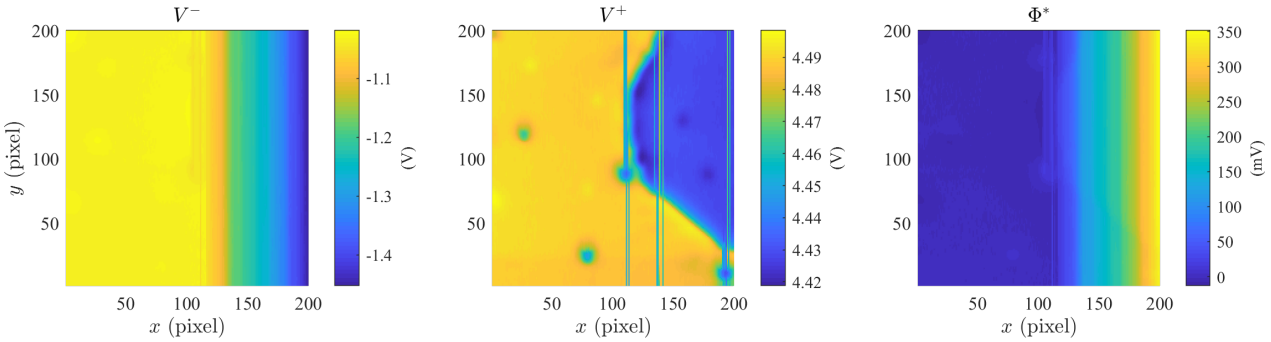
To better evaluate these results, we provide the respective error maps of the feedforward case in Fig. 4.25. They reveal that the errors are in the range of $-5\,\mathrm{mV}$ to $5.6\,\mathrm{mV}$. As the total Φ^* variation is $190.25\,\mathrm{mV}$, the relative errors w.r.t. this variation are $-2.6\,\%$ to $2.9\,\%$. It can be furthermore observed that the extremum seeking control error image presents some small oscillations almost everywhere, whereas the slope tracking control error image generally presents fewer oscillations, except for some larger oscillations in the right part of the sample above the molecular island.

In addition, we quantify the image quality in Table 4.3, using the image mean square error (MSE) per pixel

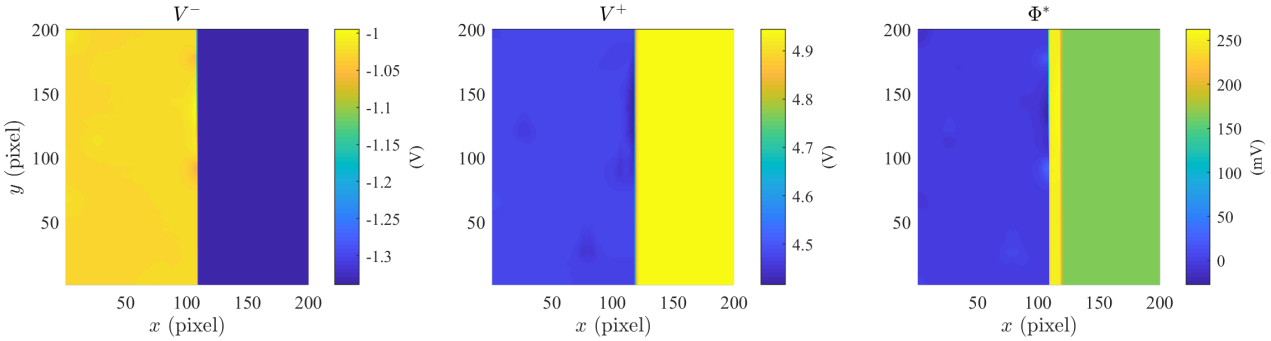
$$MSE = \frac{1}{n_{\mathrm{rows}} n_{\mathrm{cols}}} \sum_{i=1}^{n_{\mathrm{cols}}} \sum_{j=1}^{n_{\mathrm{rows}}} \left(V_{\mathrm{ref}}^{\mp}(i,j) - V^{\mp}(i,j) \right)^2$$

and the image peak signal-to-noise ratio (PSNR)

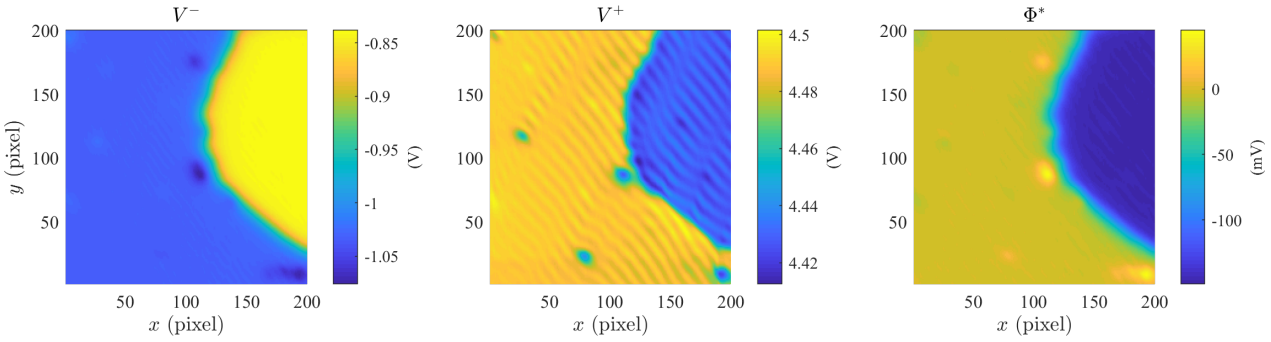
$$PSNR = 10 \log \left(\frac{MAX_{\mathrm{I}}^2}{MSE} \right) ,$$



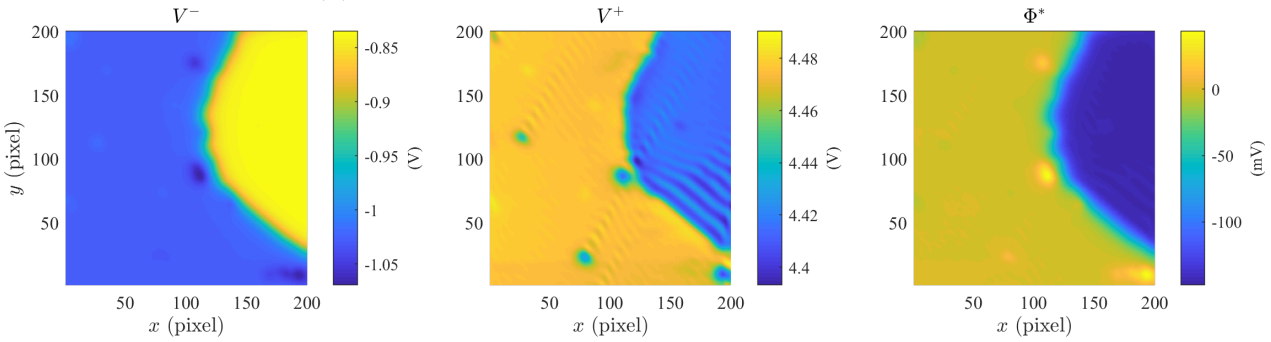
(a) Extremum seeking controller without feedforward



(b) Slope tracking controller without feedforward



(c) Extremum seeking controller with feedforward



(d) Slope tracking controller with feedforward

Figure 4.24: Resulting images with the 2DOF controller. Depicted are the resulting negative dip maps $V^-(\boldsymbol{p})$ (left), positive dip maps $V^+(\boldsymbol{p})$ (middle), and electrostatic potential $\Phi^*(\boldsymbol{p})$ (right).

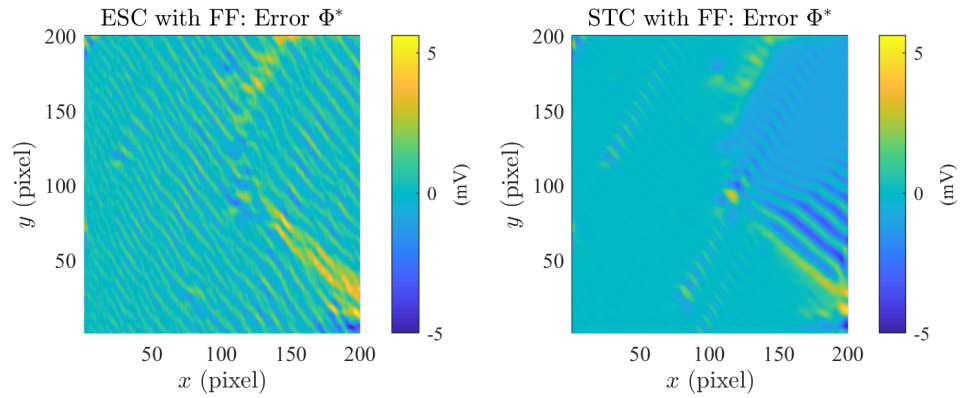


Figure 4.25: Resulting image errors of the 2DOF controllers.

where n_{rows} and n_{cols} are the number of image rows and columns and MAX_{I} the maximum possible pixel value of the image, which is taken from the range of the image data type and is automatically determined by MATLAB. The MSE and PSNR are often used in imaging science as objective quality measures [12, 171].

Based on Table 4.3 we can conclude from a quantitative viewpoint that the slope tracking controller has the potential to outperform the extremum seeking controller. However, for different controller parameters and/or other surfaces, the results may change and therefore more experience via simulations and experiments has to be gathered. Note that the systematic slope tracking control error e_{STC}, as discussed in Section 4.3.1.2, is already included in these results and is apparently not very important for the final Φ^* image.

Table 4.3: MSE and PSNR per pixel for Fig. 4.25.

	MSE [mV]	PSNR [dB]
Extremum seeking	0.804	60.9
Slope tracking	0.507	63.0

The total scan time for each of the results was 4 h (2 h per dip map). This is approximately 17 times faster than the original image generation process, based on spectroscopy grids that would have taken 66.7 h (see Section 4.2.2). In fact, such a long measurement time would be completely impractical or close to impossible in experiments. Therefore, without the 2DOF approach images of that size and resolution would not be possible.

4.4.2 2DOF Control with Gaussian Process Feedforward

We evaluate the potential of the Gaussian process-based feedforward generation (FF-GP) by comparing the resulting maps assembled from offline last line feedforward

(FF-LL) predictions and offline GP predictions (Section 4.4.2.1). By the term offline we refer to the case that we use only the experimentally acquired $V^{\mp}(\boldsymbol{p})$ reference data to compute the predictions. After that, we move on to the online/closed-loop case (Section 4.4.2.2), where the data points used to generate the predictions are also influenced by the employed feedback controller and therefore deviate from the reference $V^{\mp}(\boldsymbol{p})$ values.

4.4.2.1 Offline Comparison

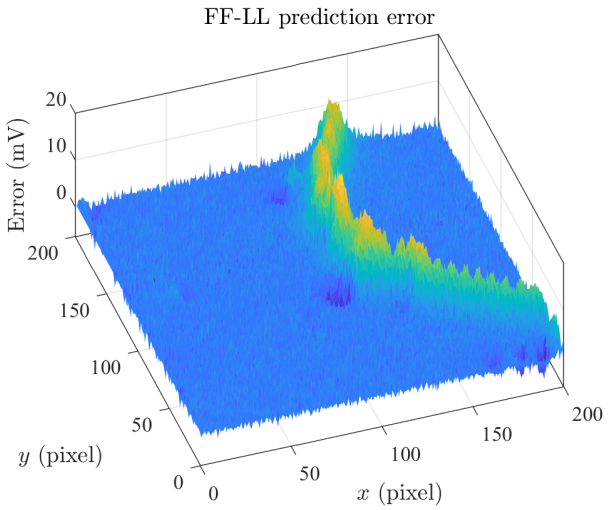
The offline last column/line predicted map is simply generated by substituting a column/line with the previous one. The resulting error maps for the negative and positive dip are depicted in Fig. 4.26a and Fig. 4.26b. The errors are small where the respective $V^{\mp}(\boldsymbol{p})$ map is relatively flat (columns are similar to their precursors) and errors are large when the $V^{\mp}(\boldsymbol{p})$ values change considerably from one column to the next. The MSE and maximum errors for the negative and positive dip maps are listed in Table 4.4. The errors associated with the negative dip are larger because the change of the negative dip position is larger than that of the positive dip position ($\approx 200\,\mathrm{mV}$ vs. $\approx 100\,\mathrm{mV}$, see Fig. 4.19). Thus, also the errors from one column to the next are larger for the negative dip position map.

Table 4.4: Offline comparison of last line (FF-LL) and Gaussian process based (FF-GP) feedforward in Fig. 4.26.

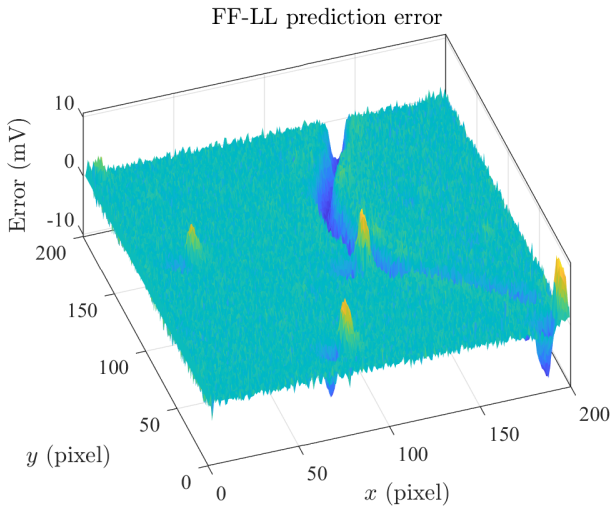
	FF-LL		FF-GP	
	V^-	V^+	V^-	V^+
MSE [mV]	7.003	2.511	0.7588	0.9626
max. error [mV]	17.24	10.66	8.450	8.955

Concerning the GP predictions, the models $\mathcal{GP}_{V^-}$ and $\mathcal{GP}_{V^+}$ for the negative and positive dip maps, respectively, are assigned a constant zero prior mean function and the squared exponential kernel (2.7) with one and the same length scale parameter for both inputs $\boldsymbol{p} = (x, y)$. The GPs are then trained using columns 90 to 120, which cover the edge of the large island. The optimized hyperparameters are $\boldsymbol{\theta}_{V^-} = \{c = -0.9784, l = 5.8766, \sigma_\mathrm{f}^2 = 0.0265\}$ and $\boldsymbol{\theta}_{V^+} = \{c = 4.4706, l = 4,7710, \sigma_\mathrm{f}^2 = 0.0137\}$. The predictions are generated using the previous five columns as training data. The resulting error maps of the GP predictions are presented in Fig. 4.26c and Fig. 4.26d and we observe that the errors are considerably smaller on average. This is also confirmed quantitatively by means of smaller MSE and maximum errors in Table 4.4.

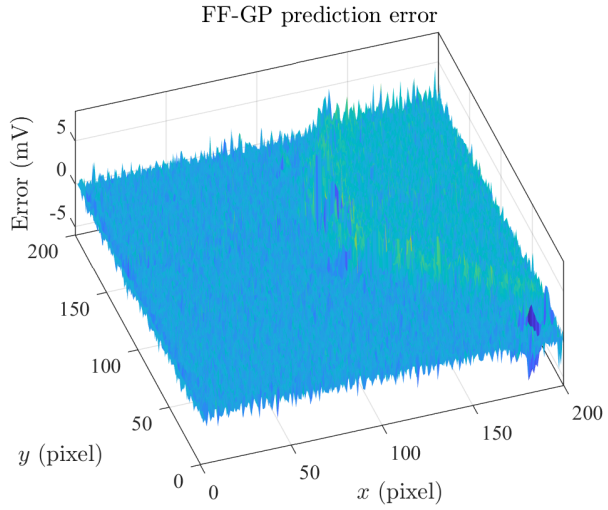
The superior prediction capabilities of the Gaussian process feedforward, especially in regions where the dip maps (and therewith the electrostatic potential) change considerably, is reflected in Fig. 4.27. There, we compare the average pixel error per



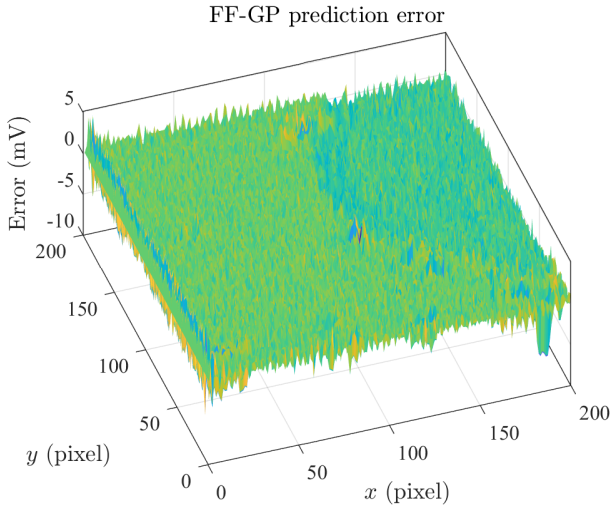
(a) Negative dip map $V^-(\boldsymbol{p})$.



(b) Positive dip map $V^+(\boldsymbol{p})$.

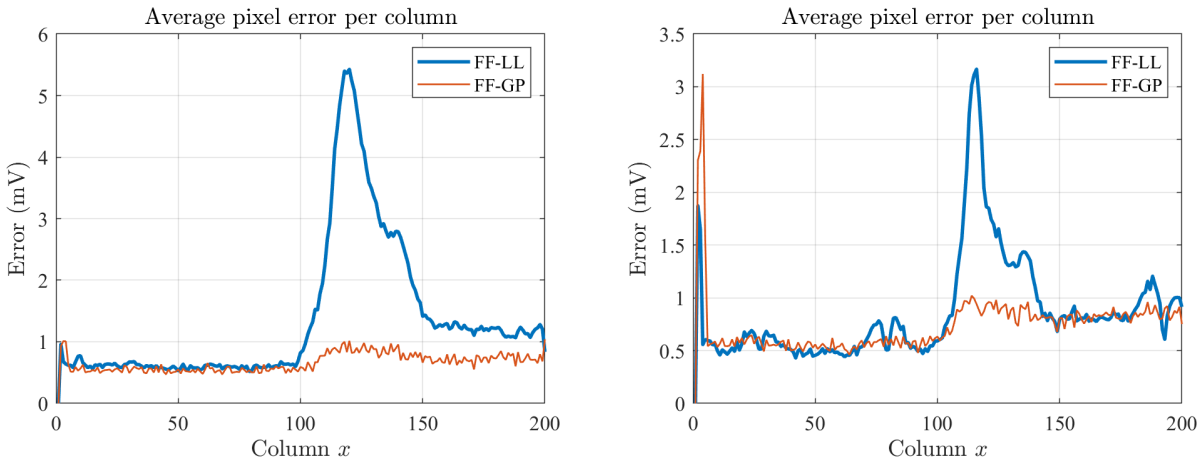


(c) Negative dip map $V^-(\boldsymbol{p})$.



(d) Positive dip map $V^+(\boldsymbol{p})$.

Figure 4.26: Comparison of the offline computed (best outcome) last line feedforward (FF-LL, top row) and Gaussian process feedforward (FF-GP, bottom row) prediction errors for the negative (left column) and positive (right column) dip maps.



(a) Results for negative dip map $V^{-}(\boldsymbol{p})$. **(b)** Results for positive dip map $V^{+}(\boldsymbol{p})$.

Figure 4.27: Comparison of the average pixel error in each column.

column

$$\frac{1}{n_{\text{rows}}} \sum_{i=1}^{n_{\text{rows}}} |V^{\mp}(x,i) - \hat{V}^{\mp}(x,i)| \ ,$$

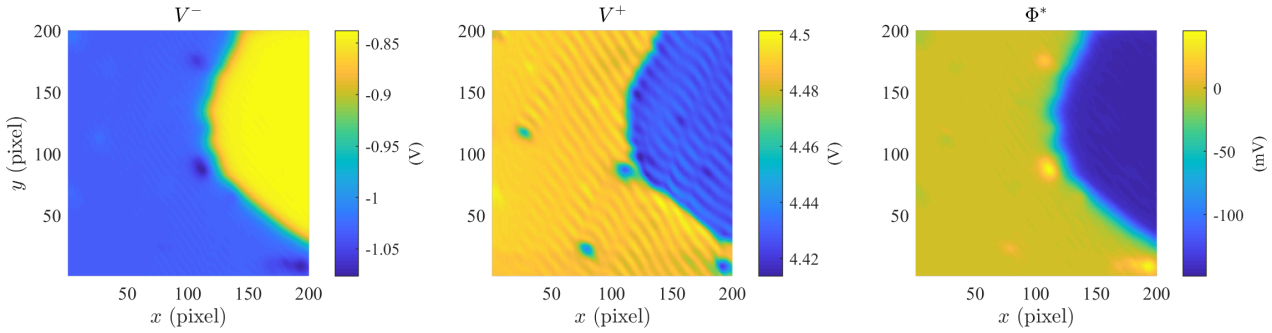
where x denotes the column index, n_{rows} is the number of rows or pixels in one column, and $\hat{V}^{\mp}$ the predicted voltage value by either the last line or the Gaussian process feedforward

The resulting errors are similar for the first 100 columns. An exception appears right at the beginning of the dip maps, where the Gaussian process feedforward error is larger because it does not yet have the required five columns of training data points. After the first 100 columns, where the large island appears, the Gaussian process feedforward clearly outperforms the last line feedforward, especially where the $V^{\mp}(\boldsymbol{p})$ values change considerably.

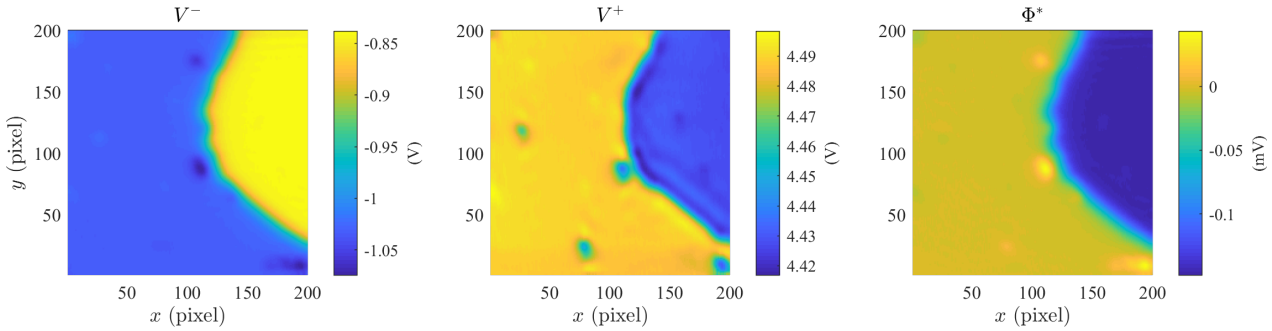
4.4.2.2 Closed-loop Comparison

For the closed-loop case, we now use the Gaussian process feedforward together with the extremum seeking and slope tracking controller to determine the $V^{\mp}(\boldsymbol{p})$ values of the complete surface and generate the resulting electrostatic potential image $\Phi^{*}(\boldsymbol{p})$ computed by (4.1). We compare the obtained results with those of the last line feedforward of Section 4.4.1.

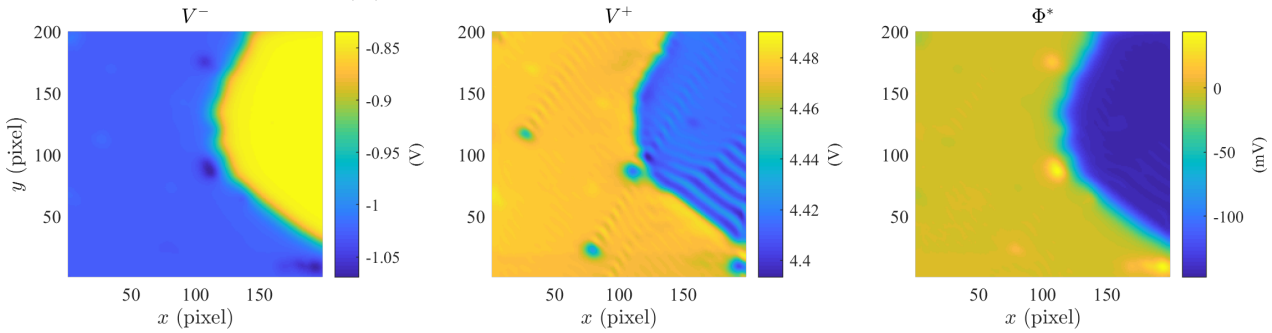
Fig. 4.28a (same as Fig. 4.24c) and Fig. 4.28b show the resulting maps of the extremum seeking controller with the FF-LL and FF-GP, respectively. We observe that both dips are tracked successfully, with the FF-GP outperforming the FF-LL. The oscillations that appear in Fig. 4.28a are significantly attenuated by the FF-GP. The results for the slope tracking controller (Fig. 4.28c and Fig. 4.28d) are qualitatively similar to that of the extremum seeking controller, i.e., the FF-GP outperforms the



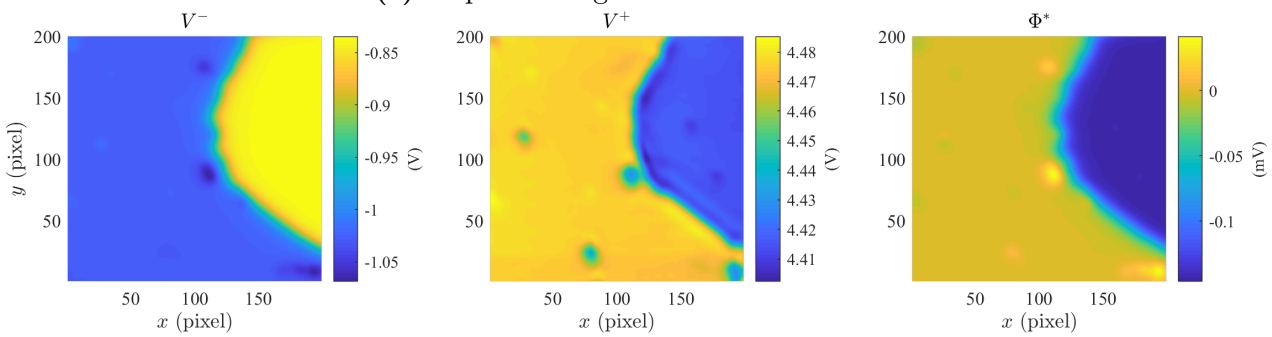
(a) Extremum seeking controller with FF-LL



(b) Extremum seeking controller with FF-GP

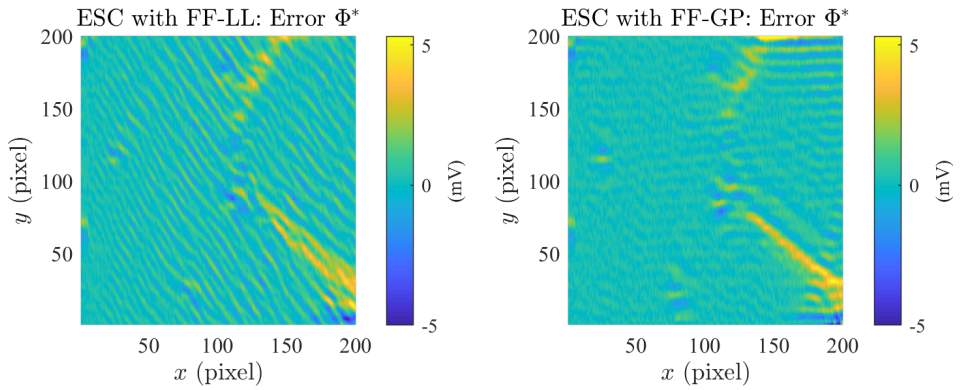


(c) Slope tracking controller with FF-LL

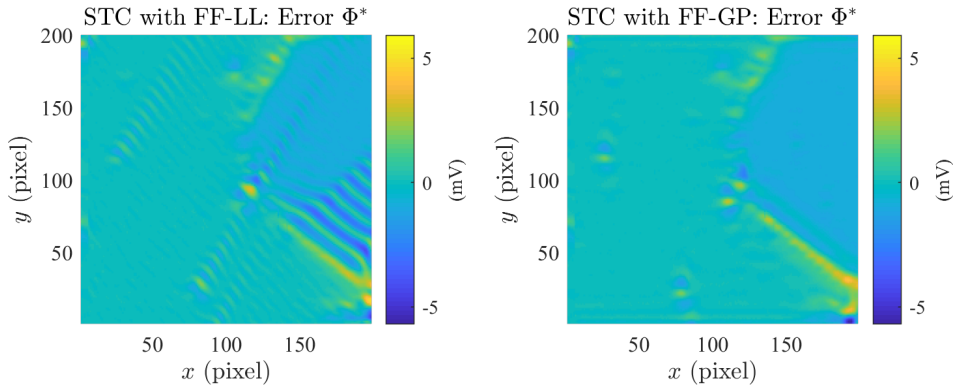


(d) Slope tracking controller with FF-GP

Figure 4.28: Comparison of the 2DOF controllers with the last line feedforward (FF-LL) and the Gaussian process feedforward (FF-GP). Depicted are the resulting negative dip maps $V^-(\boldsymbol{p})$ (left), positive dip maps $V^+(\boldsymbol{p})$ (middle), and electrostatic potential $\Phi^*(\boldsymbol{p})$ (right). The images generated with the FF-LL (Fig. 4.28a and Fig. 4.28c) are the same as Fig. 4.24c and Fig. 4.24d.



(a) 2DOF with extremum seeking controller



(b) 2DOF with slope tracking controller

Figure 4.29: Resulting image errors of the 2DOF controllers (extremum seeking and slope tracking controller) with the FF-LL and FF-GP. The error images on the left are the same as in Fig. 4.25.

FF-LL.

For better comparison, we evaluate the respective error maps in Fig. 4.29. In case of the extremum seeking controller (Fig. 4.29a), the small oscillations that appear almost everywhere with the FF-LL are reduced when instead the FF-GP is employed. The same goes for the larger oscillations in case of the slope tracking controller in Fig. 4.29b. Some (relatively) large errors, however, remain in both cases at the island edge in the right part of the sample. The resulting error image of the slope tracking controller with the FF-GP looks qualitatively better than that generated with the extremum seeking controller. This observation is also confirmed quantitatively in Table 4.5, where we provide the respective MSE and PSNR values. The MSE of the slope tracking controller is significantly smaller than that of the extremum seeking controller and the PSNR is larger.

Thus, we conclude, despite the fact that both the extremum seeking and slope tracking controller with the FF-GP yield good closed-loop performance with small errors in the final Φ^* image, that the slope tracking controller outperforms the extremum seeking controller both qualitatively and quantitatively. Again, for different controller parameters and/or other surfaces, though, the results may turn out differently. For

Table 4.5: MSE and PSNR per pixel for Fig. 4.29.

	MSE [mV]	PSNR [dB]
ESC FF-LL	0.804	60.9
ESC FF-GP	0.653	61.7
STC FF-LL	0.507	63.0
STC FF-GP	0.397	64.0

instance, online hyperparameter optimization of the Gaussian process could further reduce the errors and change the outcome for both extremum seeking and slope tracking controller.

4.5 Experimental Results

In this section, we briefly present some information on the implementation of the 2DOF controller in the experimental setup, discuss the SQDM operation procedure, and present experimentally generated images of nanoscale samples with the presented control approach of this work. For the 2DOF controller we only present results for the last line feedforward. The Gaussian process-based feedforward scheme is subject of future work.

4.5.1 Implementation

The 2DOF controller is designed and tested in MATLAB/Simulink. For the experimental implementation it is automatically converted into C-code and then loaded to and executed on a dSPACE ds1104 controller board. The controller board is installed in a PC and connected via analog-digital and digital-analog converters to a Createc non-contact atomic force/scanning tunneling microscope (STM/NC-AFM, Fig. 4.30) that operates at 5 K and under ultra-high vacuum. The microscope is equipped with a qPlus sensor tuning fork [63] with resonance frequency $f_0 = 31.2\,\mathrm{kHz}$ and stiffness $\kappa_0 = 1800\,\mathrm{Nm}^{-1}$.

Note that the scanning procedure and acquisition of the $V^{\mp}(\boldsymbol{p})$ data are different and largely independent processes (see Fig. 4.31). The scanning procedure is controlled by the hardware and associated computer of the Createc STM/NC-AFM. This involves the generation of the reference signal $\boldsymbol{p}_{\mathrm{ref}}$ for the microscope's piezo stages and control of the stages such that the reference signal is tracked with high accuracy, which yields a precise scanning pattern. The PC on which the 2DOF controller is running is responsible for regulation of the V_{b} voltage. To this end, it also receives the current tip position $\boldsymbol{p}$.

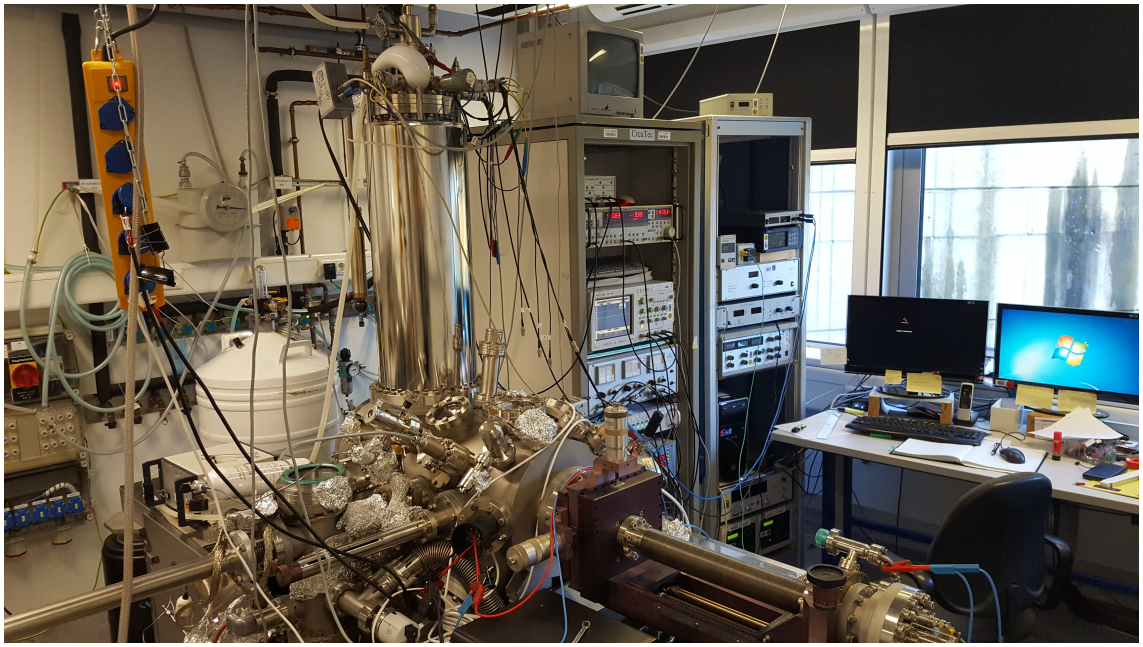


Figure 4.30: The Createc non-contact atomic force/scanning tunneling microscope, located in the laboratory of the Peter Grünberg Institute at the Jülich Research Centre.

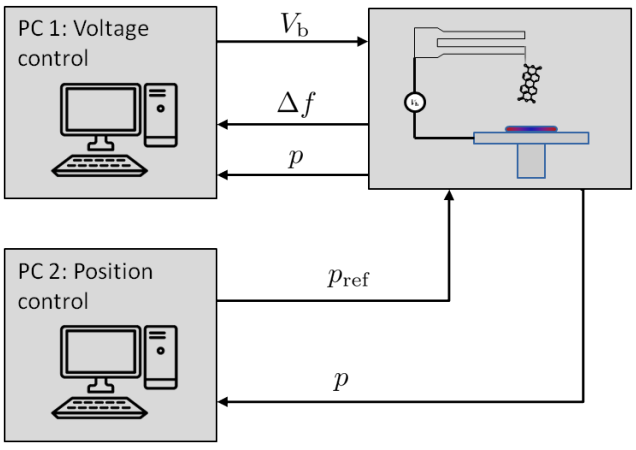


Figure 4.31: Experimental SQDM set-up.

4.5.2 Scanning and Image Generation Procedure

The procedure used to acquire a new image is as follows:

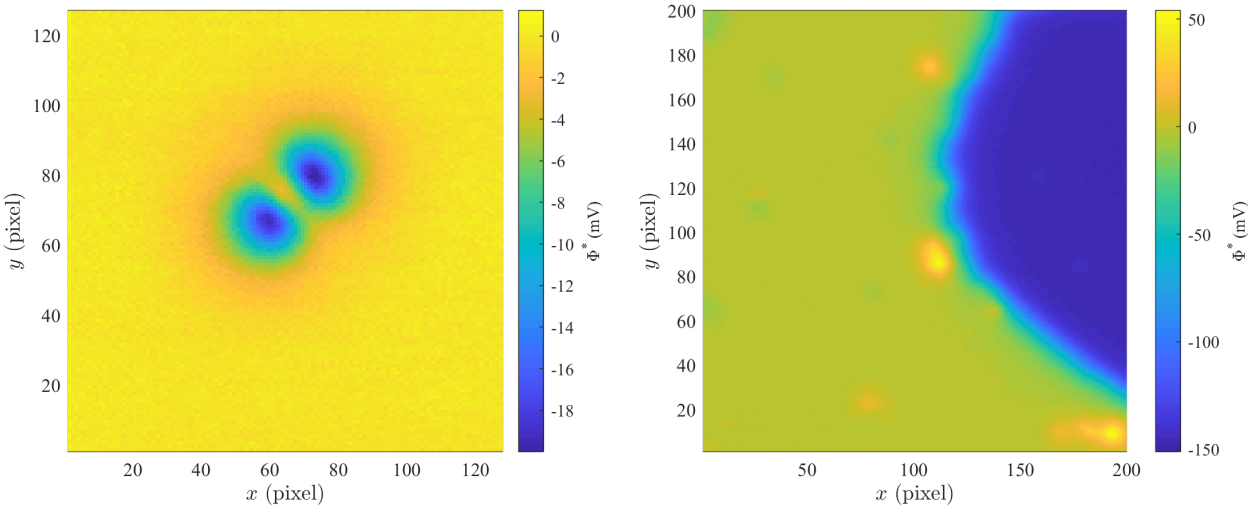
1. Move to the first image pixel and measure the local spectrum $\Delta f^{\mp}(V_\mathrm{b})$ of the respective dip $V^{\mp}(\boldsymbol{p})$.
2. Adjust V_b manually to the reference point within the dip (the selection of the reference point depends on whether the slope tracking or the extremum seeking controller is used, see Fig. 4.12).
3. Start the 2DOF controller.
4. Start the raster scanning protocol.
5. Enable the feedforward after one or more lines have been scanned.
6. Increase scanning speed if desired.
7. Finish the scan for the current dip $\Delta f^{\mp}(V_\mathrm{b})$.
8. Goto 1) and repeat for the other dip $\Delta f^{\pm}(V_\mathrm{b})$.

4.5.3 Experimental Results

We present a collection of obtained SQDM images.

Fig. 4.32a is one of the first images generated via SQDM and the 2DOF controller [195] and shows the electrostatic potential of a single PTCDA molecule. Accordingly, the covered image area is relatively small with $150 \times 150\,\text{Å}^2$. On the other hand, Fig. 4.32b, which is used throughout this chapter as reference image, is the image that covers the largest area so far generated via SQDM with a dimension of $600 \times 600\,\text{Å}^2$. As explained further above, the time required for the image generation without the 2DOF controller would have been 66.7 h. With the controller the scan time was reduced to 4 h in total, which made the generation of the image only possible in the first place.

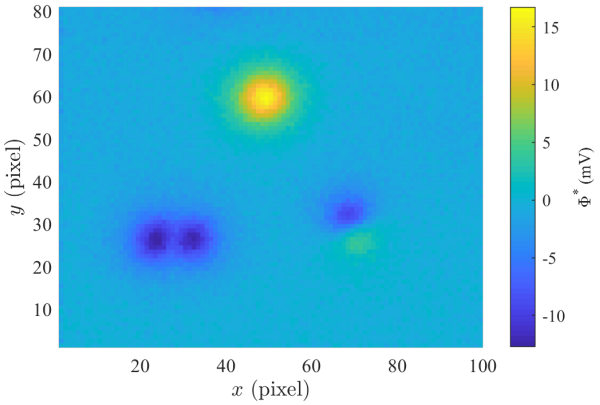
This also holds for the images in Fig. 4.33, especially for Figs. 4.33b and 4.33c, which are divided into 256×256 pixels. Using the original image generation procedure, based on spectroscopy grids, the time required for the corresponding measurements of each $V^{\mp}(\boldsymbol{p})$ map would be approximately 109 h. The scan times with the 2DOF controller for each $V^{\mp}(\boldsymbol{p})$ map were 2 h for Fig. 4.33b and 0.75 h for Fig. 4.33c, equalling a speed increase of a factor of 54.5 and 145 respectively. These factors are expected to increase further with the implementation of the Gaussian process feedforward in the experimental setup.



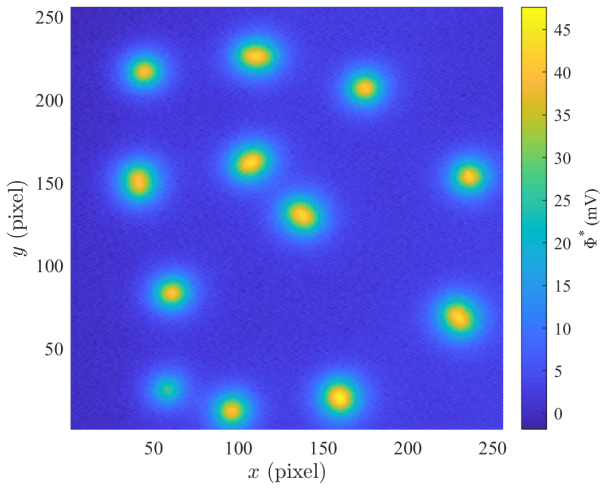
(a) SQDM image of a single PTCDA molecule. Dimensions: $150 \times 150\,\text{Å}^2$ divided into 128×128 pixels. Measured at a height of $z = 24.4\,\text{Å}$ above the surface and a scan speed of $18\,\text{Å/s}$.

(b) The SQDM image published in [196] encompasses a large PTCDA island on the right and several more individual atoms and molecules located along the island edge or to its left. Dimensions: $600 \times 600\,\text{Å}^2$ divided into 200×200 pixels. Measured at a height of $z = 20\,\text{Å}$ above the surface and a scan speed of $33\,\text{Å/s}$.

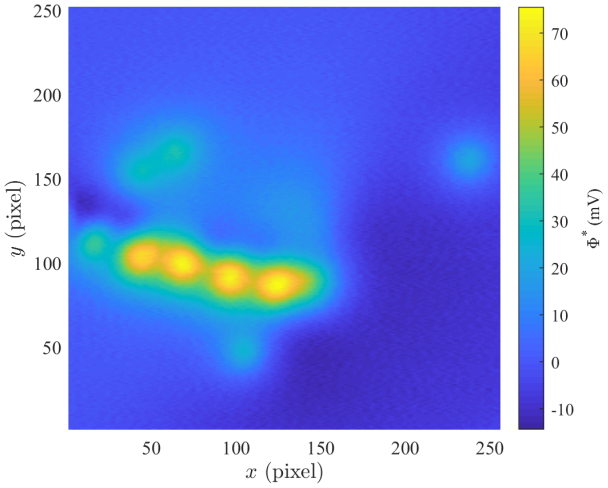
Figure 4.32: Experimentally acquired SQDM images of electrostatic potential Φ^*. All experiments were conducted on silver Ag(111) substrate.



(a) SQDM image of a PTCDA molecule, a PTCDA-Ag_2 molecule, and a Ag adatom. Dimensions: $260 \times 210\,\text{Å}^2$ divided into 100×81 pixels pixels. Measured at a height of $z = 29 \pm 1.5\,\text{Å}$ above the surface and a scan speed of $14\,\text{Å/s}$.



(b) SQDM image of many Ag dimers (two adatoms right next to each other) and one adatom. Dimensions: $360 \times 360\,\text{Å}^2$ divided into 256×256 pixels. Measured at a height of $z = 29 \pm 1.5\,\text{Å}$ above the surface and a scan speed of $36\,\text{Å/s}$.



(c) SQDM image of a structural defect of the Ag(111) surface. The defect is decorated with PTCDA molecules that are alternatingly standing on their long side and lying flat. Dimensions: $200 \times 200\,\text{Å}^2$ divided into 256×256 pixels. Measured at a height of $z = 27 \pm 1.5\,\text{Å}$ above the surface and a scan speed of $30\,\text{Å/s}$.

Figure 4.33: Experimentally acquired SQDM images of electrostatic potential Φ^*. All experiments were conducted on silver Ag(111) substrate.

4.6 Conclusions

In this chapter, we have presented a control approach for scanning quantum dot microscopy, a recently developed microscopy technique that generates images of the electrostatic potential of nanostructures with unprecedented accuracy. A key enabling part of this is the developed two-degree-of-freedom controller of this chapter, without which most of the so far experimentally generated images would not have been possible. It now allows to continuously scan a sample, which puts scanning quantum dot microscopy in line with other scanning probe microscopy techniques like scanning tunneling or atomic force microscopy.

The presented control framework encompasses a feedback and a feedforward part. For the feedback part, we have developed two different controllers, namely an extremum seeking controller that directly tracks the minimum of the involved dips that arise in scanning quantum dot microscopy and a controller that tracks a reference point on the dips' slope. We have discussed the individual working principles, respective advantages and drawbacks, and provided guidelines for controller parameterization. Among these, we have shown that the sharpness of the dips plays a central role in the control performance, which leads to the recommendation that practitioners of scanning quantum dot microscopy should aim for sharper dips if possible. Simulations with the used reference data and the chosen controller parameters have shown that both feedback controllers yield good closed-loop performance, though the slope tracking controller slightly outperforms the extremum seeking controller, despite having an intrinsic nonzero error.

For the feedforward part, we outlined two different approaches. The first approach stores and filters the dips' position values of the previously scanned line during the scanning process and uses it as a prediction for the next line. The second approach utilizes Gaussian processes to model each of the dips' position maps via data of their already scanned parts. These Gaussian process models then allow generating better predictions for the upcoming lines to be scanned by the microscope.

In simulations, we could show how the utilization of a feedforward decreases the resulting image error and with that the probability of losing the dips. This leads to a several times faster image generation process and enables to scan larger images in reasonable time than before, which was also verified in experiments. The simulations have also shown that the more complex Gaussian process-based feedforward outperforms the simple last line approach, thereby illustrating the potential of Gaussian processes in cutting edge technologies.

The presented work leaves room for improvements and further development. Regarding the performance of the two feedback controllers, the obtained simulation results have to be further validated in experiments. In particular, if the slope tracking controller really and always outperforms the extremum seeking controller or if this depends

on the sample to be scanned and the chosen controller parameters.

Related to this, a further improvement would be an automatic controller parameter adaptation scheme for varying scan speeds. As was shown in this chapter, several parameters (e.g. the controller gains) can be tuned to achieve fast convergence, which is important for high scan speeds. The respective parameterizations, however, come at the expense of larger transient oscillations. Thus, for larger scan times (lower scan speeds) other parameterizations are better suited and an automatic controller parameter adaptation scheme could account for this.

Regarding the Gaussian process feedforward, so far, a priori knowledge of the involved hyperparameters was assumed to show the potential for improving closed-loop performance. For successful experimental application, however, such a priori knowledge cannot be assumed to be available for new unseen samples. Hence, the hyperparameters have to be calculated and updated online during the scanning process. Since hyperparameter optimization is computationally expensive, especially for real-time operation, the deployment of methods such as sparse Gaussian process approaches and optimization in parallel to the scanning process are currently investigated.

Another interesting question that could be investigated is whether the uncertainty measure provided by the GP (in the form of its posterior variance) can be used in a beneficial way for the performance of the closed-loop. One idea is to check if the uncertainty should influence the current scan speed, for instance, decrease the scan speed if the uncertainty is large and vice versa.

5 Conclusions and Outlook

This thesis considers Gaussian process supervised learning and its deployment in control. As a specific example of machine learning algorithms, Gaussian processes are capable to learn and make designated use of the increasingly available data to enhance control performance, which is an advantageous property to cope with the technological developments and challenges of today's world. For example, learning from data enables both simple and complex systems to better adapt to changing process conditions or their surroundings, thereby increasing autonomy and performance. Furthermore, through the steady increase of computational power and theoretical developments, the benefits of Gaussian processes can be better absorbed into the design and the execution of control systems. This thesis aimed at pushing further in this direction.

In Chapter 2 we reviewed Gaussian processes for regression in supervised learning and the literature of its utilization in a control context. In particular, we considered the cases of reference and disturbance learning, as well as learning models of dynamical systems for model-based control. Thereby, we set up the frame for the developments of the following chapters, where we presented both theoretical and practical relevant contributions regarding the combination of Gaussian processes and control.

In Chapter 3 we presented recursive Gaussian process model predictive control (rGP-MPC), a specific model predictive control scheme that employs Gaussian process prediction models which are learned online. The approach was designed in such a way as to maximize the possible cases for application (e.g. no process model available, changing process conditions, fast dynamics, or non-accessible process states). We derived conditions under which the involved optimal control problem is recursively feasible and the resulting control law nominally and inherently robustly stable, despite the changing prediction model. These guarantees are not limited to rGP-MPC but are applicable to a wide spectrum of model predictive control schemes that use Gaussian process prediction models and other machine learning algorithms.

In Chapter 4 we presented the first control framework utilizing Gaussian processes for the novel microscopy technique *scanning quantum dot microscopy*, which generates images of the electrostatic potential of nano-size objects (e.g. atoms or molecules). The developed control framework is a two-degree-of-freedom controller with two feedback and two feedforward controllers. In one of the feedforward parts, Gaussian processes are learning and predicting specific disturbances which can then be accounted for. This helps in correct operation and improves the overall performance. The developed control framework allows generating images by continuously scanning the sample and puts scanning quantum dot microscopy in line with other established scanning probe

microscopy techniques.

Specific outlooks and possible future undertakings of rGP-MPC and control of scanning quantum dot microscopy are discussed at the end of the respective chapters and we conclude with a general outlook on Gaussian processes in control.

Over the last two decades, there have been many theoretical works and some practical applications that combine Gaussian processes and control. This development is furthermore accelerating as the increasing number of scientific publications in recent years indicates and it can be interpreted as an expression of the great interest to combine the benefits of system and control theory with Gaussian processes in particular and machine learning in general. While methods from machine learning are strongly based on data and enable adaptive and high performance behavior, methods from control allow for safe behavior with respective guarantees. The latter are particularly important when it comes to processes that are safety critical, such as self-driving cars or autonomous drones in urban areas. At the moment, the number of concrete real world applications that employ machine learning is lacking behind its possibilities. Therefore, the ongoing fusion of the two fields is necessary and expected to intensify in upcoming years to develop further concepts that enable and enhance the autonomy of today's present and arising technologies. As a side effect, the two communities of control theory and machine learning, which can be considered as once separated, are approaching each other, blurring the lines between them and therewith creating new opportunities for research, collaborations, and technological advancements.

A Appendix

A.1 Function Definitions and Lemmas

In this section, we provide the definitions of important functions and classes of functions that are used in this thesis.

Definition 7 (Distance function)**.** *The distance of a point $\boldsymbol{z} \in \mathbb{R}^p$ to a set $\mathcal{Y} \subset \mathbb{R}^p$ is defined by the function $d(\boldsymbol{z}, \mathcal{Y}) := \inf_{\boldsymbol{y}\in\mathcal{Y}} \|\boldsymbol{y} - \boldsymbol{z}\|_\infty$, where $\|\cdot\|_\infty$ is the infinity norm. If $\boldsymbol{z} \in \mathcal{Y}$, then $d(\boldsymbol{z}, \mathcal{Y}) = 0$.*

Definition 8 (Positive (semi-) definite function)**.** *A function $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is positive definite if $f(0) = 0$ and $f(x) > 0$ for all $x \neq 0$. It is positive semidefinite if $f(0) = 0$ and $f(x) \geq 0$ for all $x \neq 0$, i.e., the function can be zero also at other locations than $x = 0$.*

Definition 9 ($\mathcal{K}$-function)**.** *A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}$-function if it is continuous, $\alpha(0) = 0$, and if it is strictly increasing.*

Note that a $\mathcal{K}$-function is a special case of a positive definite function.

Definition 10 ($\mathcal{K}_\infty$-function)**.** *A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}_\infty$-function if it is a $\mathcal{K}$-function and unbounded, i.e., $\lim_{x\to\infty} \alpha(x) = \infty$.*

Definition 11 ($\mathcal{KL}$-function)**.** *A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{KL}$-function if $\beta(s, t)$ is $\mathcal{K}_\infty$ in s for any value of t and $\lim_{t\to\infty} \beta(s, t) = 0, \forall s \geq 0$.*

Lemma 1 ([107], Lemma 1)**.** *Let f be a function $f(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^c$. Then, f is uniformly continuous in $\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathcal{A} \subseteq \mathbb{R}^a$ and all $\boldsymbol{y} \in \mathcal{B} \subseteq \mathbb{R}^b$ if and only if there exists a $\mathcal{K}_\infty$-function $\sigma(\cdot)$, such that*

$$\|f(\boldsymbol{x}_1, \boldsymbol{y}) - f(\boldsymbol{x}_2, \boldsymbol{y})\| \leq \sigma(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|), \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{A},\ \forall \boldsymbol{y} \in \mathcal{B}\ .$$

Lemma 2 ([107], Lemma 2)**.** *Consider a discrete-time system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ with $\boldsymbol{x}_k \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$. Denote with $\boldsymbol{x}_{k+i} = \phi(i, \boldsymbol{x}_k, \mathbf{u})$ the solution to this equation at time instant i with initial condition $\boldsymbol{x}_k$ and input sequence $\mathbf{u}$. Assume that $f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is uniformly continuous in $\boldsymbol{x}_k$ for all $\boldsymbol{x}_k \in \mathcal{X}$ and all $\boldsymbol{u}_k \in \mathcal{U}$, and let $\sigma_x(\cdot)$ be a suitable function such that $\|f(\boldsymbol{x}_1, \boldsymbol{u}) - f(\boldsymbol{x}_2, \boldsymbol{u})\| \leq \sigma_x(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|)$. Then,*

$$\|\boldsymbol{x}_{k+i} - \boldsymbol{z}_{k+i}\| \leq \sigma_x^i(\|\boldsymbol{x}_k - \boldsymbol{z}_k\|)\ .$$

A.2 Stability Theory

Stability theory is a wide subject with many different concepts and definitions of stability and results on how stability can be guaranteed. In this section, we provide only the necessary definitions and results required for this work.

A.2.1 Nominal Stability

Definition 12 (Positive and control invariant sets)**.**

- *A set $\mathcal{X}$ is positive invariant for the system $\boldsymbol{x}^+ = f(\boldsymbol{x})$ if $\boldsymbol{x}^+ \in \mathcal{X}$ for all $\boldsymbol{x} \in \mathcal{X}$.*
- *A set $\mathcal{X}$ is control invariant for the system $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{u})$ if for all $\boldsymbol{x} \in \mathcal{X}$ there exists a $\boldsymbol{u} \in \mathcal{U}$ such that $\boldsymbol{x}^+ \in \mathcal{X}$.*

Definition 13 ((Global) asymptotic stability)**.** *Assume that the set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is positive invariant for the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k)$ with $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and where the origin is an equilibrium of the system, i.e., $f(0) = 0$. The origin is asymptotically stable in $\mathcal{X}$ if there exists a $\mathcal{KL}$-function $\beta(\cdot, \cdot)$ such that for each $\boldsymbol{x}_0 \in \mathcal{X}$*

$$\|\boldsymbol{x}_k\| \leq \beta(\boldsymbol{x}_0, k) \quad \forall k \in \mathbb{N}_0 \ .$$

If $\mathcal{X} = \mathbb{R}^{n_x}$, the origin is globally asymptotically stable. The set $\mathcal{X}$ is called region of attraction.

Definition 14 (Lyapunov function)**.** *Assume that the set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is positive invariant for the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k)$ with $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and where the origin is an equilibrium of the system. A function $V : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$ is called a Lyapunov function in $\mathcal{X}$ for $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k)$ if there exist functions $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty$ and a continuous and positive definite function $\alpha_3(\cdot)$ such that for any $\boldsymbol{x}_k \in \mathcal{X}$*

$$\begin{aligned} \alpha_1(\|\boldsymbol{x}_k\|) \leq V(\boldsymbol{x}_k) \leq \alpha_2(\|\boldsymbol{x}_k\|) \\ V(\boldsymbol{x}_{k+1}) - V(\boldsymbol{x}_k) \leq -\alpha_3(\|\boldsymbol{x}_k\|) \ . \end{aligned}$$

Theorem 6 (Lyapunov stability, [160])**.** *Assume that the set $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is positive invariant for the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k)$ with $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and where the origin is an equilibrium of the system. If there exists a Lyapunov function in $\mathcal{X}$ for the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k)$, then the origin is asymptotically stable in $\mathcal{X}$. If $\mathcal{X} = \mathbb{R}^{n_x}$, the origin is globally asymptotically stable.*

A.2.2 Robust (input-to-state) Stability

In robust stability, one deals with uncertain systems of the form $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{e})$, where the signal $\boldsymbol{e} \in \mathcal{E}$ represents an uncertain or unknown part of the process. This can be, for instance, an unknown disturbance to the system or a mismatch between the true

process and the employed model. This mismatch can also be considered as an error, which is also why we denote the uncertainty with $\boldsymbol{e}$. One way to establish (robust) stability despite uncertainty is via the concept of input-to-state stability. Most of the following is motivated by [107].

Definition 15 (Input-to-state stability)**.** *A system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{e}_k)$ with the origin as an equilibrium is input-to-state stable (ISS) if there exist a $\mathcal{KL}$-function $\beta(\cdot,\cdot)$ and a $\mathcal{K}$-function $\gamma(\cdot)$ such that*

$$\|\boldsymbol{x}_k\| \leq \beta(\|\boldsymbol{x}_0\|, k) + \gamma\Big(\sup_{k\geq 0}\|\boldsymbol{e}_k\|\Big) \tag{A.1}$$

holds for all initial states $\boldsymbol{x}_0$, uncertainties $\boldsymbol{e}_k$, and for all $k \in \mathbb{N}_0$.

Input-to-state stability combines nominal stability as well as uniformly bounded influence of uncertainty in a single condition. It implies asymptotic stability of the undisturbed (nominal) system (with $\boldsymbol{e}_k \equiv 0$) and a bounded effect of the uncertainty on the state evolution. Furthermore, if the error signal $\boldsymbol{e}_k$ fades, the uncertain system asymptotically converges to the equilibrium.

Definition 16 (Robust positive and control invariant sets)**.**

- *A set $\mathcal{X}$ is robust positive invariant for the system $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{e})$ if $\boldsymbol{x}^+ \in \mathcal{X}$ for all $\boldsymbol{x} \in \mathcal{X}$ and all $\boldsymbol{e} \in \mathcal{E}$.*
- *A set $\mathcal{X}$ is robust control invariant for the system $\boldsymbol{x}^+ = f(\boldsymbol{x}, \boldsymbol{e})$ if for all $\boldsymbol{x} \in \mathcal{X}$ and all $\boldsymbol{e} \in \mathcal{E}$ there exists a $\boldsymbol{u} \in \mathcal{U}$ such that $\boldsymbol{x}^+ \in \mathcal{X}$.*

Definition 17 (ISS-Lyapunov function)**.** *Assume that $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is robust positive invariant for $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{e}_k)$. A function $V : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$ is called an ISS-Lyapunov function in $\mathcal{X}$ for $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{e}_k)$ if there exist functions $\alpha_1(\cdot), \alpha_2(\cdot) \in \mathcal{K}_\infty$, a continuous and positive definite function $\alpha_3(\cdot)$, and a $\mathcal{K}$-function $\lambda(\cdot)$ such that for all $\boldsymbol{x}_k \in \mathcal{X}$ and all $\boldsymbol{e}_k \in \mathcal{E}$*

$$\begin{aligned} \alpha_1(\|\boldsymbol{x}_k\|) \leq V(\boldsymbol{x}_k) &\leq \alpha_2(\|\boldsymbol{x}_k\|) \\ V(\boldsymbol{x}_{k+1}) - V(\boldsymbol{x}_k) &\leq -\alpha_3(\|\boldsymbol{x}_k\|) + \lambda(\|\boldsymbol{e}_k\|)\ . \end{aligned}$$

Theorem 7 (ISS via ISS-Lyapunov function, [107])**.** *Consider the system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{e}_k)$ with the origin as an equilibrium and where the uncertainty $\boldsymbol{e}_k \in \mathcal{E}$ is bounded by the compact set $\mathcal{E}$. If this system admits an ISS Lyapunov function in $\mathcal{X}$ w.r.t. $\boldsymbol{e}_k$, then it is input-to-state stable w.r.t. $\boldsymbol{e}_k$ in $\mathcal{X}$.*

A.3 Recursive Cholesky Factor Update

The Cholesky decomposition of a symmetric positive matrix $\boldsymbol{K} = \boldsymbol{K}^\mathsf{T} > 0$ is $\boldsymbol{K} = \boldsymbol{R}^\mathsf{T}\boldsymbol{R}$, where $\boldsymbol{R} = \mathrm{chol}(\boldsymbol{K})$ is an upper triangular matrix that is called the *Cholesky factor*. According to [145], the Cholesky factor can be updated recursively as presented in the following.

Consider the covariance matrix $\boldsymbol{K}$, represented in block form as

$$\begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{13} \\ \boldsymbol{K}_{13}^\mathsf{T} & \boldsymbol{K}_{33} \end{bmatrix}$$

and its Cholesky factor

$$\begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{13} \\ 0 & \boldsymbol{R}_{33} \end{bmatrix}.$$

Now, due to a new data point the updated covariance matrix is

$$\begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \boldsymbol{K}_{13} \\ \boldsymbol{K}_{12}^\mathsf{T} & \boldsymbol{K}_{22} & \boldsymbol{K}_{23} \\ \boldsymbol{K}_{13}^\mathsf{T} & \boldsymbol{K}_{23}^\mathsf{T} & \boldsymbol{K}_{33} \end{bmatrix},$$

which differs from the previous by insertion of a new row and column. The updated Cholesky factor

$$\begin{bmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} & \boldsymbol{S}_{13} \\ 0 & \boldsymbol{S}_{22} & \boldsymbol{S}_{23} \\ 0 & 0 & \boldsymbol{S}_{33} \end{bmatrix}$$

can be computed via

$$\begin{aligned} \boldsymbol{S}_{11} &= \boldsymbol{R}_{11} & \boldsymbol{S}_{22} &= \mathrm{chol}\Big(\boldsymbol{K}_{22} - \boldsymbol{S}_{12}^\mathsf{T}\boldsymbol{S}_{12}\Big) \\ \boldsymbol{S}_{12} &= \boldsymbol{R}_{11}^\mathsf{T} \backslash \boldsymbol{K}_{12} & \boldsymbol{S}_{23} &= \boldsymbol{S}_{22}^\mathsf{T} \backslash \Big(\boldsymbol{K}_{23} - \boldsymbol{S}_{12}^\mathsf{T}\boldsymbol{S}_{13}\Big) \\ \boldsymbol{S}_{13} &= \boldsymbol{R}_{13} & \boldsymbol{S}_{33} &= \mathrm{chol}\Big(\boldsymbol{R}_{33}^\mathsf{T}\boldsymbol{R}_{33} - \boldsymbol{S}_{23}^\mathsf{T}\boldsymbol{S}_{23}\Big)\,. \end{aligned} \tag{A.2}$$

On the other hand, if the current covariance matrix in block form

$$\begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \boldsymbol{K}_{13} \\ \boldsymbol{K}_{12}^\mathsf{T} & \boldsymbol{K}_{22} & \boldsymbol{K}_{23} \\ \boldsymbol{K}_{13}^\mathsf{T} & \boldsymbol{K}_{23}^\mathsf{T} & \boldsymbol{K}_{33} \end{bmatrix}$$

with Cholesky factor

$$\begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} & \boldsymbol{R}_{13} \\ 0 & \boldsymbol{R}_{22} & \boldsymbol{R}_{23} \\ 0 & 0 & \boldsymbol{R}_{33} \end{bmatrix}$$

is reduced by one row and column (removing one data point), such that we obtain

$$\begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{13} \\ \boldsymbol{K}_{13}^\mathsf{T} & \boldsymbol{K}_{33} \end{bmatrix},$$

the downdated Cholesky factor

$$\begin{bmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{13} \\ 0 & \boldsymbol{S}_{33} \end{bmatrix}$$

can be computed via

$$\begin{aligned} \boldsymbol{S}_{11} &= \boldsymbol{R}_{11} \\ \boldsymbol{S}_{13} &= \boldsymbol{R}_{13} \\ \boldsymbol{S}_{33} &= \mathrm{chol}\Big(\boldsymbol{R}_{23}^\mathsf{T}\boldsymbol{R}_{23} + \boldsymbol{R}_{33}^\mathsf{T}\boldsymbol{R}_{33}\Big) . \end{aligned} \tag{A.3}$$

A.4 Terminal Components Computation

In this section, we present a more detailed explanation of how to compute a terminal cost function $V_\mathrm{f}(\boldsymbol{x})$, as well as the associated terminal control law $\kappa_\mathrm{f}(\boldsymbol{x})$ and terminal region $\mathcal{X}_\mathrm{f}$ for model predictive control, as employed for the rGP-MPC scheme in Section 3.7. A variant of the presented approach was published for a different MPC scheme in [119].

A.4.1 Invariance of Terminal Region

We consider a linearized version of the discrete-time prediction model

$$\begin{aligned} \boldsymbol{x}^+ &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \text{s.t. } \boldsymbol{x} &\in \mathcal{X} \\ \boldsymbol{u} &\in \mathcal{U} \end{aligned}$$

with state $\boldsymbol{x} \in \mathbb{R}^{n_x}$, input $\boldsymbol{u} \in \mathbb{R}^{n_u}$, and matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of appropriate dimensions. As the terminal controller we choose the linear control law

$$\boldsymbol{u} = \kappa_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{k}^\mathsf{T}\boldsymbol{x}$$

and define $\boldsymbol{k}^\mathsf{T} := \boldsymbol{s}^\mathsf{T}\boldsymbol{P}$ with $\boldsymbol{s} \in \mathbb{R}^{n_x}$, $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_x}$, and $\boldsymbol{P} = \boldsymbol{P}^\mathsf{T} > 0$. For the terminal cost, we use the quadratic function

$$V_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x} .$$

The matrix $\boldsymbol{P}$ is determined such that $V_\mathrm{f}(\boldsymbol{x})$ is a Lyapunov function for the closed-loop system $\boldsymbol{x}^+ = (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{k}^\mathsf{T})\boldsymbol{x}$. Therefore, $\boldsymbol{P}$ must be positive definite for $V_\mathrm{f}(\boldsymbol{x})$ to be positive definite. Additionally, we need to ensure that $V_\mathrm{f}(\boldsymbol{x}^+) - V_\mathrm{f}(\boldsymbol{x})$ is negative

semidefinite, i.e.,

$$\begin{aligned} V_\mathrm{f}(\boldsymbol{x}) &\geq V_\mathrm{f}(\boldsymbol{x}^+) \\ \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x} &\geq (\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u})^\mathsf{T}\boldsymbol{P}(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u}) \\ 0 &\geq \boldsymbol{x}^\mathsf{T}\Big[\big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{k}^\mathsf{T}\big)^\mathsf{T}\boldsymbol{P}\big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{k}^\mathsf{T}\big)-\boldsymbol{P}\Big]\boldsymbol{x} \\ 0 &\geq \big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{k}^\mathsf{T}\big)^\mathsf{T}\boldsymbol{P}\big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{k}^\mathsf{T}\big)-\boldsymbol{P} \\ 0 &\geq \big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\boldsymbol{P}\big)^\mathsf{T}\boldsymbol{P}\big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\boldsymbol{P}\big)-\boldsymbol{P}\ . \end{aligned}$$

Substituting $\boldsymbol{G} := \boldsymbol{P}^{-1}$ (this implies $\boldsymbol{G} = \boldsymbol{G}^\mathsf{T} > 0$) gives

$$0 \geq \big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\boldsymbol{G}^{-1}\big)^\mathsf{T}\boldsymbol{G}^{-1}\big(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\boldsymbol{G}^{-1}\big)-\boldsymbol{G}^{-1}$$

and multiplied with $\boldsymbol{G}$ from both sides yields

$$\begin{aligned} 0 &\geq \big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big)^\mathsf{T}\boldsymbol{G}^{-1}\big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big)-\boldsymbol{G} \\ 0 &\leq \boldsymbol{G}-\big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big)^\mathsf{T}\boldsymbol{G}^{-1}\big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big)\ . \end{aligned} \tag{A.4}$$

The sought variables are $\boldsymbol{G}$ and $\boldsymbol{s}$ and appear in a nonlinear manner in (A.4), which makes their determination difficult. Fortunately, (A.4) can be reformulated into a linear matrix inequality (LMI) using the *Schur complement*.

Lemma 3 (Schur complement, [25])**.** *For a matrix*

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{\Pi} \\ \boldsymbol{\Pi}^\mathsf{T} & \boldsymbol{C} \end{bmatrix}$$

with $\det(\boldsymbol{D}) \neq 0$*, the matrix*

$$\boldsymbol{S} = \boldsymbol{C} - \boldsymbol{\Pi}^\mathsf{T}\boldsymbol{D}^{-1}\boldsymbol{\Pi}$$

is called the Schur complement *of* $\boldsymbol{D}$ *in* $\boldsymbol{X}$*. Moreover, the following properties hold:*

$$\boldsymbol{X} > 0, \text{ if and only if } \boldsymbol{D} > 0 \text{ and } \boldsymbol{S} > 0. \tag{A.5a}$$

$$\text{If } \boldsymbol{D} > 0, \text{ then } \boldsymbol{X} \geq 0, \text{ if and only if } \boldsymbol{S} \geq 0. \tag{A.5b}$$

(A.5b) can be applied to (A.4) with $\boldsymbol{C} = \boldsymbol{D} = \boldsymbol{G}$ and $\boldsymbol{\Pi} = \big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big)$. Then,

$$\begin{bmatrix} \boldsymbol{G} & \big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big) \\ \big(\boldsymbol{A}\boldsymbol{G}+\boldsymbol{B}\boldsymbol{s}^\mathsf{T}\big)^\mathsf{T} & \boldsymbol{G} \end{bmatrix} \geq 0 \tag{A.6}$$

is equivalent to (A.4) and hence, can be used instead of it. The LMI (A.6) is affine

in the unknowns $\boldsymbol{G}$ and $\boldsymbol{s}$ and can therefore be solved efficiently. A solution yields the matrix $\boldsymbol{P} = \boldsymbol{G}^{-1}$ that makes $V_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x}$ a Lyapunov function for the closed-loop system and defines at the same time together with $\boldsymbol{s}$ the terminal control law $\kappa_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{s}^\mathsf{T}\boldsymbol{P}\boldsymbol{x}$.

Now, one can choose a level set of the Lyapunov function as the terminal region

$$\mathcal{X}_\mathrm{f} := \left\{\boldsymbol{x} \in \mathbb{R}^{n_x} : \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x} \leq 1\right\} . \tag{A.7}$$

Then, $\boldsymbol{u} = \kappa_\mathrm{f}(\boldsymbol{x})$ is a stabilizing terminal controller on $\mathcal{X}_\mathrm{f}$. Note that (A.7) describes an ellipsoid in the state space $\mathbb{R}^{n_x}$, whose size is determined by $\boldsymbol{P}$.

A.4.2 Constraint Satisfaction

State constraint satisfaction of the terminal region $\mathcal{X}_\mathrm{f}$ translates to

$$\mathcal{X}_\mathrm{f} \subseteq \mathcal{X} \tag{A.8a}$$

and since $\boldsymbol{u} = \boldsymbol{k}^\mathsf{T}\boldsymbol{x} \in \mathcal{U}$ for all $\boldsymbol{x} \in \mathcal{X}_\mathrm{f}$, input constraint satisfaction translates to

$$\boldsymbol{k}^\mathsf{T}\mathcal{X}_\mathrm{f} \subseteq \mathcal{U} . \tag{A.8b}$$

These are general set conditions and usually hard to verify because one would have to check if they hold for every point in $\mathcal{X}_\mathrm{f}$. In the case of closed convex sets, such as polyhedral sets

$$\mathcal{X} = \left\{\boldsymbol{x} \in \mathbb{R}^{n_x} : \boldsymbol{q}_i^\mathsf{T}\boldsymbol{x} \leq r_i,\ i = 1, \ldots, n_\mathcal{X}\right\} \tag{A.9a}$$

$$\mathcal{U} = \left\{\boldsymbol{u} \in \mathbb{R}^{n_u} : \boldsymbol{v}_l^\mathsf{T}\boldsymbol{u} \leq t_l,\ l = 1, \ldots, n_\mathcal{U}\right\} \tag{A.9b}$$

with the numbers of inequalities denoted by $n_\mathcal{X}$, $n_\mathcal{U} \in \mathbb{N}$, this problem can be circumvented using the *support function* concept which allows us to reformulate conditions (A.8) as LMIs [23, 95].

Definition 18 (Support function)**.** *The support function of a closed convex set $\boldsymbol{S} \subset \mathbb{R}^{n_x}$, evaluated at a point $\boldsymbol{\eta} \in \mathbb{R}^{n_x}$, is*

$$h_{\boldsymbol{S}}(\boldsymbol{\eta}) = \sup_{\boldsymbol{x} \in \boldsymbol{S}} \boldsymbol{\eta}^\mathsf{T}\boldsymbol{x} .$$

For a given vector $\boldsymbol{\eta}$, the support function chooses the $\boldsymbol{x} \in \boldsymbol{S}$ for which the scalar product $\boldsymbol{\eta}^\mathsf{T}\boldsymbol{x}$ is maximal. In particular, it yields the supremum of all possible scalar products of the points of the set $\boldsymbol{S}$ with a given vector $\boldsymbol{\eta}$. Any nonempty closed and convex set $\boldsymbol{S}$ can be uniquely represented by its support function $h_{\boldsymbol{S}}(\cdot)$ via

$$\boldsymbol{S} = \left\{\boldsymbol{x} \in \mathbb{R}^{n_x} : \boldsymbol{\eta}^\mathsf{T}\boldsymbol{x} \leq h_{\boldsymbol{S}}(\boldsymbol{\eta}),\ \forall \boldsymbol{\eta} \in \mathbb{R}^{n_x}\right\} .$$

For instance, the support function of an ellipsoidal set as defined by (A.7) is given by

$h_{\mathcal{X}_\mathrm{f}}(\boldsymbol{\eta}) = \sqrt{\boldsymbol{\eta}^\mathsf{T}\boldsymbol{P}^{-1}\boldsymbol{\eta}}$.

A.4.2.1 Satisfaction of State Constraints

Given the two closed convex sets $\mathcal{X}_\mathrm{f}$ and $\mathcal{X}$, checking the state constraint condition $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$ is equivalent to checking if $h_{\mathcal{X}_\mathrm{f}}(\boldsymbol{\eta}) \leq h_{\mathcal{X}}(\boldsymbol{\eta})$ holds for all $\boldsymbol{\eta} \in \mathbb{R}^{n_x}$. In case that $\mathcal{X}$ is a polyhedral set as defined in (A.9), checking $\mathcal{X}_\mathrm{f} \subseteq \mathcal{X}$ simplifies even further to verifying that $h_{\mathcal{X}_\mathrm{f}}(\boldsymbol{q}_i) \leq r_i$ holds for all $i \in \{1, \ldots, n_\mathcal{X}\}$. Putting all this together, we arrive at the equivalence

$$\mathcal{X}_\mathrm{f} \subseteq \mathcal{X} \quad \Leftrightarrow \quad \sqrt{\boldsymbol{q}_i^\mathsf{T}\boldsymbol{P}^{-1}\boldsymbol{q}_i} \leq r_i, \quad \forall i \in \{1, \ldots, n_\mathcal{X}\} \ .$$

The right hand side inequality can be reformulated by means of the Schur complement into the set of LMIs

$$\begin{bmatrix} \boldsymbol{G} & (\boldsymbol{G}\boldsymbol{q}_i) \\ (\boldsymbol{G}\boldsymbol{q}_i)^\mathsf{T} & r_i^2 \end{bmatrix} \geq 0, \quad \forall i \in \{1, \ldots, n_\mathcal{X}\} \ , \tag{A.10}$$

with the same $\boldsymbol{G}$ as in (A.6).

A.4.2.2 Satisfaction of Input Constraints

The same procedure can be applied to transform the input constraint condition into a set of LMIs. First, note that the inclusion $\boldsymbol{k}^\mathsf{T}\mathcal{X}_\mathrm{f} \subseteq \mathcal{U}$ holds if the inequality $h_{\boldsymbol{k}^\mathsf{T}\mathcal{X}_\mathrm{f}}(\boldsymbol{\eta}) \leq h_{\mathcal{U}}(\boldsymbol{\eta})$ is satisfied for all $\boldsymbol{\eta} \in \mathbb{R}^{n_u}$. Since $\mathcal{U}$ is polyhedral, this simplifies to $h_{\boldsymbol{k}^\mathsf{T}\mathcal{X}_\mathrm{f}}(\boldsymbol{v}_l) \leq t_l$ for all $l \in \{1, \ldots, n_\mathcal{U}\}$. Applying the identity $h_{XY}(\boldsymbol{\eta}) = h_Y(X^\mathsf{T}\boldsymbol{\eta})$ [95] yields $h_{\mathcal{X}_\mathrm{f}}(\boldsymbol{k}\boldsymbol{v}_l) \leq t_l$ for all $l \in \{1, \ldots, n_\mathcal{U}\}$ and we arrive at the equivalence

$$\boldsymbol{k}^\mathsf{T}\mathcal{X}_\mathrm{f} \subseteq \mathcal{U} \quad \Leftrightarrow \quad \sqrt{(\boldsymbol{k}^\mathsf{T}\boldsymbol{v}_l)^\mathsf{T}\boldsymbol{P}^{-1}(\boldsymbol{k}^\mathsf{T}\boldsymbol{v}_l)} \leq t_l, \quad \forall l \in \{1, \ldots, n_\mathcal{U}\} \ .$$

Again, the right hand side inequality can be reformulated by means of the Schur complement into the set of LMIs

$$\begin{aligned} &\begin{bmatrix} \boldsymbol{P}^{-1} & (\boldsymbol{P}^{-1}\boldsymbol{k}^\mathsf{T}\boldsymbol{v}_l) \\ (\boldsymbol{P}^{-1}\boldsymbol{k}^\mathsf{T}\boldsymbol{v}_l)^\mathsf{T} & t_l^2 \end{bmatrix} \\ =&\begin{bmatrix} \boldsymbol{G} & (\boldsymbol{s}^\mathsf{T}\boldsymbol{v}_l) \\ (\boldsymbol{s}^\mathsf{T}\boldsymbol{v}_l)^\mathsf{T} & t_l^2 \end{bmatrix} \geq 0, \quad \forall l \in \{1, \ldots, n_\mathcal{U}\} \ . \end{aligned} \tag{A.11}$$

A.4.3 Optimization Problem

In order to determine the terminal cost function $V_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T}\boldsymbol{P}\boldsymbol{x}$, together with the terminal control law $\kappa_\mathrm{f}(\boldsymbol{x}) = \boldsymbol{k}^\mathsf{T}\boldsymbol{x}$ and corresponding terminal region $\mathcal{X}_\mathrm{f}$, (A.6), (A.10) and (A.11) have to be satisfied. To this end we use them as constraints to the opti-

mization problem

$$\begin{aligned}
&\max_{\boldsymbol{G},\boldsymbol{s}} \ \log\big(\det(\boldsymbol{G})\big) \\
&\text{s.t. } \boldsymbol{G} = \boldsymbol{G}^{\mathsf{T}} > 0 \\
&\quad \begin{bmatrix} \boldsymbol{G} & \big(\boldsymbol{A}\boldsymbol{G} + \boldsymbol{B}\boldsymbol{s}^{\mathsf{T}}\big)^{\mathsf{T}} \\ \big(\boldsymbol{A}\boldsymbol{G} + \boldsymbol{B}\boldsymbol{s}^{\mathsf{T}}\big) & \boldsymbol{G} \end{bmatrix} \geq 0 \\
&\quad \begin{bmatrix} \boldsymbol{G} & (\boldsymbol{G}\boldsymbol{q}_i) \\ (\boldsymbol{G}\boldsymbol{q}_i)^{\mathsf{T}} & r_i^2 \end{bmatrix} \geq 0, \quad \forall i \in \{1, \dots, n_{\mathcal{X}}\} \\
&\quad \begin{bmatrix} \boldsymbol{G} & (\boldsymbol{s}^{\mathsf{T}}\boldsymbol{v}_l) \\ (\boldsymbol{s}^{\mathsf{T}}\boldsymbol{v}_l)^{\mathsf{T}} & t_l^2 \end{bmatrix} \geq 0, \quad \forall l \in \{1, \dots, n_{\mathcal{U}}\} \ ,
\end{aligned} \tag{A.12}$$

whose objective is to maximize $\mathcal{X}_\mathrm{f}$ by means of maximizing $\log(\det(\boldsymbol{G}))$ (see [25, 119]). (A.12) is a semidefinite program and can be efficiently solved employing, e.g. MATLAB together with the YALMIP toolbox and the `solvesdp` command.

A.5 Extremum Seeking Control

In this section, we provide the derivation of the extremum seeking control approach according to [96] as shown in Fig. A.1.

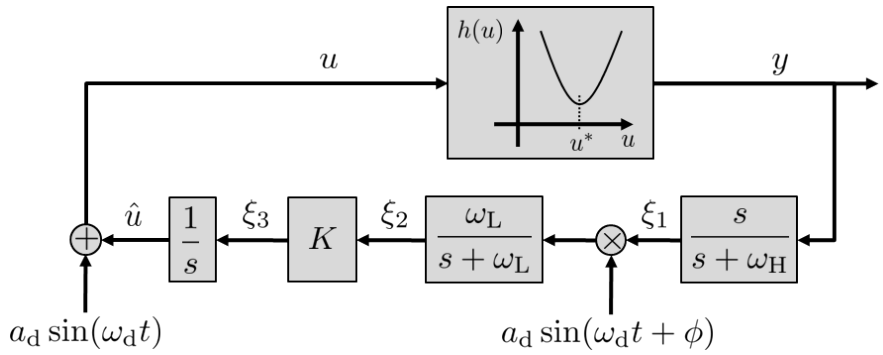


Figure A.1: Block diagram of the extremum seeking control approach.

Let $h(u) : \mathbb{R} \to \mathbb{R}$ be a smooth and locally convex function for which there exists a minimum h^* at $u = u^*$. The objective of the extremum seeking controller is to find this minimum.

The Taylor series expansion up to order two of $h(u)$ around an evaluation point $\hat{u}$ is

$$h(u) \approx h(\hat{u}) + \frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} \underbrace{(u - \hat{u})}_{\Delta u} + \frac{\mathrm{d}^2 h}{\mathrm{d}u^2}\Big|_{\hat{u}} \underbrace{(u - \hat{u})^2}_{\Delta u^2} \ .$$

Now, in extremum seeking control, the point $\hat{u}$ is disturbed by a so-called sinusoidal *dither* signal $d(t) = a_d \sin(\omega_d t)$. Then, the input becomes $u(t) = \hat{u} + a_d \sin(\omega_d t)$ and

one obtains for small dither amplitudes a_d

$$h\big(\hat{u} + a_d \sin(\omega_d t)\big) \approx h(\hat{u}) + \frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} a_d \sin(\omega_d t) + \frac{\mathrm{d}^2 h}{\mathrm{d}u^2}\Big|_{\hat{u}} a_d^2 \sin^2(\omega_d t) \ .$$

The resulting output signal of $h(u(t))$ is passed through a high pass filter $G_{\mathrm{H}}(s) = \frac{s}{s+\omega_{\mathrm{H}}}$ to eliminate any constant offsets, which yields

$$\xi_1(t) \approx \frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} a_d \sin(\omega_d t + \phi) + \frac{\mathrm{d}^2 h}{\mathrm{d}u^2}\Big|_{\hat{u}} a_d^2 \sin^2(\omega_d t + \phi)$$

and where $\phi = \arg\big(G_{\mathrm{H}}(\mathrm{j}\omega_d)\big)$ is the phase shift introduced by the high pass filter. The signal $\xi_1(t)$ is then multiplied by the phase shifted dither signal $a_d \sin(\omega_d t + \phi)$, obtaining

$$\frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} a_d^2 \sin^2(\omega_d t + \phi) + \frac{\mathrm{d}^2 h}{\mathrm{d}u^2}\Big|_{\hat{u}} a_d^3 \sin^3(\omega_d t + \phi) \ ,$$

which can be expressed via $\sin^2(x) = \frac{1}{2}(1 - \cos(2x))$ as

$$\frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} a_d^2 \frac{1}{2}\big[1 - \cos\big(2(\omega_d t + \phi)\big)\big] + \frac{\mathrm{d}^2 h}{\mathrm{d}u^2}\Big|_{\hat{u}} a_d^3 \sin^3(\omega_d t + \phi) \ .$$

This signal is then smoothed by the low pass filter $G_{\mathrm{L}}(s) = \frac{\omega_{\mathrm{L}}}{s+\omega_{\mathrm{L}}}$, which in turn leads to the signal $\xi_2(t)$. Since $\cos(\cdot)$ and $\sin^3(\cdot)$ have zero mean, one obtains for small filter frequencies ω_{L}

$$\lim_{t\to\infty} \xi_2(t) = \frac{a_d^2}{2} \frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}}$$

at the current operating point $\hat{u}$. Choosing $K := {}^{-2}/_{a_d^2}$ in Fig. A.1, one obtains

$$\lim_{t\to\infty} \xi_3(t) = -\frac{\mathrm{d}h}{\mathrm{d}u}\Big|_{\hat{u}} \ ,$$

i.e. $\xi_3(t)$ tends to the negative gradient of $h(\hat{u})$. Hence, $\xi_3(t)$ can be used subsequently to implement a gradient descent approach, realized by the integration of $\xi_3(t)$, turning $\hat{u}$ into an estimation of the minimizer u^*.

Bibliography

[1] P. Abrahamsen. *A Review of Gaussian Random Fields and Correlation Functions*. Norsk Regnesentral/Norwegian Computing Center Oslo, 1997.

[2] E. R. Ackermann, J. P. De Villiers, and P. Cilliers. Nonlinear dynamic systems modeling using Gaussian processes: Predicting ionospheric total electron content over South Africa. *Journal of Geophysical Research: Space Physics*, 116(A10), 2011.

[3] V. Adetola and M. Guay. Adaptive output feedback extremum seeking receding horizon control of linear systems. *Journal of Process Control*, 16(5):521–533, 2006.

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

[5] B. M. Åkesson and H. T. Toivonen. A neural network model predictive controller. *Journal of Process Control*, 16(9):937–946, 2006.

[6] T. Albrecht, P. Grütter, D. Horne, and D. Rugar. Frequency modulation detection using high-Q cantilevers for enhanced force microscope sensitivity. *Journal of Applied Physics*, 69(2):668–673, 1991.

[7] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings. On the inherent robustness of optimal and suboptimal nonlinear MPC. *Systems & Control Letters*, 106:68–78, 2017.

[8] A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg, and E. Almomani. A survey of phishing email filtering techniques. *IEEE Communications Surveys & Tutorials*, 15(4):2070–2090, 2013.

[9] M. Araki and H. Taguchi. Two-degree-of-freedom PID controllers. *International Journal of Control, Automation, and Systems*, 1(4):401–411, 2003.

[10] K. B. Ariyur and M. Krstić. *Real-Time Optimization by Extremum-Seeking Control*. John Wiley and Sons Inc., 2003.

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

[12] I. Avcibaş, B. Sankur, and K. Sayood. Statistical evaluation of image quality measures. *Journal of Electronic Imaging*, 11(2):206–223, 2002.

[13] K. Ažman and J. Kocijan. Non-linear model predictive control for models with local information and uncertainties. *Transactions of the Institute of Measure-*

ment and Control, 30(5):371–396, 2008.

[14] O. Bastani. Safe planning via model predictive shielding. *arXiv preprint arXiv:1905.10691*, 2019.

[15] F. Berkenkamp. *Safe Exploration in Reinforcement Learning: Theory and Applications in Robotics*. PhD thesis, ETH Zurich, 2019.

[16] F. Berkenkamp, A. Krause, and A. P. Schoellig. Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016.

[17] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. In *Conference on Decision and Control (CDC)*, pages 4661–4666. IEEE, 2016.

[18] F. Berkenkamp and A. P. Schoellig. Learning-based robust control: Guaranteeing stability while improving performance. In *International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[19] F. Berkenkamp, A. P. Schoellig, and A. Krause. Safe controller optimization for quadrotors with Gaussian processes. In *International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

[20] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances Neural Information Processing Systems*, pages 908–918, Red Hook, USA, 2017. Curran Associates Inc.

[21] J. Bethge, B. Morabito, J. Matschek, and R. Findeisen. Multi-mode learning supported model predictive control with guarantees. In *Conference on Nonlinear Model Predictive Control (NMPC)*, pages 616–621. IFAC, 2018.

[22] G. Binnig, H. Rohrer, C. Gerber, and E. Weibel. Surface studies by scanning tunneling microscopy. *Physical Review Letters*, 49(1):57, 1982.

[23] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. Systems & Control. Birkhäuser, Boston, 2008.

[24] J. Boedecker, J. T. Springenberg, J. Wülfing, and M. Riedmiller. Approximate real-time optimal control based on sparse Gaussian process models. In *Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–8. IEEE, 2014.

[25] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

[26] E. Bradford and L. Imsland. Stochastic nonlinear model predictive control using Gaussian processes. In *European Control Conference (ECC)*, pages 1027–1034, 2018.

[27] S. Brahim-Belhouari and A. Bermak. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, 2004.

[28] D. A. Bristow, M. Tharayil, and A. G. Alleyne. A survey of iterative learning control. *IEEE Control Systems Magazine*, 26(3):96–114, 2006.

[29] S. Brunton, C. Rowley, S. Kulkarni, and C. Clarkson. Maximum power point tracking for photovoltaic optimization using ripple-based extremum seeking control. *IEEE Transactions on Power Electronics*, 25(10):2531–2540, 2010.

[30] G. Cao, E. M.-K. Lai, and F. Alam. Gaussian process based model predictive control for linear time varying systems. In *International Workshop on Advanced Motion Control (AMC)*, pages 251–256. IEEE, 2016.

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

[32] G. Cao, E. M.-K. Lai, and F. Alam. Gaussian process model predictive control of unknown non-linear systems. *IET Control Theory & Applications*, 11(5):703–713, 2017.

[33] A. Carron, M. Todescato, R. Carli, L. Schenato, and G. Pillonetto. Machine learning meets Kalman filtering. In *Conference on Decision and Control (CDC)*, pages 4594–4599. IEEE, 2016.

[34] C. Centioli, F. Iannone, G. Mazza, M. Panella, L. Pangione, S. Podda, A. Tuccillo, V. Vitale, and L. Zaccarian. Maximization of the lower hybrid power coupling in the Frascati Tokamak Upgrade via extremum seeking. *Control Engineering Practice*, 16(12):1468–1478, 2008.

[35] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari. Approximating explicit model predictive control using constrained neural networks. In *American Control Conference (ACC)*, pages 1520–1527. IEEE, 2018.

[36] J.-Y. Choi, M. Krstić, K. B. Ariyur, and J. S. Lee. Extremum seeking control for discrete-time systems. *IEEE Transactions on Automatic Control*, 47(2):318–323, 2002.

[37] G. M. Clayton, S. Tien, K. K. Leang, Q. Zou, and S. Devasia. A review of feedforward control approaches in nanopositioning for high-speed SPM. *Journal of Dynamic Systems, Measurement, and Control*, 131(6):061101, 2009.

[38] J. Cochran and M. Krstić. Nonholonomic source seeking with tuning of angular velocity. *IEEE Transactions on Automatic Control*, 54(4):717–731, 2009.

[39] G. De Nicolao, L. Magni, and R. Scattolini. Stability and robustness of nonlinear receding horizon control. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*, Progress in Systems and Control Theory, pages 3–22. Birkhäuser Basel, 2000.

[40] G. De Nicolao and G. Pillonetto. A new kernel-based approach for system identification. In *American Control Conference (ACC)*, pages 4510–4516. IEEE,

2008.

[41] M. P. Deisenroth. *Efficient Reinforcement Learning using Gaussian Processes*, volume 9. KIT Scientific Publishing, 2010.

[42] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[43] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472, 2011.

[44] S. Devasia, E. Eleftheriou, and S. R. Moheimani. A survey of control issues in nanopositioning. *IEEE Transactions on Control Systems Technology*, 15(5):802–823, 2007.

[45] A. Doerr, D. Nguyen-Tuong, A. Marco, S. Schaal, and S. Trimpe. Model-based policy search for automatic tuning of multivariate PID controllers. In *International Conference on Robotics and Automation (ICRA)*, pages 5295–5301. IEEE, 2017.

[46] D. Dong. Mine gas emission prediction based on Gaussian process model. *Procedia Engineering*, 45:334–338, 2012.

[47] C. S. Draper and Y. T. Li. *Principles of Optimalizing Control Systems and an Application to the Internal Combustion Engine*. American Society of Mechanical Engineers, 1951.

[48] D. Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014.

[49] D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, volume 28, pages 1166–1174, 2013.

[50] M. L. Eaton. *Multivariate Statistics: A Vector Space Approach*. John Wiley & Sons, New York, 1983.

[51] E. Eleftheriou and S. Moheimani. *Control Technologies for Emerging Micro and Nanoscale Systems*. Springer-Verlag Berlin Heidelberg, 2011.

[52] Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University of Jerusalem Jerusalem, Israel, 2005.

[53] T. Faulwasser. *Optimization-based Solutions to Constrained Trajectory-Tracking and Path-Following Problems*, volume 3 of *Contributions in Systems Theory and Automatic Control*. Shaker, Aachen, 2013.

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

[55] R. Findeisen. *Nonlinear Model Predictive Control: A Sampled Data Feedback Perspective*. PhD thesis, Universität Stuttgart, 2005.

[56] R. Findeisen, M. A. Grover, C. Wagner, M. Maiworm, R. Temirov, F. S. Tautz, M. V. Salapaka, S. Salapaka, R. D. Braatz, and S. O. R. Moheimani. Control on a molecular scale: A perspective. In *American Control Conference (ACC)*, pages 3069–3082. IEEE, 2016.

[57] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[58] N. Fournier, C. Wagner, C. Weiss, R. Temirov, and F. Tautz. Force-controlled lifting of molecular wires. *Physical Review B*, 84(3):035435, 2011.

[59] F. Fuchs, F. Caffy, R. Demadrille, T. Meilin, and B. Greivin. High-resolution Kelvin probe force microscopy imaging of interface dipoles and photogenerated charges in organic donor–acceptor photovoltaic blends. *ACS nano*, 10(1):739–746, 2016.

[60] A. Ghaffari, M. Krstić, and D. Nešić. Multivariable Newton-based extremum seeking. *Automatica*, 48(8):1759–1767, 2012.

[61] F. J. Giessibl. Forces and frequency shifts in atomic-resolution dynamic-force microscopy. *Physical Review B*, 56(24):16010, 1997.

[62] F. J. Giessibl. Advances in atomic force microscopy. *Reviews of Modern Physics*, 75(3):949, 2003.

[63] F. J. Giessibl, S. Hembacher, M. Herz, C. Schiller, and J. Mannhart. Stability considerations and implementation of cantilevers allowing dynamic force microscopy with optimal resolution: the qPlus sensor. *Nanotechnology*, 15(2):S79, 2004.

[64] A. Grancharova, J. Kocijan, and T. A. Johansen. Explicit stochastic nonlinear predictive control based on Gaussian process models. In *European Control Conference (ECC)*, pages 2340–2347. IEEE, 2007.

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

[66] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

[67] M. F. Green, C. Wagner, P. Leinen, T. Deilmann, P. Krüger, M. Rohlfing, F. S. Tautz, and R. Temirov. Scanning quantum dot microscopy: A quantitative method to measure local electrostatic potential near surfaces. *Japanese Journal of Applied Physics*, 55(8S1):08NA04, 2016.

[68] G. Gregorčič and G. Lightbody. Gaussian process approach for modelling of nonlinear systems. *Engineering Applications of Artificial Intelligence*, 22(4-5):522–533, 2009.

[69] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel. Nominally robust model predictive control with state constraints. *IEEE Transactions on Automatic Control*, 52(10):1856–1870, 2007.

[70] L. Gross, F. Mohn, P. Liljeroth, J. Repp, F. J. Giessibl, and G. Meyer. Measuring the charge state of an adatom with noncontact atomic force microscopy. *Science*, 324(5933):1428–1431, 2009.

[71] G. Gu, L. Zhu, C. Su, H. Ding, and S. Fatikow. Modeling and control of piezo-actuated nanopositioning stages: A survey. *IEEE Transactions on Automation Science and Engineering*, 13(1):313–332, 2016.

[72] M. Guay and T. Zhang. Adaptive extremum seeking control of nonlinear dynamic systems with parametric uncertainties. *Automatica*, 39(7):1283–1293, 2003.

[73] C. Hackl, F. Larcher, A. Dotlinger, and R. Kennel. Is multiple-objective model-predictive control optimal? In *IEEE International Symposium on Sensorless Control for Electrical Drives and Predictive Control of Electrical Drives and Power Electronics*, pages 1–8, 2013.

[74] B. Helfrich, C. Lee, D. Bristow, X. Xiao, J. Dong, A. Alleyne, S. Salapaka, and P. Ferreira. Combined H_∞-feedback control and iterative learning control design with application to nanopositioning systems. *IEEE Interactions on Control Systems Technology*, 18(2):336–351, 2010.

[75] E. Hernandez Vargas, P. Colaneri, and R. Middleton. Switching strategies to mitigate HIV mutation. *IEEE Transactions on Control Systems Technology*, 22(4):1623–1628, 2014.

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

[77] L. Hewing and M. N. Zeilinger. Cautious model predictive control using Gaussian process regression. *arXiv preprint arXiv:1705.10702*, 2017.

[78] Y. Hu, V. Pecunia, L. Jiang, C.-A. Di, X. Gao, and H. Sirringhaus. Scanning Kelvin probe microscopy investigation of the role of minority carriers on the switching characteristics of organic field-effect transistors. *Advanced Materials*, 28(23):4713–4719, 2016.

[79] M. F. Huber. Recursive Gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45:85–91, 2014.

[80] R. Isermann. *Digital Control Systems*. Springer-Verlag Berlin Heidelberg, 1981.

[81] R. E. Kalman. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana (2)*, 5:102–119, 1960.

[82] S. Kamthe and M. P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv preprint arXiv:1706.06491*, 2017.

[83] B. Karg and S. Lucia. Deep learning-based embedded mixed-integer model predictive control. In *European Control Conference (ECC)*, pages 2075–2080, 2018.

[84] E. D. Klenske, M. N. 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.

[85] J. Ko, D. J. Klein, D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *International Conference on Robotics and Automation (ICRA)*, pages 742–747. IEEE, 2007.

[86] N. Kocić, P. Weiderer, S. Keller, S. Decurtins, S.-X. Liu, and J. Repp. Periodic charging of individual molecules coupled to the motion of an atomic force microscopy tip. *Nano Letters*, 15(7):4406–4411, 2015.

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

[88] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.

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

[90] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *American Control Conference (ACC)*, volume 3, pages 2214–2219. IEEE, 2004.

[91] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar. Predictive control with Gaussian process models. In *The IEEE Region 8 EUROCON. Computer as a Tool*, pages 352–356. IEEE, 2003.

[92] J. Kocijan and J. Přikryl. Soft sensor for faulty measurements detection and reconstruction in urban traffic. In *Mediterranean Electrotechnical Conference*, pages 172–177. IEEE, 2010.

[93] M. Kögel and R. Findeisen. Stability of NMPC with cyclic horizons. In *Symposium on Nonlinear Control Systems*, pages 809–814, 2013.

[94] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-based model predictive control for safe exploration and reinforcement learning. *arXiv preprint arXiv:1803.08287*, 2018.

[95] I. Kolmanovsky and E. G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4(4):317–367, 1998.

[96] M. Krstić and H.-H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595–601, 2000.

[97] N. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

[98] M. Lazáro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

[99] K. K. Leang and S. Devasia. Design of hysteresis-compensating iterative learning control for piezo-positioners: Application to atomic force microscopes. *Mechatronics*, 16(3-4):141–158, 2006.

[100] K. K. Leang, Q. Zou, and S. Devasia. Feedforward control of piezoactuators in atomic force microscope systems. *IEEE Control Systems Magazine*, 29(1):70–82, 2009.

[101] M. Leblanc. Sur l'electrification des chemins de fer au moyen de courants alternatifs de frequence elevee. *Revue Générale de l'Électricité*, 12(8):275–277, 1922.

[102] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pages 609–616, 2009.

[103] A. Levin and K. Narendra. Identification of nonlinear dynamical systems using neural networks. In *Neural Systems for Control*, pages 129–160. Elsevier, 1997.

[104] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6):76–105, 2012.

[105] R. Leyva, C. Alonso, I. Queinnec, A. Cid-Pastor, D. Lagrange, and L. Martinez-Salamero. MPPT of photovoltaic systems using extremum–seeking control. *IEEE Transactions on Aerospace and Electronic Systems*, 42(1):249–258, 2006.

[106] 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.

[107] D. Limón, T. Alamo, D. Raimondo, D. M. De La Peña, J. Bravo, A. Ferramosca, and E. Camacho. Input-to-state stability: A unifying framework for robust model predictive control. In *Nonlinear Model Predictive Control*, pages 1–26. Springer, 2009.

[108] D. Limón, T. Alamo, F. Salas, and E. F. Camacho. On the stability of constrained MPC without terminal constraint. *IEEE Transactions on Automatic Control*, 51(5):832–836, 2006.

[109] S.-J. Liu and M. Krstić. *Stochastic Averaging and Stochastic Extremum Seeking*. Springer-Verlag London, 2012.

[110] S.-J. Liu and M. Krstić. Newton-based stochastic extremum seeking. *Automat-*

ica, 50(3):952–961, 2014.

[111] L. Ljung. *System Identification (2nd Ed.): Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[112] J. Löfberg. Oops! I cannot do it again: Testing for recursive feasibility in MPC. *Automatica*, 48(3):550–555, 2012.

[113] S. Lucia, T. Finkler, and S. Engell. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9):1306–1319, 2013.

[114] S. Lucia and B. Karg. A deep learning-based approach to robust nonlinear model predictive control. In *Conference on Nonlinear Model Predictive Control (NMPC)*, pages 610–615, 2018.

[115] S. Lucia, M. Kögel, P. Zometa, D. E. Quevedo, and R. Findeisen. Predictive control, embedded cyberphysical systems and systems of systems – a perspective. *Annual Reviews in Control*, 41:193–207, 2016.

[116] D. G. Luenberger. *Optimization by Vector Space Methods*. Decision and Control. Wiley, New York, 1969.

[117] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, 2002.

[118] J. M. Maciejowski and X. Yang. Fault tolerant control using Gaussian processes and model predictive control. In *Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 1–12. IEEE, 2013.

[119] M. Maiworm, T. Bäthge, and R. Findeisen. Scenario-based model predictive control: Recursive feasibility and stability. In *International Symposium on Advanced Control of Chemical Processes (ADCHEM)*, pages 50–56. IFAC, 2015.

[120] M. Maiworm, D. Limón, and R. Findeisen. Online learning-based model predictive control with Gaussian process models and stability guarantees. *International Journal of Robust and Nonlinear Control*, pages 1–28, 2020.

[121] M. Maiworm, D. Limón, J. M. Manzano, and R. Findeisen. Stability of Gaussian process learning based output feedback model predictive control. In *Conference on Nonlinear Model Predictive Control (NMPC)*, pages 551–557. IFAC, 2018.

[122] M. Maiworm, C. Wagner, R. Temirov, F. S. 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.

[123] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In *International Conference on Robotics and Automation (ICRA)*, pages 1557–1563. IEEE, 2017.

[124] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. Automatic LQR tuning

based on Gaussian process global optimization. In *International Conference on Robotics and Automation (ICRA)*, pages 270–277. IEEE, 2016.

[125] 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. IEEE, 2020.

[126] J. Matschek, R. Jordanowa, and R. Findeisen. Direct force feedback using Gaussian process based model predictive control. In *IEEE Conference on Control Technology and Applications (CCTA)*, pages 8–13. IEEE, 2020.

[127] P. Matyba, K. Maturova, M. Kemerink, N. D. Robinson, and L. Edman. The dynamic organic p-n junction. *Nature Materials*, 8(8):672–676, 2009.

[128] D. Mayne. Nonlinear model predictive control: Challenges and opportunities. In *Nonlinear Model Predictive Control*, volume 26 of *Progress in Systems and Control Theory*, pages 23–44. Birkhäuser Verlag, Boston, 2000.

[129] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[130] D. Q. Mayne, S. Raković, R. Findeisen, and F. Allgöwer. Robust output feedback model predictive control of constrained linear systems. *Automatica*, 42(7):1217–1222, 2006.

[131] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[132] C. D. McKinnon and A. P. Schoellig. Learning multimodal models for robot dynamics online with a mixture of Gaussian process experts. In *International Conference on Robotics and Automation (ICRA)*, pages 322–328. IEEE, 2017.

[133] W. H. Moase, C. Manzie, and M. J. Brear. Newton-like extremum-seeking part I: Theory. In *Conference on Decision and Control (CDC) held jointly with Chinese Control Conference*, pages 3839–3844. IEEE, 2009.

[134] F. Moresco. Driving molecular machines using the tip of a scanning tunneling microscope. In *Single Molecular Machines and Motors*, pages 165–186. Springer, 2015.

[135] R. Murray-Smith, D. Sbarbaro, C. E. Rasmussen, and A. Girard. Adaptive, cautious, predictive control with Gaussian process priors. *IFAC Proceedings Volumes*, 36(16):1155–1160, 2003.

[136] M. A. Musen, B. Middleton, and R. A. Greenes. Clinical decision-support systems. In *Biomedical Informatics*, pages 643–674. Springer, 2014.

[137] C. Musumeci, A. Liscio, V. Palermo, and P. Samorì. Electronic characterization of supramolecular materials at the nanoscale by conductive atomic force and Kelvin probe force microscopies. *Materials Today*, 17(10):504–517, 2014.

[138] R. M. Neal. Monte Carlo implementation of Gaussian process models for

Bayesian regression and classification. *arXiv preprint physics/9701026*, 1997.

[139] D. Nešić, A. Mohammadi, and C. Manzie. A framework for extremum seeking control of systems with parameter uncertainties. *IEEE Transactions on Automatic Control*, 58(2):435–448, 2012.

[140] D. Nešić, T. Nguyen, Y. Tan, and C. Manzie. A non-gradient approach to global extremum seeking: An adaptation of the Shubert algorithm. *Automatica*, 49(3):809–815, 2013.

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

[142] D. Nguyen-Tuong, M. Seeger, and J. Peters. Model learning with local Gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[143] H. Y. Noh and R. Rajagopal. Data-driven forecasting algorithms for building energy consumption. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems*, volume 8692, page 86920T. International Society for Optics and Photonics, 2013.

[144] L. Ortmann, D. Shi, E. Dassau, F. J. Doyle, S. Leonhardt, and B. J. 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.

[145] M. A. Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature*. PhD thesis, Oxford University, UK, 2010.

[146] C. J. Ostafew, A. P. Schoellig, and T. D. 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.

[147] C. J. Ostafew, A. P. Schoellig, T. D. 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.

[148] Y. Pan and E. Theodorou. Probabilistic differential dynamic programming. In *Advances in Neural Information Processing Systems*, pages 1907–1915, 2014.

[149] L. Y. Pao, J. Butterworth, and D. Y. Abramovitch. Combined feedforward/feedback control of atomic force microscopes. In *American Control Conference (ACC)*, pages 3509–3515. IEEE, 2007.

[150] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

[151] J. A. Paulson, S. Streif, R. Findeisen, R. D. Braatz, and A. Mesbah. Fast stochastic model predictive control of end-to-end continuous pharmaceutical manufacturing. In *Process Systems Engineering for Pharmaceutical Manufacturing*, vol-

ume 41 of *Computer Aided Chemical Engineering*, pages 353–378. Elsevier, 2018.

[152] F. Pérez-Cruz, S. Van Vaerenbergh, J. J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaria. Gaussian processes for nonlinear signal processing: An overview of recent advances. *IEEE Signal Processing Magazine*, 30(4):40–50, 2013.

[153] D. Petelin and J. Kocijan. Control system with evolving Gaussian process models. In *Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 178–184. IEEE, 2011.

[154] D. Piga, M. Forgione, S. Formentin, and A. Bemporad. Performance-oriented model learning for data-driven MPC design. *IEEE Control Systems Letters*, 3(3):577–582, 2019.

[155] G. Pillonetto and A. Chiuso. Gaussian processes for Wiener-Hammerstein system identification. *IFAC Proceedings Volumes*, 42(10):838–843, 2009.

[156] S. V. Raković and W. S. Levine. *Handbook of Model Predictive Control*. Springer, 2019.

[157] C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 751–758, 2004.

[158] C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. MIT press, 2006.

[159] J. B. Rawlings and B. R. Bakshi. Particle filtering and moving horizon estimation. *Computers & Chemical Engineering*, 30(10-12):1529–1541, 2006.

[160] J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, Madison, 2009.

[161] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1984), 2013.

[162] M. A. Rotea. Analysis of multivariable extremum seeking algorithms. In *American Control Conference (ACC)*, pages 433–437. IEEE, 2000.

[163] A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[164] A. Scheinker and M. Krstić. *Model-free Stabilization by Extremum Seeking*. Springer, 2017.

[165] G. Schitter, F. Allgöwer, and A. Stemmer. A new control strategy for high-speed atomic force microscopy. *Nanotechnology*, 15(1):108, 2004.

[166] G. Schitter, A. Stemmer, and F. Allgower. Robust 2DOF-control of a piezoelectric tube scanner for high speed atomic force microscopy. In *American Control Conference (ACC)*, pages 3720–3725. IEEE, 2003.

[167] P. O. Scokaert and D. Mayne. Min-max feedback model predictive control

for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8):1136–1142, 1998.

[168] P. Scokaert, D. Mayne, and J. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[169] D. E. Seborg, T. F. Edgar, and D. A. Mellichamp. *Process Dynamics and Control*. Wiley, 1989.

[170] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. *Artificial Intelligence and Statistics 9*, 2003.

[171] H. R. Sheikh and A. C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing*, 15(2):430–444, 2006.

[172] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[173] A. J. Smola and P. L. Bartlett. Sparse greedy Gaussian process regression. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, pages 619–625. MIT Press, 2001.

[174] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.

[175] E. Snelson, Z. Ghahramani, and C. E. Rasmussen. Warped Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 337–344. MIT Press, 2004.

[176] R. Soloperto, M. A. 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 538–543. IFAC, 2018.

[177] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

[178] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008.

[179] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[180] R. S. Sutton, A. G. Barto, and R. J. Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2):19–22, 1992.

[181] Y. Tan, W. H. Moase, C. Manzie, D. Nešić, and I. Mareels. Extremum seeking from 1922 to 2010. In *Chinese Control Conference*, pages 14–26. IEEE, 2010.

[182] Y. Tan, D. Nešić, and I. Mareels. On non-local stability properties of extremum

seeking control. *Automatica*, 42(6):889–903, 2006.

[183] C. Toher, R. Temirov, A. Greuling, F. Pump, M. Kaczmarski, G. Cuniberti, M. Rohlfing, and F. Tautz. Electrical transport through a mechanically gated molecular wire. *Physical Review B*, 83(15):155402, 2011.

[184] 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. IEEE, 2018.

[185] B. van Niekerk, A. C. Damianou, and B. Rosman. Online constrained model-based reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, Sydney, 2017.

[186] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría. Kernel recursive least-squares tracker for time-varying regression. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1313–1326, 2012.

[187] 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.

[188] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters. Stability of controllers for Gaussian process dynamics. *The Journal of Machine Learning Research*, 18(1):3483–3519, 2017.

[189] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, A. Romer, H. Schmidt, and J. Peters. Stability of controllers for Gaussian process forward models. In *International Conference on Machine Learning*, pages 545–554, 2016.

[190] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[191] R. von Mises and H. Geiringer. *Mathematical Theory of Probability and Statistics*. Elsevier Science, 1966.

[192] K. P. Wabersich and M. N. Zeilinger. Scalable synthesis of safety certificates from data with application to learning-based control. In *European Control Conference (ECC)*, pages 1691–1697. IEEE, 2018.

[193] C. Wagner, N. Fournier, F. Tautz, and R. Temirov. Measurement of the binding energies of the organic-metal perylene-teracarboxylic-dianhydride/Au (111) bonds by molecular manipulation using an atomic force microscope. *Physical Review Letters*, 109(7):076102, 2012.

[194] C. Wagner, N. Fournier, V. G. Ruiz, C. Li, K. Müllen, M. Rohlfing, A. Tkatchenko, R. Temirov, and F. S. Tautz. Non-additivity of molecule-surface van der Waals potentials from force measurements. *Nature Communications*,

5(1):1–8, 2014.

[195] C. Wagner, M. F. Green, P. Leinen, T. Deilmann, P. Krüger, M. Rohlfing, R. Temirov, and F. S. Tautz. Scanning quantum dot microscopy. *Physical Review Letters*, 115:026101, 2015.

[196] C. Wagner, M. F. Green, M. Maiworm, P. Leinen, T. Esat, N. Ferri, N. Friedrich, R. Findeisen, A. Tkatchenko, R. Temirov, and F. S. Tautz. Quantitative imaging of electric surface potentials with single-atom sensitivity. *Nature Materials*, 18(8):853–859, 2019.

[197] C. Wagner and F. S. Tautz. The theory of scanning quantum dot microscopy. *Journal of Physics: Condensed Matter*, 31(47):475901, 2019.

[198] 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. IEEE, 2015.

[199] 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.

[200] Y. Wang, F. Gao, and F. J. Doyle III. Survey on iterative learning control, repetitive control, and run-to-run control. *Journal of Process Control*, 19(10):1589–1600, 2009.

[201] Wikipedia contributors. Deepl translator — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=DeepL_Translator&oldid=983536594`, 2020. [Online; accessed 23-October-2020].

[202] C. K. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, pages 514–520. MIT press, 1996.

[203] X. Yang and J. M. Maciejowski. Fault tolerant control using Gaussian processes and model predictive control. *International Journal of Applied Mathematics and Computer Science*, 25(1):133–148, 2015.

[204] Y. Yong, S. R. Moheimani, B. J. Kenton, and K. Leang. Invited review article: High-speed flexure-guided nanopositioning: Mechanical design and control issues. *Review of Scientific Instruments*, 83(12):121101, 2012.

[205] H. Yu and U. Ozguner. Extremum-seeking control strategy for ABS system with time delay. In *American Control Conference (ACC)*, pages 3753–3758. IEEE, 2002.

[206] C. Zhang and R. Ordóñez. Numerical optimization-based extremum seeking control with application to ABS design. *IEEE Transactions on Automatic Control*, 52(3):454–467, 2007.

[207] C. Zhang and R. Ordóñez. *Extremum-Seeking Control and Applications: A Nu-*

merical Optimization-Based Approach. Advances in Industrial Control. Springer-Verlag London, 2012.

[208] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *International Conference on Robotics and Automation (ICRA)*, pages 528–535. IEEE, 2016.

[209] P. Zometa, M. Kogel, T. Faulwasser, and R. Findeisen. Implementation aspects of model predictive control for embedded systems. In *American Control Conference (ACC)*, pages 1205–1210. IEEE, 2012.