Diss. ETH No. 24098

Nonparametric Disturbance Correction and Nonlinear Dual Control

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich

(Dr. sc. ETH Zurich)

presented by

Edgar Dietrich Klenske

Dipl.-Ing., University of Stuttgart
born 1986-08-13
citizen of Germany

accepted on the recommendation of

Prof. Dr. Melanie N. Zeilinger, examiner
Dr. Philipp Hennig, co-examiner
Prof. Dr. Carl E. Rasmussen, co-examiner

2017

Nonparametric Disturbance Correction & Nonlinear Dual Control

Edgar Dietrich Klenske

2017

Nonparametric Disturbance Correction and Nonlinear Dual Control
© 2017 Edgar D. Klenske

to the stars

Abstract

Automatic control is an important aspect of modern technology, and many devices we use on a daily basis are using automatic control for actuation and decision-making. However, many advanced automatic control methods need a model of the system to control—a mathematical representation of the system's behavior. These models are not always easy to come by because of the underlying complexity of the system or the required measurement precision. Therefore, often a big portion of time is used for identification and tuning of these models.

Machine learning methods with the available regression and inference frameworks offer a new potential in the combination with model-based control methods to speed up, or even entirely automate, the process of model-creation, identification and tuning. This potential similarly extends to disturbance prediction: Methods from time-series forecasting can be used to infer a model of the environmental disturbances, which then can be used in predictive control methods.

The first concept covered in this thesis is the identification of quasiperiodic models for disturbance forecasting. Quasiperiodic disturbances are encountered in many applications that are affected by the ubiquitous day-night-cycle, revolving mechanical parts or recurring motions from biological objects. Being able to forecast disturbances, such as the outside air temperature, recurring gear errors or the beating motion of a heart, can help to increase control performance of the affected systems, especially in combination with model predictive control. In this thesis a quasiperiodic Gaussian process regression framework is used to learn and to predict periodic disturbances. A subsequent reference tracking model predictive controller then uses these predictions to control the system to a higher precision. The benefits of using this method are not only shown with simulated experiments, but also on hardware, on a telescope tracking setup in the laboratory. Since the development of this method was driven by a real problem in astronomical imaging, it is described how the method was implemented as a software solution. The advantage of the disturbance prediction method is shown on telescope tracking experiments in the field. The use of automatically identified quasiperiodic Gaussian process models makes it possible to use disturbance forecasting on a variety of systems not necessarily known at modeling time.

The second concept presented in this thesis is nonlinear dual control. While methods for simultaneous identification and control were published already in the 1960s, most approaches are either too complex to be used in practice or too simple to retain all critical features of the original dual control framework: *caution*, *exploration* and *selectiveness*. The dual control framework in this thesis is based on one approximation to the—theoretically ideal, but fundamentally intractable—optimal dual control problem. So far being used for linear systems only, this framework is extended to nonlinear systems. This is done by employing regression methods from machine learning; namely parametric regression, Gaussian process regression and neural network regression. Furthermore, the framework, which was originally only suitable for systems with quadratic cost, is extended to a general cost setting. This makes it possible to apply dual control to problems of economic cost. An exemplary application to a nonlinear building control problem shows the potential that dual control offers for real world applications. Overall, the presented extensions make it possible to use approximation-based dual control in the context of nonlinear regression models and flexible cost structures.

Zusammenfassung

Regelungstechnik ist ein wichtiger Bestandteil der modernen Technik, und viele Geräte, die wir täglich nutzen, arbeiten nur mit Hilfe von Reglern zuverlässig. Viele moderne Regler benötigen dynamische Modelle—mathematische Beschreibungen des Systemverhaltens—um zu funktionieren. Leider ist es aufgrund der Komplexität und der erforderlichen Messgenauigkeit nicht immer einfach, diese Modelle zu erstellen. Deshalb ist die Systemidentifikation und das Anpassen der Modelle an die realen Begebenheiten oftmals mit einem hohen Zeiteinsatz verbunden.

Methoden des maschinellen Lernens bieten die Möglichkeit, den Prozess der Modellierung und Anpassung zu beschleunigen oder gänzlich zu automatisieren. Dieses Potential gilt auch für Methoden zur Vorhersage von Störungen: Algorithmen zur Zeitreihenvorhersage können genutzt werden, um den zukünftigen Verlauf externer Störungen vorherzusagen. Diese Vorhersagen können anschließend in der modellprädiktiven Regelung genutzt werden, um die Regelgenauigkeit zu verbessern.

Das erste Konzept, welches in dieser Arbeit behandelt wird, ist die Identifikation von quasiperiodischen Modellen für die Vorhersage von Störungen. Quasiperiodische Störungen findet man in vielen Anwendungen, insbesondere bei solchen, die vom Tag- und Nacht-Zyklus, rotierenden mechanischen Teilen oder anderen periodischen Bewegungen abhängen. Die Möglichkeit, Störungen wie die Außentemperatur, Getriebefehler oder das Schlagen eines Herzens vorherzusagen, kann die Regelgenauigkeit der betroffenen Systeme erheblich verbessern. In dieser Arbeit wird ein quasiperiodischer Gauß-Prozess eingesetzt, um periodische Störungen zu modellieren und vorherzusagen. Ein Referenzfolgesystem auf Basis modellprädiktiver Regelung nutzt diese Vorhersagen, um die Nachführgenauigkeit des Reglers zu verbessern. Die Vorteile dieser Methode werden nicht nur mit simulierten Experimenten, sondern auch auf einem mechanischen Versuchsträger, einem Teleskop-Aufbau im Labor, gezeigt. Für die Verbesserung der Regelung von Teleskopen in der Astrophotographie wurde der Algorithmus in einer Softwarelösung implementiert und der Nutzen der Störungsvorhersage mit Experimenten im Feldversuch demonstriert. Mit dieser Softwarelösung ist es nun möglich, die Störungsvorhersage auf einer Vielzahl von Systemen zu nutzen, die zur Zeit der Modellierung nicht notwendigerweise bekannt sein müssen.

Das zweite Konzept in dieser Arbeit ist die nichtlineare duale Regelung. Während Methoden für die zeitgleiche Identifikation und Regelung dynamischer Systeme schon in den 1960er-Jahren publiziert wurden, sind die meisten Ansätze entweder zu komplex um in der Praxis eingesetzt zu werden, oder zu einfach um alle wichtigen Eigenschaften der dualen Regelung zu erhalten: *Vorsicht*, *Exploration* und *Selektivität*. Der Ansatz in dieser Arbeit basiert auf einer Approximation des—theoretisch idealen, aber praktisch nicht lösbaren—optimalen dualen Regelungsproblems. Diese bisher nur für lineare Systeme eingesetzte Methode wird auf nichtlineare Modelle erweitert. Dies wird durch die Nutzung verschiedener nichtlinearer Regressionsmethoden erreicht: parametrische Regression, Gauß-Prozess-Regression und Regression auf Basis neuronaler Netze. Weiter wurde der Ansatz, der bisher nur für Systeme mit quadratischen Kosten nutzbar war, auf allgemeine Kostenstrukturen erweitert. Dies ermöglicht es, duale Regelung auch auf Probleme mit ökonomischen Kosten anzuwenden. Eine beispielhafte Anwendung auf nichtlineare Gebäuderegelung zeigt das Potential, welches duale Regelung für praktische Anwendungen bietet. Insgesamt machen es die vorgestellten Erweiterungen möglich, approximative duale Regelung im Kontext nichtlinearer Regressionsmodelle und flexiblen Kostenstrukturen einzusetzen.

The highest forms of understanding we can achieve
are laughter and human compassion.
— Richard P. Feynman

Acknowledgements

Even though there is only one author listed on the cover, I never was alone. I was fortunate to receive constant support by many wonderful people during my PhD adventures.

First of all, I want to thank all the past and present colleagues at the Max Planck Institute for Intelligent Systems for the good times, for inspiring discussions over coffee, lunch and foosball, and for the thought-provoking caketalks and teatalks. I am grateful for having been surrounded by so many smart people, from which I could easily learn something new every day.

Thanks to Sabrina, Andrea, Diana, Karin and Sebastian for running the department so smoothly, and to Markus Schneller for tracking down old papers and tech reports from the pre-digital age.

My work was partly supported by the Max Planck ETH Center for Learning Systems. In addition to the financial support, I am grateful for the many opportunities to travel to Zürich and elsewhere to connect with the members and fellows of the CLS.

There is a big difference between a piece of research code running in the lab, and proper software working robustly on many devices. I want to thank Raffi Enficiaud and the team of the Software Workshop at the Max Planck Institute in Tübingen for their help in making it happen. Thanks to the developers of PHD2 Guiding, especially Andy Galasso and Bruce Waddington, for the valuable input and their help with including my work into the PHD2 Guiding project.

I want to thank Bernhard Schölkopf for giving me an amazing scientific home for more than four years, for his expertise on telescope guiding, and for the star gazing sessions in Tübingen, in the Black Forest and on La Palma. It was an amazingly quiet and productive, but at the same time exciting and inspiring working environment.

I owe my deepest gratitude to my supervisors Philipp Hennig and Melanie Zeilinger, who invested a lot in my PhD. To Philipp, for the long hours in front of the blackboard, for the PhD guidance and the constant support, and for the invaluable advice, not limited to scientific matters. And to Melanie, for teaching me so many things about control, and for always keeping a calm mind despite the administrative obstacles. It was an honor working with you, and I hope we can continue working together in the future.

Finally, I would like to thank Verena for her selfless support during the times where being a PhD student was hard and exhausting, and for her love over all these years.

Edgar Klenske
Leinfelden, March 2017

it is not important to accumulate knowledge
it is important to share it

Contents

List of Figures

List of Tables

List of Algorithms

Prologue

Introduction

FEEDBACK CONTROL is one of the key enabling technologies of our time. It is hidden in many devices and appliances without most users even noticing. The use of feedback controllers dates back to the ancient Greece, where they were used to control water-based clocks. [98, §II.1] In the modern world, one of the first uses of feedback controllers were centrifugal governors [95] (Figure 1) to control steam engines—sparking the industrial revolution. Since then, feedback controllers have found their way into our lives and we are using them every day.

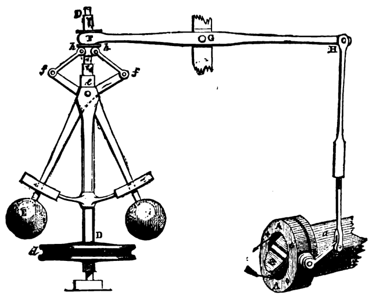


Figure 1: Centrifugal governor with steam valve. Image from [120].

[120] Routledge, *Discoveries and Inventions of the Nineteenth Century*, 1900

[98] Mayr, *The Origins of Feedback Control*, 1970

[95] Maxwell, "On Governors," 1867

A typical example is electronic stability control (ESC) in modern cars, where the steering angle is constantly measured as a proxy to the driver's steering intention. Also the turn rate of the car is measured with a gyroscope. In the case of fading tire grip, the measured turn rate deviates from the driver's intention. This discrepancy is picked up by the controller and then fed back as control inputs in the form of brake signals for the individual wheels. This way the car stays safely on the road, while without feedback control it would have departed from the road.

While there exist simple model-free feedback methods to compensate deviations from the desired value, many advanced methods require a mathematical model of the controlled system. The model predicts the state evolution of the system and can be used to either synthesize the feedback law or to calculate the feedback signal in each time step. Model-based control usually requires less tuning and can have advanced features like lookahead for reference tracking. Since we want to make use of such features, the controllers in this thesis are all model-based.

The classic way of doing model identification is physical modeling, e.g., Newton's laws can be used to deduce the mathematical equations of motion of mechanical systems. If physical modeling is not possible, either due to the lack of knowledge about the system, or because precise enough measurements can not be made, *system identification* [86] is often used to find an approximation to the dynamics.

[86] Ljung, *System Identification: Theory for the User*, 1999

For some applications, building a controller a priori is not viable. The systems in question might not be known beforehand or can be subject to changes in the dynamics. In these cases, offline system identification can lead to bad system performance or may not be able to achieve the desired controller requirements. *Adaptive control* [76]

[76] Kumar and Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*, 1986

offers the possibility to create controllers that are able to adapt to unknown plants, changing plant dynamics, or unknown environments. Adaptivity means that the control system needs a way of learning about the relevant dynamics of the plant or the environment.

Disturbance Forecasting

For many applications, controllers are built for feedback-based disturbance rejection, which means that the effect of outside disturbances is kept small by using the measured tracking error. However, sometimes feedback-based disturbance rejection is not fast enough due to high performance demand or slow measurement processes. In such cases it can help to incorporate a prediction of the disturbance to eliminate parts of the error introduced by the disturbance in advance.

General time-series forecasting is a challenging topic, but it gets more manageable if the disturbance in question is of periodic nature—for example when the origin of the disturbance is tied to the day-night-cycle or periodic motions. In such cases, the use of periodic models to predict the disturbances can considerably increase predictive performance and, thus, make the use of disturbance forecasting much more available.

Active Identification

Most adaptive control methods only learn in a passive manner. The available information is used, but it is not attempted to invest control energy for exploration or identification. In episodic settings, where the control system executes the same task multiple times, this is often sufficient. However, for non-episodic problems—the control of a single trial—active exploration is an important aspect. When a system never runs twice under the same conditions, it is crucial not only to use the available information in the best possible way, but also to actuate the system such that the relevant information is generated. This simultaneous identification and control is the realm of *dual control* [148].

[148] Wittenmark, "Adaptive Dual Control Methods: An Overview," 1995

The key to differentiate between these methods is the time at which the information is acquired. In system identification, the data is collected upfront, before the controller is used on the plant. This also means that the controller is not updated online. An adaptive controller is learning "on-the-fly", while the system is running. It collects data and updates the internal model regularly. A dual control system goes one step further and uses reasoning about the learning process itself to find the optimal actions. This means that a dual controller incorporates the predicted future information acquisition into the decision process.

Outline

Part I introduces the relevant mathematical background. Gaussian process regression (Chapter 1) is used to model system dynamics and external error sources. Model predictive control (Chapter 2) is a well-suited control framework for incorporating predictions into the control system. We also give an overview on the field of adaptive control (Chapter 3).

Part II shows how quasiperiodic Gaussian process models can be used to enhance control performance. The general concept of quasiperiodic disturbance forecasting in combination with reference tracking model predictive control is introduced (Chapter 4) and subsequently applied to an experimental problem in astronomical imaging: the periodic error correction in telescope guiding (Chapter 5). Based on this work, we developed software to use the periodic error correction feature in an open source telescope guiding system (Chapter 6).

Part III is concerned with the concept and application of dual control to nonlinear systems. We first give a general overview of the dual control literature and a classic algorithm (Chapter 7) which is then extended to modern nonlinear regression techniques (Chapter 8). We further extend this nonlinear framework to constrained systems with linear cost structure and show a potential application to building control (Chapter 9).

Chapter 10 gives a concise summary of the work presented in this thesis and concludes with an outlook on future research directions.

Publications

Parts of the work in this thesis were done in collaboration with colleagues and are based on the following publications:

Chapters 4 and 5 are based on the following journal publication:

> E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. "Gaussian Process-Based Predictive Control for Periodic Error Correction." In: *IEEE Transactions on Control Systems Technology* 24.1 (2016), pp. 110–121. DOI: `10.1109/TCST.2015.2420629`

A conference version was published in:

> E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. "Nonparametric Dynamics Estimation for Time Periodic Systems." In: *Annual Allerton Conference on Communication, Control, and Computing*. 2013, pp. 486–493. DOI: `10.1109/Allerton.2013.6736564`

Chapters 7 and 8, as well as Section 1.6.2, are based on the following journal publication:

> E. D. Klenske and P. Hennig. "Dual Control for Approximate Bayesian Reinforcement Learning." In: *Journal of Machine Learning Research* 17.127 (2016). Ed. by M. Opper, pp. 1–30. URL: `http://jmlr.org/papers/v17/15-162.html`

Chapter 9 is based on the following conference publication:

> E. D. Klenske, P. Hennig, B. Schölkopf, and M. N. Zeilinger. "Approximate Dual Control Maintaining the Value of Information with an Application to Building Control." In: *European Control Conference (ECC)*. 2016, pp. 800–806. DOI: `10.1109/ECC.2016.7810387`

Part I

Preliminaries

1 Gaussian Process Regression

Regression is the task of learning a functional relationship between input and output variables from potentially noisy observations. Regression problems are an important part of learning methods in control theory and applications. For example, regression can be used to infer state transition functions of dynamical systems from noisy measurements of the states. These learned dynamics can then be used to synthesize controllers or to predict the state evolution.

Classically in system modeling and automatic control, parametric models are used to describe the equations of motion derived from first principles[1]. The parameters in such models are the physical properties that can be measured: mass, length, etc. If possible, this is the ideal case, but often it is difficult to describe all relevant effects a priori. Since many systems are complex, it is challenging to come up with perfect parametric models for them and, thus, more flexible models that can infer unforeseen functional relationships can offer substantial benefits.

[1] For example, in mechanical systems the first principles are Newton's equations of motion.

When using flexible models, the problem of overfitting quickly arises, where even non-effects like noise are fitted, leading to poor generalization. One way to mitigate overfitting is to use a probabilistic prior on the function space. Adding a prior effectively regularizes the regression problem and brings about a global solution. A Gaussian prior on the function space is called a *Gaussian process* (GP).

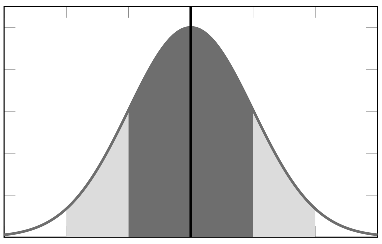


Figure 2: One-dimensional Gaussian distribution (—). The mean (—) and the bands of one (▆) and two standard deviations (▭) are highlighted.

A Gaussian Distribution over Functions

The well-known one-dimensional Gaussian (or *normal*) distribution is shown in Figure 2. The Gaussian distribution is defined by two parameters: the mean, which defines the average value, and the variance, which defines the breadth of the distribution.

The Gaussian distribution easily extends to the multivariate case, where we now have a mean *vector* and a covariance *matrix* defining the distribution. The role of the mean remains the same, but since we now have multiple variables, we not only have to define the breadth of the distribution, but also the coupling between variables, the correlation. Breadth and shape of the distribution are defined by the covariance matrix, where the off-diagonal terms define the coupling between variables. Figure 3 shows a two-dimensional Gaussian distribution.

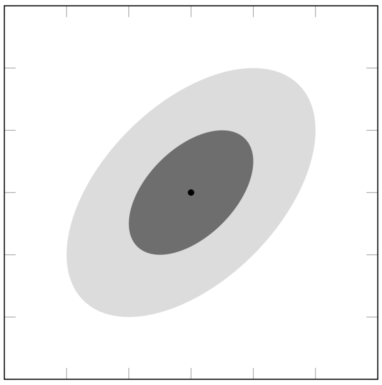


Figure 3: Two-dimensional Gaussian distribution. The mean (•) and the area of one (▆) and two standard deviations (▭) are shown.

Analogously to the extension to multiple dimensions, we can extend this notion to entire functions. Instead of a mean vector we now have a mean *function* and instead of a covariance matrix we have a covariance *function* that defines the correlation between function values at different inputs. The extension of the Gaussian distribution to functions is called Gaussian process, an example is shown in Figure 4.

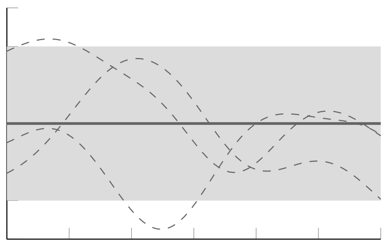


Figure 4: Gaussian process prior. Shown are the mean (——), two standard deviations (▨) and three samples (– – –).

1.1 Model and Notation

A regression task is to infer a function $f(x)$ from measurements y at locations x. Usually the measurements are corrupted by Gaussian noise:

$$y = f(x) + \gamma, \qquad \gamma \sim \mathcal{N}(0, \sigma^2), \tag{1.1}$$

where $\mathcal{N}$ is a Gaussian (normal) distribution.

For notational simplicity, this chapter only covers the scalar case of Gaussian processes. In the case of a vector-valued function f, one GP is trained for every dimension. For our purposes, a Gaussian process $\mathcal{GP}(f; m, k)$ is an infinite-dimensional probability distribution over the space of real-valued *functions* $f : \mathbb{R} \rightarrow \mathbb{R}$, such that every finite, N-dimensional linear restriction to function *values* $f(\mathbf{x}) \in \mathbb{R}^N$ (measurements) at locations $\mathbf{x} \in \mathbb{R}^N$ (measurement locations) is an N-variate Gaussian distribution $\mathcal{N}(f(\mathbf{x}); m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}))$. It is parametrized by a mean function $m(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^N$, and a covariance function $k(\mathbf{x}, \mathbf{x}) : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^{N \times N}$. The mean has a relatively straightforward role; it simply shifts predictions. The covariance function's responsibility is more intricate. It can be interpreted as a similarity measure over function values, expressed in terms of the inputs, and controls the shape of the Gaussian process belief in the space of functions. It has to be chosen such that, for any $\mathbf{x} \in \mathbb{R}^N$, the matrix $k(\mathbf{x}, \mathbf{x}) \in \mathbb{R}^{N \times N}$, also known as the *kernel* matrix, is positive semidefinite.

As common in the Gaussian process literature, without loss of generality, we assume the mean function m to be zero to simplify notation. Note that this does not imply that only zero-mean Gaussian processes should be considered for practical applications. Rather, the choice of mean function should be part of the modeling considerations.

1.2 Inference in Gaussian Processes

The main inference machinery in the GP regression framework is Bayes' rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}, \qquad p(\mathcal{M}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{M}) \times p(\mathcal{M})}{p(\mathcal{D})}$$

where p is a probability density function, $\mathcal{D}$ stands for the data and $\mathcal{M}$ for the model. When the prior is a Gaussian process and the likelihood is Gaussian, the posterior is again a Gaussian process. The mean and covariance function of the posterior can be calculated in closed form with linear algebra calculations.

Even if the Gaussian process $\mathcal{GP}(f; m, k)$ is an infinite-dimensional object with mean function m and covariance function k, we can reason about any finite amount of data points $\mathbf{x} = [x_1, \ldots, x_N]$ and prediction point x by evaluating the mean and covariance function at those locations only. This amounts to the application of the marginalization rule of Gaussian algebra [21, §2.3] and results in a multivariate Gaussian distribution. Stacking the predictive value y and the vector of noise-free function evaluations $\mathbf{y} = [y_1; \ldots; y_N]$ into one vector results in the following joint distribution

[21] Bishop, *Pattern Recognition and Machine Learning*, 2006

$$\begin{bmatrix} y \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{xx} & k_{x\mathbf{x}} \\ k_{\mathbf{x}x} & k_{\mathbf{x}\mathbf{x}} \end{bmatrix} \right), \tag{1.2}$$

where we have introduced the shorthand notation $k_{\cdot\cdot} = k(\cdot, \cdot)$.

Since we have measured $\mathbf{y}$, but not y, we apply the conditioning rule [21, §2.3] of Gaussian algebra to obtain

$$y \sim \mathcal{N}\left(k_{x\mathbf{x}} K^{-1} \mathbf{y}, k_{xx} - k_{x\mathbf{x}} K^{-1} k_{\mathbf{x}x} \right), \tag{1.3}$$

where $K = k(\mathbf{x}, \mathbf{x})$. The matrix K is called the Gram matrix (or *kernel* matrix). We can write the predictive mean and predictive covariance explicitly as

$$m^{|\mathbf{x},\mathbf{y}}(x) = k_{x\mathbf{x}} K^{-1} \mathbf{y} \tag{1.4a}$$
$$k^{|\mathbf{x}}(x, x') = k_{xx'} - k_{x\mathbf{x}} K^{-1} k_{\mathbf{x}x'}, \tag{1.4b}$$

which can easily be implemented.[2] Figure 5 shows an example of a GP posterior with noise-free observations. After conditioning on the measurements, the function space is restricted to functions that pass through them.

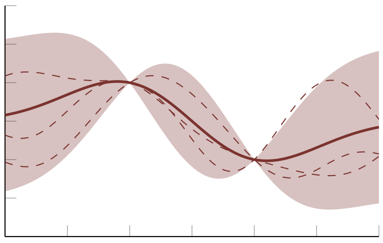


Figure 5: Gaussian process posterior after two noise-free observations. Shown are the mean (——), two standard deviations (▆) and three samples (- - -).

[2] Note that, while the formulation (1.4) is mathematically concise, the Cholesky decomposition [115] of the Gram matrix is usually used to carry out the calculations for increased speed and numerical stability.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

So far, we only considered noise-free observations. Adding independent and identically distributed Gaussian observation noise to the GP framework is done simply by adding measurement noise to the Gram matrix K:

$$K_{\text{noisy}} = k(\mathbf{x}, \mathbf{x}) + \sigma^2 I \tag{1.5}$$

The inference under noisy measurements results in a similar posterior, with the difference that the posterior mean is closer to the prior, and the pointwise posterior distribution is not as narrow as in the noise-free case. Figure 6 shows an example of GP inference under noise.

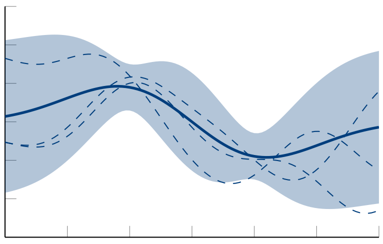


Figure 6: Gaussian process posterior after two noisy observations. Shown are the mean (——), two standard deviations (▆) and three samples (- - -).

1.3 Sampling from Gaussian Processes

Gaussian processes are generative models. This means that it is possible to draw samples from the prior or posterior distribution, similar to drawing from a Gaussian distribution. This is of two-fold importance:

First, methods that use the GP model might need samples for numerical marginalization, when analytic integration is intractable. Instead of calculating an expectation directly, a sum over samples can serve as approximation

$$\mathbb{E}_f\left[c(f)\right] = \int c(f)\mathcal{GP}(f;m,k)df \approx \frac{1}{S}\sum_{i=1}^{S} c(s_i), \tag{1.6}$$

where $c(f)$ is a functional operator and s_i are the samples from the GP.

Secondly, samples are a great tool for analyzing the function space defined by the GP. It is important to note that the mean of a GP looks fundamentally different from samples from the same process, see Figures 4 and 5. This is due to the smoothing property of the mean [115, § 2.6]. For choosing the covariance function and its parameters[3] it is, thus, helpful to compare samples of the selected GP with real data to see if they look similar. This can help in the analysis of how the chosen model fits the data.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

[3] See Sections 1.4 and 1.5.

For finite-dimensional datasets, where the samples s are vectors of function *evaluations* at locations $\mathbf{x}$, the sampling process is relatively straightforward, since it is equivalent to sampling from a multivariate Gaussian distribution with the covariance of the GP:

$$s = L\rho, \tag{1.7}$$

where L is a matrix satisfying $LL^\intercal = k(\mathbf{x},\mathbf{x})$ and ρ is a random Gaussian vector of appropriate size. k can be any suitable positive semidefinite covariance function, especially also the posterior covariance of a GP.

1.4 Choosing a Covariance Function

The covariance function defines how samples and predictions of the Gaussian process look like, by shaping the underlying probability distribution in the function space. So far, we considered the covariance function as given. But where does it come from, and how should it be chosen?

There is an alternate way of deriving Gaussian processes, usually called the "weight-space view" [115, § 2.1]. We will not reproduce the entire derivation here, but instead point out certain aspects.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

Consider general linear regression

$$f(x) = w^{\mathsf{T}}\Phi(x), \tag{1.8}$$

where w are weights and $\Phi(x)$ is a vector of potentially nonlinear feature functions. The shape of the functions that can be represented by this model is defined by the shape of the chosen feature functions.

The notion of general linear regression can be expressed in the GP framework and can be extended to the nonparametric case with infinitely many features.[4] We then obtain a covariance function (or *kernel*) that is defined by the feature functions that we have chosen in the first place. This means that, by the choice of covariance function, we can choose the shape of the functions that can be represented by the Gaussian process.

[4] This is usually done by reformulating, and application of the famous "kernel trick" [125, §2.2].

[125] Schölkopf and Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, 2002

Theoretically, every positive semidefinite kernel can be used as covariance function for a Gaussian process. However, for this thesis, we only consider stationary covariance functions, for which the covariance depends on the distance $r = |x - x'|$ of the inputs and not on the location itself.

1.4.1 Output Correlation

The covariance function of a Gaussian process defines a mapping from the distance r in input space to correlation in output space. For radial basis functions, for example, this means that the function values of two points that are close in input space are correlated to a higher degree than the function values of points that are far away in input space. Figure 7 visualizes this.

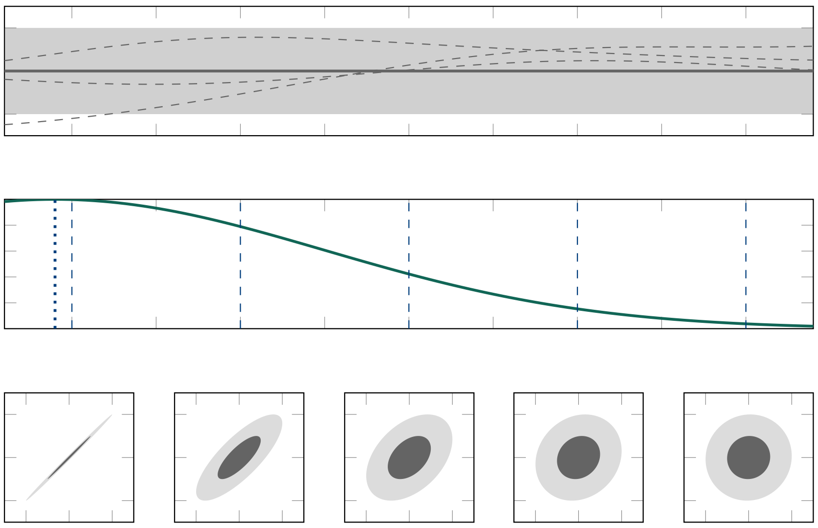


Figure 7: Correlation of different points in output space for a radial basis function. **Top:** GP with mean (——), two standard deviations (▨) and three samples (– – –). **Middle:** Correlation function (——) for the reference location (······) and evaluation locations (– – –). **Bottom:** Two-dimensional Gaussian distributions between the reference location and the different evaluation locations, shown as areas for 1 and 2 standard deviations (■/▨).

This correlation between function values is responsible for the overall shape of samples of the GP. If points that are relatively close in input space have a high correlation, this means that there is not much variability in the sampled functions, and the inputs have to move further away to allow for significant changes in the function values. If the correlation between nearby points is low, this allows for high variability within shorter distance: The functions are more flexible.

1.4.2 Examples of Covariance Functions

There are many different covariance functions to choose from. In this section we only provide those which are relevant for this thesis.

Square Exponential Covariance Function

One important covariance function is the square exponential[5]

$$k_{\text{SE}}(x, x'; \theta, \ell) = \theta^2 \exp\left(-\frac{|x - x'|^2}{2\ell^2}\right), \tag{1.9}$$

where θ^2 is the signal variance and ℓ the length scale parameter. Using this covariance function is equivalent to infinite-dimensional Gaussian feature regression. It is differentiable and integrable and therefore compatible to many use-cases. The shape of the square exponential covariance function is shown in Figure 8.

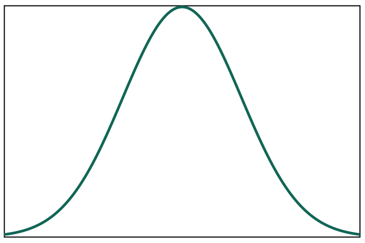


Figure 8: Square exponential covariance function.

[5] Also known as "squared exponential", "exponentiated quadratic", "radial basis function" (RBF) or "Gaussian" covariance function.

Periodic Covariance Function

Much more specific is the periodic covariance function [92]

$$k_{\text{P}}(x, x'; \theta, \ell, \lambda) = \theta^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi}{\lambda}(x - x')\right)}{\ell^2}\right), \tag{1.10}$$

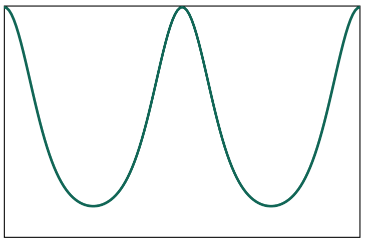


Figure 9: Periodic covariance function.

[92] MacKay, "Introduction to Gaussian Processes," 1998

where θ^2 and ℓ are similar parameters as above, and λ is the period length. This periodic covariance is essentially a square exponential covariance where the inputs are warped through a sine. A GP with this covariance function can only represent periodic functions with a specified period length λ. This might seem restrictive, but it can be valuable because it is much more data efficient than other covariances and has better extrapolation performance because stronger assumptions on the underlying function space are made. The shape of the periodic covariance function is shown in Figure 9.

1.4.3 Combined Covariance Functions

In practice, the functions we want to learn are often mixtures of different signals and it is hard to find a covariance function that suits all of them simultaneously. Instead of trying to fit all components with a single covariance function, which usually leads to poor data efficiency and/or predictive performance, the Gaussian process framework is flexible enough to allow for combinations of covariance functions.

In practice, combined covariance functions are obtained by element-wise addition or multiplication of the kernel matrices:

$$k_\mathrm{c}(\mathbf{x},\mathbf{x}) = k_1(\mathbf{x},\mathbf{x}) + k_2(\mathbf{x},\mathbf{x}) \tag{1.11}$$

or

$$k_\mathrm{c}(\mathbf{x},\mathbf{x}) = k_1(\mathbf{x},\mathbf{x}) \odot k_2(\mathbf{x},\mathbf{x}), \tag{1.12}$$

where k_c stands for the kernel combination and k_1, k_2 are the individual kernels. The meaning of the additive kernel is relatively simple: It adds two different functions on top of each other. [115, §4.2] One example for a GP with additive covariance structure is shown in Figure 10.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

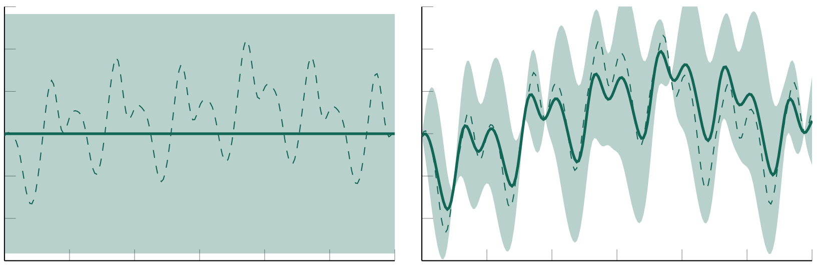


Figure 10: Gaussian process prior (left) and posterior (right) for an additive covariance. The covariance function is combined from a square exponential and a periodic covariance. Shown are the mean (——), two standard deviations (▨) and one sample (- - -) each.

The effect of multiplying kernels is more intricate. While it can be shown that element-wise multiplications still retain the positive-definiteness of the resulting matrices [126], it is harder to grasp what this means for the function. In a way, the multiplication acts similarly to a logical and, so that there is a high correlation between points that have a high correlation under *both* covariance functions.

[126] Schur, "Bemerkungen zur Theorie der Beschränkten Bilinearformen mit Unendlich Vielen Veränderlichen," 1911

Output Projections

When inference is done with a covariance combination, it is also possible to split the prediction to the different parts for additional interpretation possibilities or for the subsequent use in other algorithms. This is done with the Gaussian algebra of Equations (1.2) and (1.3) by conditioning the prediction on the kernels that we are interested in.

If, for example, the prediction should be done only for kernel 1, the resulting predictive posterior amounts to [40]

[40] Duvenaud, "Automatic Model Construction with Gaussian Processes," 2014

$$m_1^{|\mathbf{x},\mathbf{y}}(x) = k_1(x,\mathbf{x})K_c^{-1}\mathbf{y} \tag{1.13a}$$

$$k_1^{|\mathbf{x}}(x,x') = k_1(x,x') - k_1(x,\mathbf{x})K_c^{-1}k_1(\mathbf{x},x'). \tag{1.13b}$$

where K_c is the Gram matrix for a combined kernel and k_1 is the kernel function for only one of the kernels. The effect of this conditioning of the posterior is shown in Figure 11, using the same kernel combination as in Figure 10.

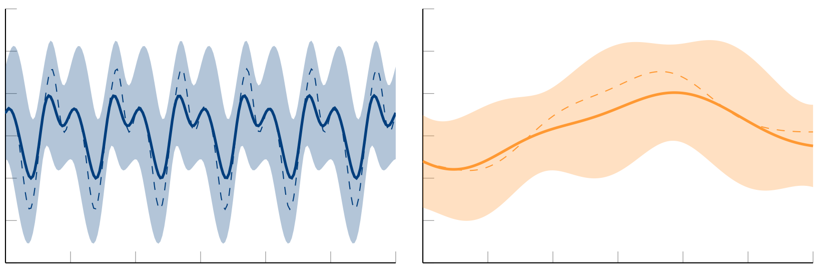


Figure 11: Comparing different output projections from a combined covariance. The Gaussian conditioning shows the periodic component (left) and the square exponential component (right) respectively. Shown are the mean (—/—), two standard deviations (■/■) and one sample (- - -/- - -) each.

1.5 Setting the Kernel Parameters

Most covariance functions have parameters, like the signal variance θ^2 or the length scale ℓ in Equation (1.9). These parameters are usually called *hyper*parameters. The reason for this name is the idea of a Gaussian process being an infinite-dimensional linear regression model: In linear regression, the weights for the different features are called parameters, and the Gaussian process has infinitely many of these parameters.[6] To distinguish the parameters of the covariance function from the linear regression parameters, they are called *hyperparameters*.

[6] These methods are also referred to as *non*parametric in the literature, to stress the fact that there is no *finite* amount of features to select.

1.5.1 The Role of the Hyperparameters

Even though covariance functions like the square exponential are *universal kernels* that can theoretically learn any function [101], it is important to set the hyperparameters correctly. Otherwise the learning can be inefficient and might need much more data than a GP with a suitable set of hyperparameters. [144]

[101] Micchelli, Xu, and Zhang, "Universal Kernels," 2006

[144] van der Vaart and van Zanten, "Information Rates of Nonparametric Gaussian Process Methods," 2011

Output Variance

The scale factor (θ^2 in Equation (1.9)) of the covariance function defines the output variance of the GP, i. e. the range over which functions typically vary in value. The effect of this parameter is visualized in Figure 12.

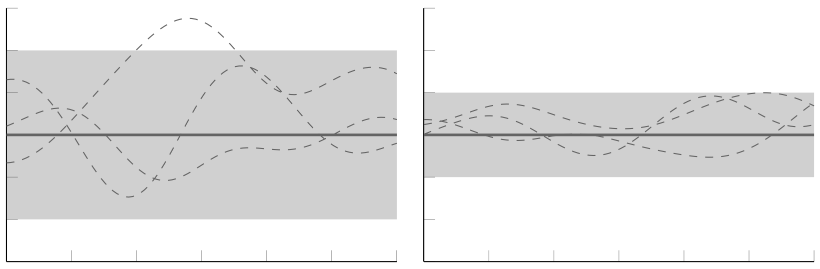


Figure 12: Gaussian process with high signal variance (left) compared to one with low signal variance (right), using a square exponential kernel. Shown are the mean (——), two standard deviations (▨) and three samples (- - -) each. The higher signal variance allows for larger function values.

Length Scale

Most covariance functions have a length scale parameter (ℓ in Equation (1.9)). Often, changing the length scale parameter is equivalent to a scaling of the input distance. This parameter defines how much the function values can vary relative to the input distance. The effect of this parameter is visualized in Figure 13.

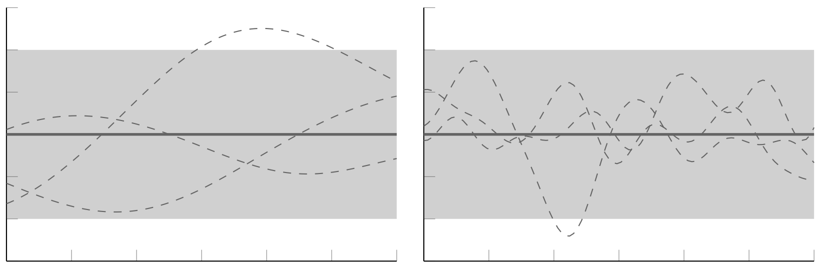


Figure 13: Gaussian process with long length scale (left) compared to one with short length scale (right), using a square exponential kernel. Shown are the mean (——), two standard deviations (▨) and three samples (- - -) each. The shorter length scale allows for more variation within the same distance in input space.

Period Length

The periodic covariance has a period length parameter (λ in Equation (1.10)). This parameter defines the distance in input space after which the function repeats itself. The effect of this parameter is visualized in Figure 14.

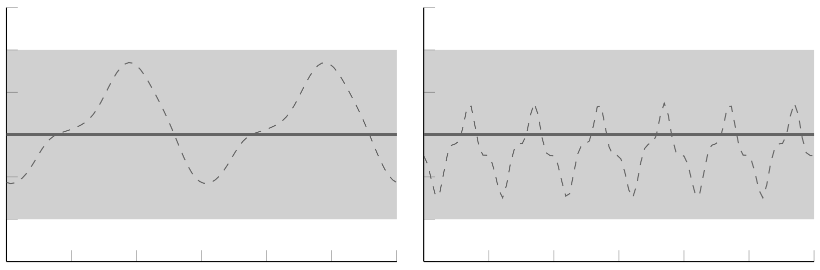


Figure 14: Gaussian process with long period length (left) compared to one with short period length (right), using a periodic kernel. Shown are the mean (——), two standard deviations (▨) and one sample (- - -) each. The sampled functions are perfectly periodic with the chosen period length.

1.5.2 The Hyperparameter Likelihood

For many applications it can be enough to choose the hyperparameters from physical reasoning, but this is not always the case. Especially for automatic parameter tuning it is important to assess how good the parameters fit the data.

Assume that all kernel parameters are subsumed in the parameter vector $\boldsymbol{\eta}$. Inferring good values for $\boldsymbol{\eta}$ is important for good modeling performance. The fundamental framework for GP inference is provided by Bayes' theorem. The likelihood for observations $\mathbf{y}$ at locations $\mathbf{x}$, conditioned on the parameters $\boldsymbol{\eta}$, can be found by marginalization over the unknown function f, which is feasible because both $p(\mathbf{y}|f)$ and $p(f|\boldsymbol{\eta})$ are Gaussian distributions:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x},\boldsymbol{\eta}) &= \int p(\mathbf{y}|f)p(f|\boldsymbol{\eta})df \\ &= \int \mathcal{N}(\mathbf{y}; f(\mathbf{x}), \sigma^2 I)\mathcal{GP}(f; 0, k(\boldsymbol{\eta}))df \\ &= \mathcal{N}(\mathbf{y}; 0, K(\boldsymbol{\eta})). \end{aligned} \tag{1.14}$$

These calculations are easier to perform in log domain, where the logarithm of the marginal likelihood is given by

$$\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\eta}) = -\frac{1}{2}\mathbf{y}^\mathsf{T} K(\boldsymbol{\eta})^{-1}\mathbf{y} - \frac{1}{2}\log|K(\boldsymbol{\eta})| - \frac{N}{2}\log 2\pi. \tag{1.15}$$

With this likelihood, it is easy to compare the model fit for different sets of hyperparameters.

1.5.3 Hyperparameter Optimization

One way of setting the hyperparameters is to maximize the marginal likelihood (1.15),

$$\boldsymbol{\eta}^* = \arg\max_{\boldsymbol{\eta}} p(\mathbf{y}|\mathbf{x},\boldsymbol{\eta}). \tag{1.16}$$

This is usually done with a gradient-based optimizer; often quasi-Newton methods, such as the BFGS algorithm [103, §6.1], are used. Since the optimization leads to the maximum likelihood (ML) solution for the hyperparameters, it is usually called *type-II maximum likelihood*[7], to distinguish it from the GP inference itself. [115, §5.4.1] In a way, hyperparameter optimization can be seen as a second layer of inference on top of the Gaussian process.

[103] Nocedal and Wright, *Numerical Optimization*, 2006

[7] The maximum likelihood approach is also known as *evidence maximization* in the literature.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

Using the ML estimate is one of the most widely studied and best understood strategies in statistics. [147, §9.3 – §9.6] It is not without weaknesses, e. g., the optimization is prone to get stuck in local minima. Some of these weaknesses are often resolved if enough data is

[147] Wasserman, *All of Statistics: A Concise Course in Statistical Inference*, 2010

available, or by the use of customized optimization algorithms. Other approaches, for example, integrating the hyperparameters over ML estimates or cross validation, have been examined in the past and found to perform worse than the above type-II maximum likelihood approach in practice, see, e. g., [91].

[91] MacKay, "Comparison of Approximate Methods for Handling Hyperparameters," 1999

1.5.4 Priors on the Hyperparameters

In order to make the method more robust, it can be beneficial to introduce priors on the parameters. For the strictly positive parameters $\boldsymbol{\eta}$, gamma priors [13, §8.3] are a classic choice:

[13] Barber, *Bayesian Reasoning and Machine Learning*, 2011

$$p(\boldsymbol{\eta}|\boldsymbol{\kappa},\boldsymbol{\tau}) = \prod_i \frac{\boldsymbol{\eta}_i^{\kappa_i-1}\exp(-\frac{\boldsymbol{\eta}_i}{\tau_i})}{\Gamma(\kappa_i)\tau_i^{\kappa_i}}, \tag{1.17}$$

where κ_i and τ_i are tuning-parameters[8], and Γ is the gamma function.

[8] Sometimes also referred to as hyper-hyperparameters.

Again, the maximization is easier to perform in log domain, in which the effect of the prior is additive, leading to the following optimization problem:

$$\begin{aligned}\boldsymbol{\eta}^* &= \arg\max_{\boldsymbol{\eta}} p(\boldsymbol{\eta}|\mathbf{y},\mathbf{x}) = \arg\max_{\boldsymbol{\eta}} p(\mathbf{y}|\mathbf{x},\boldsymbol{\eta})p(\boldsymbol{\eta}) \\ &= \arg\max_{\boldsymbol{\eta}} \left(\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\eta}) + \log p(\boldsymbol{\eta})\right). \end{aligned} \tag{1.18}$$

In (1.18), the prior effectively turns into a regularizer, simplifying optimization and avoiding degeneracy. The additional computational cost is negligible compared to the matrix inversion needed for (1.15).

1.6 Numerical Effort and Approximations

Gaussian processes are generally considered to be a relatively expensive method. This is due to the matrix inversion and determinant calculation in Equation (1.3) which both have (naïve[9]) asymptotic complexity of $\mathcal{O}(N^3)$ in the number of samples N. For large amounts of training data, this can be prohibitive.

[9] Sometimes, lower complexity numbers are reported for using the Strassen algorithm [136] and further improvements, but they are rarely implemented in practice.

[136] Strassen, "Gaussian Elimination is Not Optimal," 1969

In the light of using GPs for applications in automatic control, there is another point to consider: It is rarely, if ever, acceptable to have growing inference cost over time. Control algorithms should run reliably fast and therefore have almost constant runtime for both the inference and the control part.

There are many different ways of dealing with the numerical complexity of Gaussian processes, see Chapter 8 of the textbook by Rasmussen and Williams [115] for an overview. In the following, we review two methods that will be used in subsequent chapters of this work.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

1.6.1 Subset-of-Data Approximation

One of the simplest and most efficient approximation methods is the Subset-of-Data (SD[10]) method. The idea is to reduce the number N of available data points by considering only a smaller subset of M data points with $M \ll N$. Since M can be chosen in advance, the runtime of the algorithm is known and remains constant.

[10] In the literature also abbreviated as "SoD".

Changing the set of data points does not change the inference algorithm at all; therefore, this method can be implemented quickly and efficiently. Of course, the considered data points need to be chosen at runtime. Ideally the selection should be done according to how informative a data point is, but this optimization can be demanding, too. Hence, usually approximative methods are used. For example, depending on the expected distribution of data points in the dataset, the used data can be randomly sampled or selected by optimization of some criterion, e. g., a differential entropy score [81].

[81] Lawrence, Seeger, and Herbrich, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine," 2003

1.6.2 Sparse Spectrum Approximation

By Mercer's theorem [73, §3.a], the kernel can be decomposed into a converging series over eigenfunctions $\phi(x)$, as

[73] König, *Eigenvalue Distribution of Compact Operators*, 1986

$$k(x, x') = \sum_{l=1}^{\infty} \lambda_l \phi_l(x) \phi_l^*(x'), \tag{1.19}$$

where ϕ_l are functions that are orthonormal relative to some measure μ (the precise choice of which is irrelevant for the time being), with the property

$$\int k(x, x') \phi_l(x') d\mu(x') = \lambda_l \phi_l(x). \tag{1.20}$$

In this sense, Gaussian process regression can be seen as "infinite-dimensional" Bayesian linear regression, where the infinite inner product (1.19) is tractable because of the kernel trick [125, §2.2].

[125] Schölkopf and Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, 2002

Using the kernel formulation comes at the cost of a growing Gram matrix and, thus, rising inference cost, and is rarely acceptable for practical control applications. Therefore, it is often necessary to project the GP belief onto a finite representation, replacing the infinite sum in Equation (1.19) with a finite inner product of a low-dimensional explicit feature map $\Phi(x)$

$$k(x, x') \approx \Phi(x)^\intercal \Lambda \Phi(x'), \tag{1.21}$$

where Λ is a diagonal eigenvalue matrix. This bounds the computational cost of the inference (1.4), because the more efficient formulae of general linear regression can be used

$$m^{|\mathbf{x},\mathbf{y}}(x) = \frac{1}{\sigma^2}\Phi(x)^\intercal A^{-1}\Phi(\mathbf{x})\mathbf{y} \tag{1.22a}$$

$$k^{|\mathbf{x}}(x,x') = \Phi(x)^\intercal A^{-1}\Phi(x), \tag{1.22b}$$

where $A = \sigma^{-2}\Phi(\mathbf{x})\Phi(\mathbf{x})^\intercal + \Lambda^{-1}$.

We define the feature map Φ that projects the inputs onto a predefined finite basis of functions, drawn from the eigenspectrum of the kernel with respect to the Lebesgue measure. Similar approaches have been recently proposed in the literature [82], [113]. The following provides a short, self-contained introduction:

[82] Lázaro-Gredilla et al., "Sparse Spectrum Gaussian Process Regression," 2010

[113] Rahimi and Recht, "Random Features for Large-Scale Kernel Machines," 2008

By Bochner's theorem [133, §2.5], the covariance function $k(r)$ (with $r = |x - x'|$) of a stationary mean-square continuous random process can be represented as the Fourier transform of a positive finite measure and, if that measure has a density $S(s)$, as the Fourier dual of S:

[133] Stein, *Interpolation of Spatial Data: Some Theory for Kriging*, 1999

$$k(r) = \int_{-\infty}^{\infty} S(s)e^{2\pi\imath sr}ds, \tag{1.23}$$

where $\imath$ is the imaginary unit. This means that the eigenfunctions of the kernel are trigonometric functions, and stationary covariance functions, like the commonly used square exponential kernel (1.9), can be approximated by cosine basis functions as

$$k(x,x') \approx \tilde{k}(x,x') = \frac{\theta^2}{F}\sum_{i=1}^{F}\cos(\omega_i|x-x'|), \tag{1.24}$$

where F is the number of features, and the frequencies ω_i of the feature functions can be sampled from the power spectrum of the process.[11] An example of such kernel approximation is shown in Figure 15. With increasing number of features, the approximation can be chosen as close to the true covariance function as needed, while keeping the number of features in a range that is still feasible within the time constraints of the control algorithm.

[11] For example, the Latin hypercube sampling technique [99] can be used.

[99] McKay, Beckman, and Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," 1979

1.7 Extensions

The Gaussian process framework is powerful and has many useful features and extensions. Two further concepts are important for this thesis and will thus be presented here.

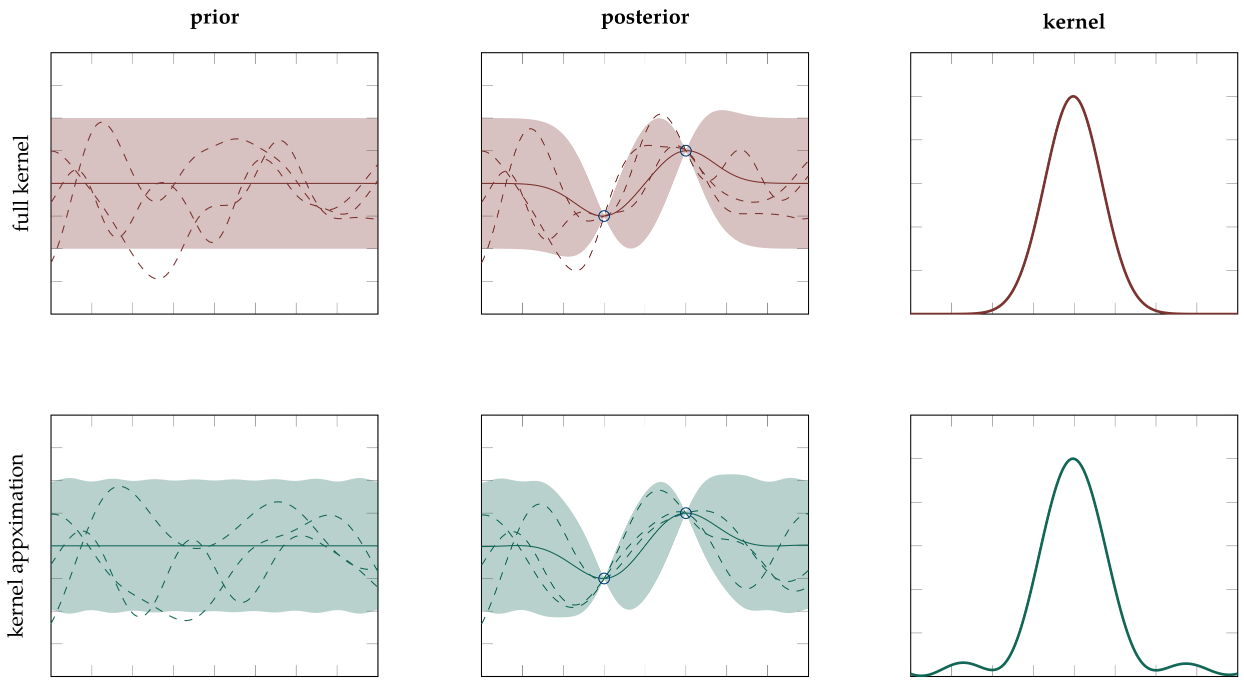


Figure 15: Comparison of a finite kernel approximation to the full kernel. Prior (left), posterior (middle) and kernel function (right) of both the full kernel function (top row) and the approximate kernel (bottom row). Shown are the mean (—/—), two standard deviations (🟥/🟩) and three samples each (- - -/- - -)

1.7.1 Heteroscedastic Noise

Usually noise is assumed uniform for all measurements (homoscedastic), but this may not be satisfied in practice. Not all measurement processes have constant noise level, therefore it can be useful to consider heteroscedastic noise instead, allowing for variable noise variance.

The additive noise matrix in Equation (1.5) can also be given in the form of a diagonal matrix

$$K_{\text{noisy}} = k(\mathbf{x}, \mathbf{x}) + \begin{pmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_N^2 \end{pmatrix}, \tag{1.25}$$

where σ_i^2 for $i = \{1 \dots N\}$ is the noise variance for the i-th measurement. This way a sensor or measurement method with variable noise level can be modeled accurately within the Gaussian process framework.

1.7.2 Explicit Feature Functions

Since Gaussian processes are related to general linear regression, it is possible to combine both methods. This is useful if certain parts of a regression problem can be modeled as parametric features and other parts can not. Even though a GP can potentially learn everything, it is much more efficient to model as much as possible in the form of parametric features and train a more general GP for the remainder only.

An additive combination of general linear regression and Gaussian process can be modeled as

$$g(x) = \beta^\top \psi(x) + f(x), \tag{1.26}$$

where $\beta \sim \mathcal{N}(b, B)$ is a parameter vector, ψ is a set of parametric feature functions and $f(x) \sim \mathcal{GP}(0, k_f)$.

A numerically stable way of formulating the predictive mean and covariance is [115, §2.7]

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

$$m_g^{|\mathbf{x},\mathbf{y}}(x) = m_f^{|\mathbf{x},\mathbf{y}}(x) + \bar{\beta}^\top R(x) \tag{1.27a}$$

$$k_g^{|\mathbf{x}}(x, x') = k_f^{|\mathbf{x}}(x, x') + R(x)^\top \left(B^{-1} + \Psi K^{-1} \Psi^\top\right)^{-1} R(x'), \tag{1.27b}$$

where the feature matrix Ψ collects the feature vectors $\psi(\mathbf{x})$ for all data points, $\bar{\beta} = (B^{-1} + \Psi K^{-1} \Psi^\top)^{-1}(\Psi K^{-1}\mathbf{y} + B^{-1} b)$, and $R(x) = \psi(x) - \Psi K^{-1} k_f(\mathbf{x}, x)$. These calculations represent the inference in the joint model, combining general linear regression with a Gaussian process. If the prior on the parametric part of the model should be uninformative, one can obtain the limit case by letting $B^{-1} \to 0$, which is possible in the formulation of Equation (1.27).

2 Discrete-Time Optimal Control

Model-based optimal control is successfully used in many control systems and poses a prerequisite for this thesis. Since the topic is rather broad, we only cover the topics relevant for this thesis.

We briefly introduce linear time-invariant (LTI) systems and the concepts of stability and controllability (Section 2.1). For LTI systems, we describe different types of model-based optimal controllers: Dynamic programming based optimal control (Section 2.3) and model predictive control by constrained optimization (Section 2.4). We introduce the concept of receding horizon control (Section 2.5) and describe some important extensions of discrete-time optimal control (Section 2.6).

2.1 Linear Time-Invariant Systems

Linear time-invariant (LTI) systems are often used in model-based control due to their advantageous computational properties. Even though most systems are not linear, often a linear approximation to a nonlinear system offers good performance in the vicinity of the operating point. [105, §2.7]

[105] Ogata, *Modern Control Engineering*, 2010

For this chapter, we use the common discrete-time LTI system

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k), \quad k \in \mathbb{N}, \tag{2.1}$$

where $t_k \in \mathbb{R}$ denotes the k-th time instant relative to the sampling time Δt, $x \in \mathbb{R}^{n_x}$ is the state, $A \in \mathbb{R}^{n_x \times n_x}$ is the discrete-time state-transition matrix and $B \in \mathbb{R}^{n_x \times n_u}$ is the discrete-time input matrix.

Starting from a continuous model

$$\dot{x}(t) = A_c x(t) + B_c u(t), \tag{2.2}$$

with A_c and B_c of appropriate size, the discrete-time model can be obtained by exact discretization [46, §3.2]. Throughout this thesis, we consider discrete-time systems, if not stated otherwise.

[46] Friedland, *Control System Design: An Introduction to State-space Methods*, 2005

2.1.1 Stability and Controllability

Informally, *stability* for linear systems can be viewed as the property of keeping the system in a bounded domain. For system (2.1) and $u(t) \equiv 0$, the state evolution starting from state $x(0)$ amounts to $A^k x(0)$ after k time steps. This state evolution only is bounded for $k \rightarrow \infty$ if the absolute values of all eigenvalues λ of the dynamics matrix A are smaller than 1.

Definition 2.1 ([104, §4.3])
A discrete-time autonomous linear system is stable, if the eigenvalues λ of the state transition matrix A lie inside of the unit circle.

[104] Ogata, *Discrete-time Control Systems*, 1995

While the stability property describes whether the system states experience infinite growth, *controllability* describes whether the system can be moved from any bounded starting point to any bounded target point in the state space with a finite number of control actions.

Definition 2.2 ([104, §6.2])
A discrete-time linear system is controllable, if the controllability matrix $\mathcal{C} = [B, AB, A^2B, \dots, A^{T-1}B]$ has full rank.

[104] Ogata, *Discrete-time Control Systems*, 1995

2.2 Optimal Control

While the definition of optimal control in the literature is intricate, this thesis focuses on a class of optimal controllers that are optimization-based. Depending on a chosen cost function, the goal of optimal control is to find the control inputs that minimize a cost criterion, e.g.,

$$J(x(t_k), \mathbf{u}) = l_T(x_T) + \sum_{i=0}^{T-1} l_i(x_i, u_i), \qquad x_0 = x(t_k), \tag{2.3}$$

where l_i is the stage cost, l_T is the terminal cost, $\mathbf{u} = [u_0, \dots, u_{T-1}]$ is the input trajectory, and $i = 0 \dots T$ is the time index along the horizon. The corresponding states $x_1, \dots, x_T$ are defined according to the discrete-time dynamics $x_{i+1} = Ax_i + Bu_i$, given the initial state x_0.

Note the difference between the planning states in subset notation x_i and the physical states in parenthesis notation $x(t_k)$, which is depicted in Figure 16. The first planning state x_0 is identical to the current physical state $x(t_k)$.

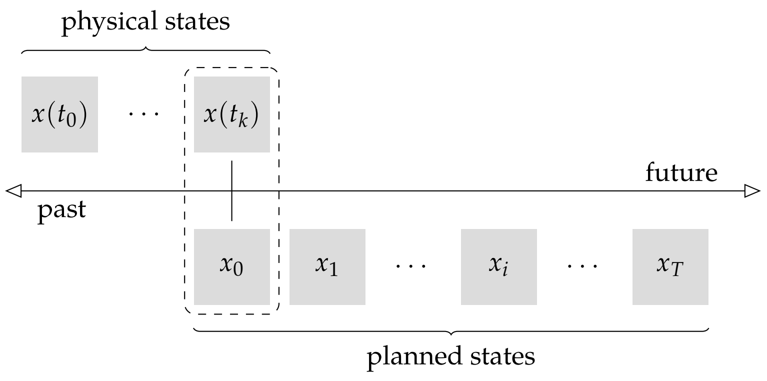


Figure 16: Relation between physical states (denoted by $x(t_k)$) and planning states of the algorithm (denoted by x_i).

Depending on the problem at hand, different cost functions can be chosen for l. Usually quadratic cost is used for states and inputs in tracking problems, it is defined as

$$l_i(x_i, u_i) = x_i^\mathsf{T} Q_i x_i + u_i^\mathsf{T} R_i u_i, \qquad l_T(x_T) = x_T^\mathsf{T} Q_T x_T, \tag{2.4}$$

where Q_i are symmetric positive semidefinite matrices and R_i are a symmetric positive definite matrices.

Linear cost is often used in systems of economic type, e. g., for evaluating energy cost or monetary value, it is defined as

$$l_i(x_i, u_i) = q_i^\mathsf{T} x_i + r_i^\mathsf{T} u_i, \qquad l_T(x_T) = q_T^\mathsf{T} x_T, \tag{2.5}$$

where q_i and r_i are cost scaling vectors.

An optimal controller finds the control input for the current time step by minimizing the cost (2.3) with respect to the control inputs $\mathbf{u}$

$$u^*(t_k) = u_0^* \qquad \mathbf{u}^* = \arg\min_{\mathbf{u}} J(x(t_k), \mathbf{u}), \tag{2.6}$$

where u_0^* is the first element of the optimal control trajectory $\mathbf{u}^*$; optimality is denoted by $\cdot^*$.

The optimization of the control inputs can be done in different ways. We highlight two commonly used approaches: dynamic programming (Section 2.3) and model predictive control (Section 2.4).

2.3 Dynamic Programming

The first optimization technique that we introduce for optimal control is dynamic programming[1] (DP), introduced by Richard Bellman [19]. It is considered an important milestone for optimization and automatic control because it enabled the use of optimal planning and optimal control for many systems where this was too hard a problem before. A comprehensive overview on the topic is given by Bertsekas [20].

[1] Note that, being coined earlier than the term "computer programming", the term *programming* usually stands for "optimization" in the mathematical context. Nowadays this can be a bit confusing. See [30] for a full explanation.

[30] Dantzig, "Linear Programming," 2002

[19] Bellman, *Dynamic Programming*, 1957

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

Dynamic programming speeds up optimization in dynamical systems by breaking down the overall optimization problem into smaller subproblems that are quickly solved. The drawback is that, since dynamic programming for continuous systems only works efficiently for problems with closed-form solution, it can not be straightforwardly used for systems with constraints. In such cases, the more general model predictive control approach (Section 2.4) can be used.

2.3.1 The Dynamic Programming Equation

The fundamental idea behind dynamic programming is the *principle of optimality*: Informally it means that, for any given state of the system, the optimal action only depends on the state and not on the way this state was reached. This is an obvious property of route planning problems (illustrated in Figure 17): The optimal route from any point on the way between start and destination does not depend on the locations visited before. Once a certain state is reached, only the way from there to the destination is relevant.

In the light of the principle of optimality, the optimal cost for the cost function (2.3) can be written in the form of nested minimizations

$$\begin{aligned} J^*(x(t_k)) &= \min_{\mathbf{u}} J(x(t_k), \mathbf{u}) \\ &= \min_{u_0} \left[l_0(x_0, u_0) + \min_{u_1} \left[l_1(x_1, u_1) + \dots \right] \right], \end{aligned} \tag{2.7}$$

where $x_0 = x(t_k)$, and $x_{i+1} = Ax_i + Bu_i$. This reformulation is possible because the inner part (the truncated subproblem starting from $i = 1$) does not depend on the optimization variable u_0 of the outer part. Replacing the optimal cost of the inner subproblem by $J_1^*(x_1)$, we obtain

$$J_0^*(x_0) = \min_{u_0} \left[l_0(x_0, u_0) + J_1^*(x_1) \right]. \tag{2.8}$$

More generally, we can write this as the recursive dynamic programming equation [20, §1.3]

$$J_i^*(x_i) = \min_{u_i} \left[l_i(x_i, u_i) + J_{i+1}^*(x_{i+1}) \right], \tag{2.9}$$

where J_i^* is the optimal accumulated cost from time step i to the end of the horizon, and l_i is the cost for the current state and input.

Using this formulation stresses the fact that the optimal cost-to-go J_{i+1}^* only depends on the next state and not on past states. This makes it possible to break down the optimization problem into small subproblems, every subproblem depending only on the current state.

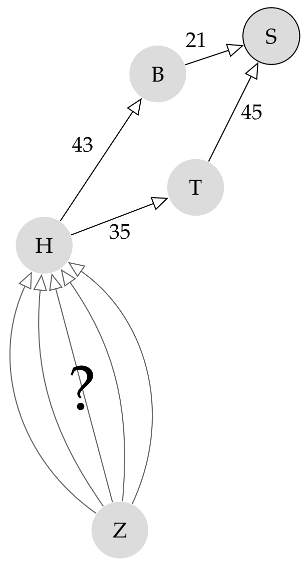


Figure 17: Simple routing problem. A student wants to go from Zurich (Z) to Stuttgart (S). No matter which of the many possible routes he takes to the intermediate town H, the rest of the problem is identical to the problem of finding the shortest path from H to S.

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

2.3.2 The Finite-Horizon Linear Quadratic Regulator

Most quadratic optimization problems in dynamic settings can be broken down into smaller subproblems with the dynamic programming equation. Consider the dynamical system (2.1) and quadratic cost over a horizon $0 \dots T$:

$$J(x_0, \mathbf{u}) = x_T^\mathsf{T} Q_T x_T + \sum_{i=0}^{T-1} \left(x_i^\mathsf{T} Q_i x_i + u_i^\mathsf{T} R u_i \right), \tag{2.10}$$

where the state weights Q_i are symmetric positive semidefinite matrices and the input weight R_i is a symmetric positive definite matrix. In order to obtain the overall minimizing input trajectory $\mathbf{u}^*$, a large optimization problem needs to be solved.

However, using the recursive formulation originating from the DP equation makes it possible to solve this optimization problem with low computational budget. This formulation implicitly exploits the sparse structure of the optimization problem due to the Markov characteristic of the dynamic system.

For the last time step, the cost simply is

$$J_T(x_T) = x_T^\intercal Q_T x_T = x_T^\intercal V_T x_T, \tag{2.11}$$

where V_T is the quadratic value matrix, initialized at the last time step. Using the dynamic system equation (2.1) in the DP equation (2.9) we can compute the value matrix V_i recursively, resulting in the discrete-time Riccati equation (DRE) [20, §4.1]

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

$$V_i = A^\intercal V_{i+1} A - A^\intercal V_{i+1} B \left(B^\intercal V_{i+1} B + R_i\right)^{-1} B^\intercal V_{i+1} A + Q_i. \tag{2.12}$$

The DRE not only defines the optimal cost-to-go depending on the current state

$$J_0^*(x(t_k)) = x_0^\intercal V_0 x_0, \qquad x_0 = x(t_k), \tag{2.13}$$

but also the optimal control law

$$u_0^*(x(t_k)) = -\left(B^\intercal V_1 B + R_0\right)^{-1} B^\intercal V_1 A x_0, \qquad x_0 = x(t_k), \tag{2.14}$$

which is called the finite-horizon linear quadratic regulator (f-LQR).

Note that this equation defines a control *policy* for each time step up to the horizon, which returns an optimal control *action* for the states $x_i = x(t_{k+i})$. This is important for systems under uncertainty: An optimal plan consisting of pre-computed control actions can fail when the states change unexpectedly only by a small amount. A policy-based optimal controller, on the other hand, can deal with state uncertainties by calculating the optimal action based on the current state in each time step.

2.3.3 The Infinite-Horizon Linear Quadratic Regulator

For continuously running systems, the optimal controller should optimize the cost up to an infinite horizon. In practice this is not possible because of the infinite number of recursion steps that would need to be carried out. Letting $T \to \infty$ and assuming constant cost $Q_i = Q \; \forall i$, $R_i = R \; \forall i$ in the discrete-time Riccati equation (2.12) results in the *algebraic Riccati equation*

$$V_\infty = A^\intercal V_\infty A - A^\intercal V_\infty B \left(B^\intercal V_\infty B + R\right)^{-1} B^\intercal V_\infty A + Q, \tag{2.15}$$

which has a steady-state solution if the pair (A, B) is controllable. [20]

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

The steady-state solution can be found, e. g., by iterating Equation (2.12) until convergence, or by the Schur method [79]. The static feedback law

[79] Laub, "A Schur Method for Solving Algebraic Riccati Equations," 1979

$$u_\infty^*(x(t_k)) = -\left(B^\intercal V_\infty B + R\right)^{-1} B^\intercal V_\infty A x(t_k) := L x(t_k) \tag{2.16}$$

is usually called infinite-horizon linear quadratic regulator (∞-LQR) and is an instance of optimal static feedback control for a steady state. The ∞-LQR is popular in practice because the state-feedback controller has negligible computational cost once the feedback-gain L is computed. Therefore, using the ∞-LQR is a powerful design method for state-feedback controllers if a model of the system is available and there are no constraints.

2.3.4 Reference Tracking with Lookahead

Controllers often have the goal of controlling the system to a setpoint or "desired state." Especially in linear systems, the optimal control problem can be transformed easily so that the goal state is the origin. In that case, applying standard ∞-LQR control can be enough to create a feedback controller.

For many applications, however, it is not enough to keep the state close to a setpoint. In robotics, for example, being able to track a desired reference trajectory $\mathbf{x}^{\text{ref}}$ is important for the execution of a planned motion. If the reference trajectory is known in advance, the dynamic programming equations can be implemented accordingly. Carrying out the finite-horizon dynamic programming calculations including a future reference trajectory entails the necessity of using a quadratic ansatz that also includes linear terms in the value function:

$$J_i^*(x_i) = x_i^\intercal V_i x_i + v_i^\intercal x_i + \text{const}\,. \tag{2.17}$$

For reference tracking dynamic programming, the recursive equations for calculating V_i and v_i are (see Appendix A.1):

$$V_i = A^\intercal \left(V_{i+1} - V_{i+1}B\left(B^\intercal V_{i+1} B + R_i\right)^{-1} B^\intercal V_{i+1}\right) A + Q_i \qquad V_T = Q_T \tag{2.18a}$$

$$v_i = A^\intercal \left(v_{i+1} - V_{i+1}B\left(B^\intercal V_{i+1} B + R_i\right)^{-1} B^\intercal v_{i+1}\right) - Q_i x_i^{\text{ref}} \qquad v_T = -Q_T x_T^{\text{ref}}, \tag{2.18b}$$

and the optimal control is defined by

$$u_0^*(x(t_x)) = u_0^*(x_0) = -\left(B^\intercal V_1 B + R_0\right)^{-1}\left[B^\intercal V_1 A x_0 + v_1\right]. \tag{2.19}$$

The important advantage of finite-horizon reference tracking lies in the "lookahead" (or "preview") property. A static state-feedback controller, such as the ∞-LQR, can only ever act based on the current deviation from the desired setpoint. This means that every change in reference can only be taken into account after it occurs. A controller with access to the future trajectory, on the other hand, can already act in a feed-forward fashion, which often results in better system behavior. An example of this is shown in Figure 18, where the pure feedback controller "waits" much longer and acts more abruptly than the dynamic programming based optimal controller, which benefits from the lookahead property.

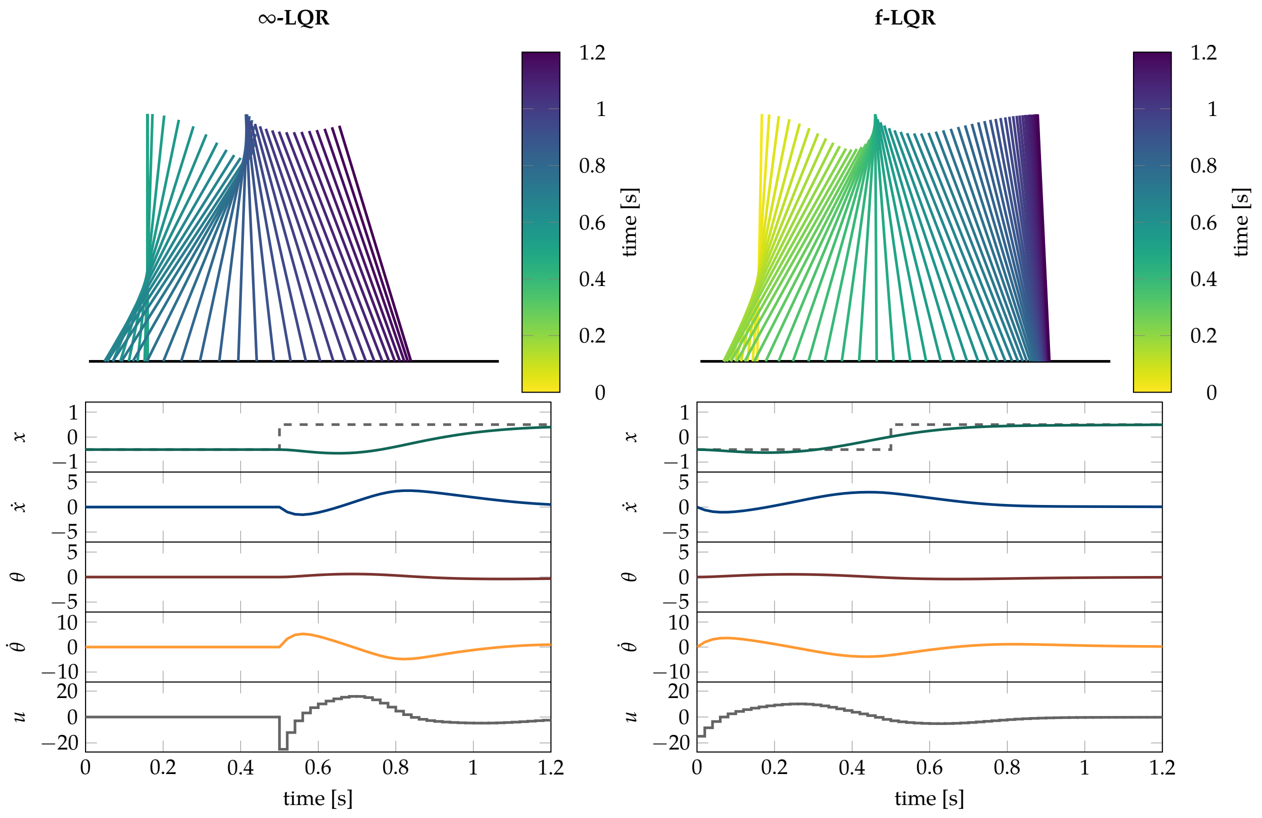


Figure 18: Comparison between the control behavior of ∞-LQR (left) and f-LQR (right) for an inverted pendulum on a cart. The reference position jumps at 0.5 s from -0.5 to 0.5. The pendulum is actuated, with its base moving along the rail. The pendulum is plotted for each sampling time, where the color indicates the time. The states and input are plotted underneath, where x is the horizontal position, θ the angle and u the input.

2.3.5 Stochastic Systems

When considering stochastic control systems of the form

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k) + \xi(t_k) \qquad \xi(t_k) \sim \mathcal{N}(0, D) \qquad (2.20a)$$

$$y(t_k) = Cx(t_k) + \gamma(t_k) \qquad \gamma(t_k) \sim \mathcal{N}(0, W), \qquad (2.20b)$$

the state evolution is not deterministic any more, and since states are not fully measurable, there is uncertainty about the true states $x(t_k)$. The optimal solution in the linear case is to maintain a Gaussian belief over the states by Kalman filtering [64, 122].

[64] Kálmán, "A New Approach to Linear Filtering and Prediction Problems," 1960

[122] Särkkä, *Bayesian Filtering and Smoothing*, 2013

In order to account for the future uncertainty along the control horizon, we denote the Kalman filtered state covariance as $\Sigma_{i+1|i}$ after prediction from time step i and as $\Sigma_{i+1|i+1}$ after updating with the measurement y_{i+1} made at time step $i+1$. Of course, the future measurement is not yet available, but the predictive covariance is deterministic and can therefore be incorporated in the calculations below.

For the uncertain system, the cost function is defined in expectation:

$$J_T(x_T) = \mathop{\mathbb{E}}_{x_T}\left\{x_T^\mathsf{T} Q_T x_T\right\} \tag{2.21}$$

$$J_i(x_i, u_i) = \mathop{\mathbb{E}}_{x_i}\left\{x_i^\mathsf{T} Q_i x_i + u_i^\mathsf{T} R_i u_i + J_{i+1}^*(x_{i+1})\right\}. \tag{2.22}$$

After some linear algebra manipulations (Appendix A.2), we obtain the same recursion for the value matrix (2.12), and the same control policy (2.14). Nonetheless, the cost function is different in this case:

$$J_i^*(x_i) = \mathop{\mathbb{E}}_{x_i}\left\{x_i^\mathsf{T} V_i x_i\right\} + \operatorname{tr}\left\{\sum_{j=i}^{T-1}\left[\Sigma_{j+1|j} - \Sigma_{j+1|j+1}\right] V_{j+1}\right\} + \operatorname{tr}\left\{\sum_{j=i}^{T} Q_j \Sigma_{j|j}\right\}. \tag{2.23}$$

The additional terms arising from uncertainty are usually not considered, since they do not influence the control policy. [20, §5.2]

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

2.4 Model Predictive Control

Constraints are ubiquitous. Finite actuator power and other physical limitations are encountered in many practical systems. Model predictive control (MPC) is a framework that can explicitly incorporate such constraints. MPC is similar to dynamic programming in the way the model is used for predictions; the fundamental difference is the optimization procedure. While in dynamic programming the Bellman principle is used to construct an optimal policy for each time step recursively, in MPC the entire optimization problem is solved at once, which makes it much easier to incorporate constraints. However, this comes at a price, since dealing with the full optimization problem and the need for constrained optimization methods makes the overall procedure computationally demanding. Therefore, MPC originally was used only in slow control systems, e. g., in the processing industry. Nowadays, MPC has found widespread use, partly due to the increasing computational power available in control systems.

Model predictive control constitutes a large field of research. Comprehensive overviews can be found, e. g., in [90], [119], [117].

[90] Maciejowski, *Predictive Control with Constraints*, 2002

[119] Rossiter, *Model-based Predictive Control: A Practical Approach*, 2003

[117] Rawlings and Mayne, *Model Predictive Control: Theory and Design*, 2009

2.4.1 Constrained Finite-Time Optimal Control

In addition to the system definition (2.1), we consider constraints on states and inputs resulting in the constrained LTI system

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k) \tag{2.24a}$$
$$u(t_k) \in \mathbb{U} \quad \forall k, \qquad x(t_k) \in \mathbb{X} \quad \forall k, \tag{2.24b}$$

where $\mathbb{U}$ and $\mathbb{X}$ are polytopic sets defining the input and state constraints. Taking these constraints into account, dynamic programming has no closed form solution, but the more general MPC methods can be used.

The standard case of a model predictive control problem is to find the optimal state and input trajectory for the constrained dynamical system

$$(\mathbf{x}^*, \mathbf{u}^*) = \arg\min_{\mathbf{x},\mathbf{u}} \quad l_T(x_T) + \sum_{i=0}^{T-1} l_i(x_i, u_i) \tag{2.25a}$$
$$\text{s.t.} \quad x_0 = x(t_k) \tag{2.25b}$$
$$x_{i+1} = Ax_i + Bu_i \qquad i = 0 \dots T-1 \tag{2.25c}$$
$$u_i \in \mathbb{U} \qquad i = 0 \dots T-1 \tag{2.25d}$$
$$x_i \in \mathbb{X} \qquad i = 1 \dots T \tag{2.25e}$$

where $\mathbf{x} := [x_1, \dots, x_T]$ is the state trajectory and $\mathbf{u} := [u_0, \dots, u_{T-1}]$ are the controls. After the optimization problem (2.25) is solved, the optimal control action $u^*(t_k) = u_0^*$ can be extracted from the optimal control trajectory $\mathbf{u}^*$.

An example of the different behaviors of unconstrained (f-LQR) and constrained (MPC) optimization under input saturation is shown in Figure 19. Not taking the input constraints into account leads to a failing experiment for the f-LQR controller.

2.4.2 Optimization

In order to solve the MPC optimization problem with readily available optimization software (e. g., [52], [61]), it is necessary to reformulate the MPC problem into standard forms. For example, most quadratic programming (QP) solvers use the problem formulation

[52] Grant and Boyd, *CVX: Matlab Software for Disciplined Convex Programming, Version 2.1*, 2014

[61] IBM, *ILOG CPLEX Optimizer*, 2016

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \quad \frac{1}{2}\mathbf{z}^\mathsf{T} H \mathbf{z} + h^\mathsf{T}\mathbf{z} \tag{2.26a}$$
$$\text{s.t.} \quad A_{\text{eq}}\mathbf{z} = b_{\text{eq}} \tag{2.26b}$$
$$A_{\text{in}}\mathbf{z} \leq b_{\text{in}}, \tag{2.26c}$$

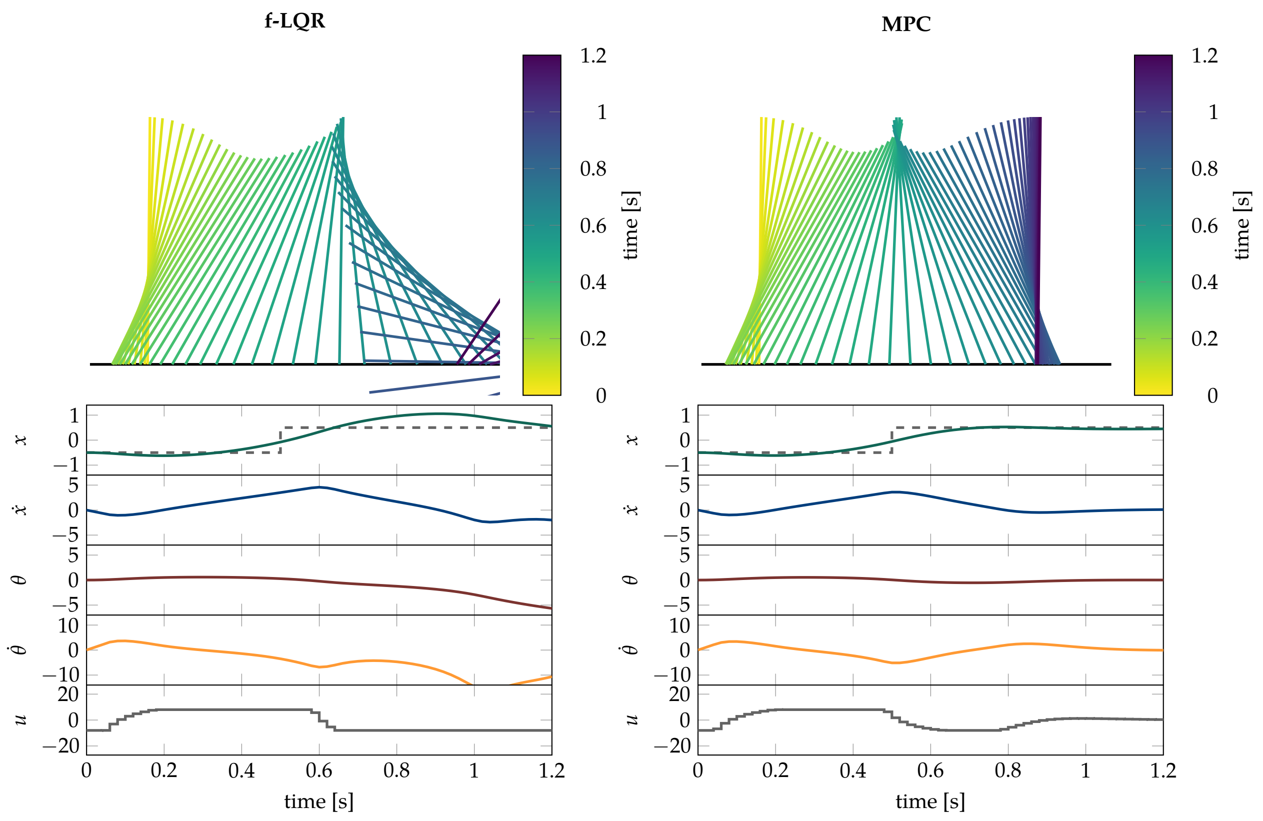


Figure 19: Comparison between the control behavior of f-LQR (left) and MPC (right) for an inverted pendulum on a cart with constrained input ($-8 \le u \le 8$, enforced by clipping). The reference position jumps at 0.5 s from -0.5 to 0.5. Everything else as in Figure 18.

where $\mathbf{z}$ is the vector that collects all optimization variables $x_{1 \ldots T}$ and $u_{0 \ldots N-1}$, H defines the quadratic and h the linear part of the quadratic cost function. The dynamics is encoded as equality constraints with A_{eq} and b_{eq}, state constraints as inequality constraints with A_{in} and b_{in}.

A comprehensive overview on optimization methods is given, e. g., by Nocedal and Wright [103], or by Boyd and Vandenberghe [22].

[103] Nocedal and Wright, *Numerical Optimization*, 2006

[22] Boyd and Vandenberghe, *Convex Optimization*, 2004

2.4.3 Reference Tracking

Until the end of the horizon, a change in reference can be incorporated by changing the cost function to the more general form

$$l_i(x_i, u_i) = (x_i - x_i^{\text{ref}})^\intercal Q_i (x_i - x_i^{\text{ref}}) + u_i^\intercal R_i u_i \tag{2.27a}$$

$$l_T(x_T) = (x_T - x_T^{\text{ref}})^\intercal Q_T (x_T - x_T^{\text{ref}}), \tag{2.27b}$$

so that the cost function penalizes the deviation from the reference trajectory $\mathbf{x}^{\text{ref}}$. This leads to additional linear terms in the objective function of (2.26a).

Again, it is important to note that this reference tracking can offer lookahead if the reference is known a-priori.

2.5 Receding Horizon Control

Most control systems run indefinitely. Therefore, most model predictive controllers are implemented in a receding horizon fashion: After each measurement, the optimal control problem is solved up to the end of the horizon, starting with the current state. Only the first control action is executed, after which the next measurement is made. Figure 20 depicts this procedure. The fixed-length horizon is advanced by one time step at each sampling time, hence the name "receding horizon".

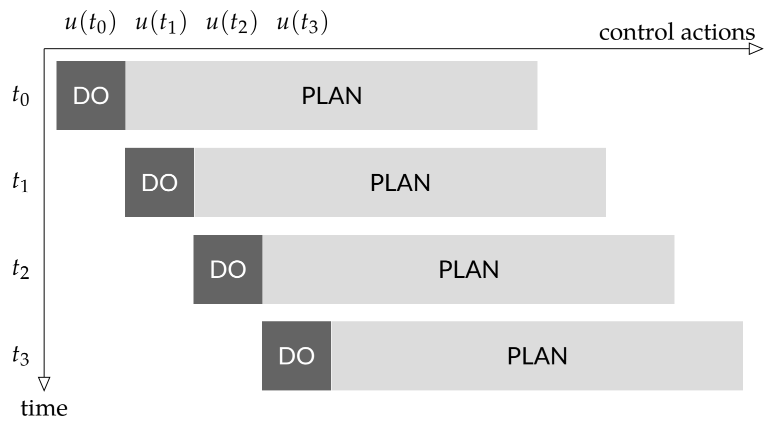


Figure 20: The receding horizon principle. The optimal control is planned for the whole horizon, but only the first control action is executed on the system. After one sampling interval, the plan is calculated anew.

2.5.1 Accounting for the Infinite Horizon

For a system running indefinitely, the cost objective in Equation (2.25a) should be the infinite sum

$$\sum_{i=0}^{\infty} l_i(x_i, u_i), \tag{2.28}$$

but, of course, it is not possible to optimize this objective, due to the infinite number of variables. Instead, the basic MPC (2.25) is formulated with a receding horizon of finite length. This change in the objective function can lead to instabilities and performance loss. Therefore, it is often attempted to account for the infinite horizon MPC by adjusting the terminal weight matrix Q_T accordingly [90, §6.2], or by constraining the final state x_T to be in a suitable feasible set.

[90] Maciejowski, *Predictive Control with Constraints*, 2002

The Riccati Terminal Weight

For the unconstrained linear quadratic problem, finding a steady-state solution for the infinite horizon cost is possible by carrying out the Riccati recursion to the steady-state solution (Section 2.3). This amounts to using the infinite-horizon LQR for the time steps after the horizon ends. The necessary terminal weight is obtained by solving the discrete-time algebraic Riccati equation

$$V_\infty^{\mathrm{R}} = A^\intercal V_\infty^{\mathrm{R}} A - (A^\intercal V_\infty^{\mathrm{R}} B)(R + B^\intercal V_\infty^{\mathrm{R}} B)^{-1}(B^\intercal V_\infty^{\mathrm{R}} A) + Q \qquad (2.29)$$

for the terminal weight $Q_T = V_\infty^{\mathrm{R}}$. The algebraic Riccati equation has a solution if the pair (A, B) is controllable. [20, §4.1]

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

The Lyapunov Terminal Weight

Another option for determining a terminal weight is to use the solution to the discrete-time Lyapunov equation

$$V_\infty^{\mathrm{L}} = A^\intercal V_\infty^{\mathrm{L}} A + Q. \qquad (2.30)$$

Using $Q_T = V_\infty^{\mathrm{L}}$ as a terminal weight amounts to not issuing control actions after the end of the horizon. Along the horizon, this leads to more aggressive control behavior than the Riccati terminal weight. Because no control is used after the end of the horizon, the discrete-time Lyapunov equation only has a solution if A is asymptotically stable. [90, §6.2]

[90] Maciejowski, *Predictive Control with Constraints*, 2002

2.6 Extensions

Due to its simplicity in the mathematical formulation, discrete-time optimal control is a flexible framework that is easily extensible for specific needs. In addition to the reference tracking, which is somewhat different for dynamic programming and general MPC, two other extensions are used in this thesis: The use of disturbance predictions and nonlinear dynamics.

2.6.1 Incorporating Disturbance Predictions

Often there are disturbances in a control system that are known beforehand or can be predicted to a certain extend. Consider the dynamics

$$x(t_{k+1}) = Ax(t_k) + Bu(t_k) + d(t_k), \qquad (2.31)$$

with additional time-dependent disturbance $d(t_k)$. A prediction $\tilde{d}(t_k)$ of the disturbance can easily be incorporated by adding this disturbance prediction to the state-transition function used in the algorithm. For MPC, this means adding it to the equality constraint (2.25c) and for DP this means adding it for all state transitions in the recursion (2.9).

If the disturbance can be predicted to high precision, this can increase the controller performance significantly. Disturbance prediction is frequently used, e. g., for building control with MPC [107].

[107] Oldewurtel et al., “Reducing Peak Electricity Demand in Building Climate Control Using Real-time Pricing and Model Predictive Control,” 2010

2.6.2 Nonlinear Dynamics

Standard MPC works well for linear systems because the underlying optimization problem usually results in an easy-to-solve form, e. g., a quadratic program in the case of the common ℓ_2 norm cost. Linear MPC is often also applied to nonlinear systems that are sufficiently linear close to the operating point. However, when a system is strongly nonlinear or needs to leave the area close to the linearization point, nonlinear MPC methods are necessary.

There is a large body of literature on nonlinear MPC methods, see, e. g., [75], [3] for an overview of the topic.

[75] Kouvaritakis and Cannon, *Nonlinear Predictive Control: Theory and Practice*, 2001

[3] Allgöwer et al., “Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview,” 1999

In this thesis we use the common approach of sequential linearization and optimization to solve the underlying nonlinear program. The resulting algorithm is sequential quadratic programming (SQP) or sequential linear programming (SLP), depending on the cost function.

For a general discrete-time system of the form

$$x(t_{k+1}) = f(x(t_k), u(t_k)), \tag{2.32}$$

the linearization is done along the horizon for each time step

$$x_{i+1} \approx f(\bar{x}_i, \bar{u}_i) + \underbrace{\frac{\partial}{\partial x_i} f \bigg|_{\bar{x}_i, \bar{u}_i}}_{A_i} (x_i - \bar{x}_i) + \underbrace{\frac{\partial}{\partial u_i} f \bigg|_{\bar{x}_i, \bar{u}_i}}_{B_i} (u_i - \bar{u}_i), \tag{2.33}$$

where the Jacobians A_i and B_i are calculated along an existing trajectory. This trajectory can be initialized, e. g., with a steady state, zero, or the optimized trajectory from the last time step. Equation (2.33) can then be used to replace the equality constraint (2.25c) to account for the nonlinear dynamics function.

The linearization along the trajectory is done iteratively, usually several times per time step. However, for systems with sufficiently high control rate, it can be enough to linearize once per time step. [35]

[35] Diehl, Bock, and Schlöder, “A Real-time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control,” 2005

Using nonlinear MPC methods makes it possible to execute complex motions, such as the swing-up of an inverted pendulum on a cart (Figure 21), that are not possible with linear MPC. The linear MPC fails in this case because the system dynamics is highly nonlinear between the two reference points.

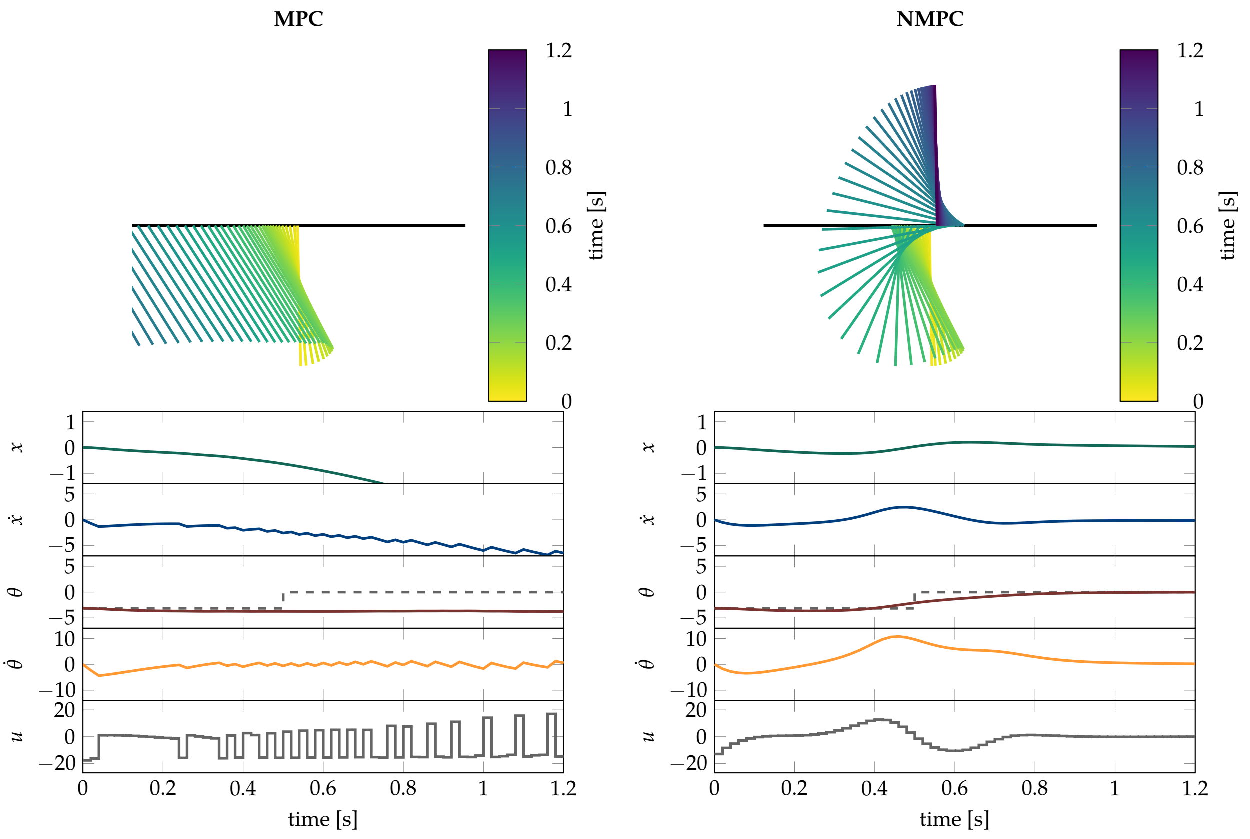


Figure 21: Comparison between the control behavior of linear MPC (left) and nonlinear MPC based on SQP (right) for a swing-up of an inverted pendulum on a cart. The reference angle jumps from π to 0 at $t = 0.5\,\mathrm{s}$. The linear MPC is linearized about the target position. Everything else as in Figure 18.

3 Adaptive Control

Controllers that are able to adjust themselves to different environmental conditions are usually subsumed under the umbrella term "adaptive control". Adaptivity is an important concept because there is always a discrepancy between assumed models and the real world. In fact, for many control systems a big proportion of development time is spent in hand-crafting and tuning models. Adaptive control systems aim at learning models at runtime, or at adjusting them if the environment is changing.

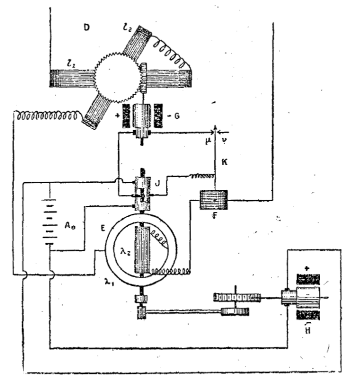


Figure 22: Schematics of one of the first adaptive controllers in the literature, implemented in hardware. [83]

The idea of adaptive control is quite old, and many sources (e. g., [5]) report the paper by Leblanc [83] as the first paper on this topic, see Figure 22 for an illustration of the hardware implementation. Since then, adaptive controllers have found many applications in the automatic control of technical systems.

[83] Leblanc, "Sur l'électrification des chemins de fer au moyen de courants alternatifs de fréquence élevée," 1922

[5] Ariyur and Krstić, *Real-time Optimization by Extremum-Seeking Control*, 2003

Unfortunately, the term *adaptive control* has no clear definition and the meaning often depends on the author. For example, in their textbook on this topic, Åström and Wittenmark [7] define "an adaptive controller [as] a controller with adjustable parameters and a mechanism for adjusting the parameters". The definition of adaptive control now depends on the definition of a parameter, which also is not always clear.

[7] Åström and Wittenmark, *Adaptive Control*, 1994

Usually, the states are expected to change more rapidly and as direct response to control actions and disturbances. Parameters, on the other hand, are changing more slowly and are therefore seen on a different time scale. [43, §3.1] For example, while position, velocity and acceleration are usually states, the physical properties like weight or length of mechanical components are parameters. But also slowly-varying states like the angle of attack of a plane can be seen as parameters, since they change slowly, but govern other parts of the overall dynamics. In hierarchical control systems, the states of a higher level can be viewed as parameters of a lower one.

[43] Filatov and Unbehauen, *Adaptive Dual Control*, 2004

3.1 Types of Adaptive Controllers

Typically, adaptive controllers are classified into four different types of adaptive control schemes [7, 123]:

[7] Åström and Wittenmark, *Adaptive Control*, 1994

[123] Sastry and Bodson, *Adaptive Control: Stability, Convergence and Robustness*, 2011

Gain Scheduling This is the simplest case of an adaptive control system. Based on a pre-defined condition (for example, one of the states or parameters being in a certain range), the control system switches between different setpoints or operating conditions. A static feedback control law is provided for every setpoint. The system is adaptive because the local controllers are tuned for each setpoint individually. [84]

[84] Leith and Leithead, "Survey of Gain-scheduling Analysis and Design," 2000

Model-Reference Adaptive Control This is a form of extremum seeking control, where a performance criterion based on a reference trajectory is optimized by parameter tuning. Usually this is done with a gradient-based optimization scheme, where the gradients are evaluated by numeric differentiation. [77]

[77] Landau, "A Survey of Model Reference Adaptive Techniques–Theory and Applications," 1974

Self-Tuning Regulators This is a model-based control strategy. A model of the system dynamics is learned or maintained online (e. g., via parameter tracking in a parametric model). The model of the system is then used to calculate a controller with standard model-based control techniques. [9]

[9] Åström and Wittenmark, "On Self-tuning Regulators," 1973

Dual Control[1] This is the theoretically ideal way of performing adaptive control. An optimal dual controller would perform nonlinear stochastic optimal control on a system state that comprises both the states and the belief over the system dynamics. This type of controller is of such complexity that it is challenging to apply it in practice. [148]

[1] Dual control is an integral part of this thesis and will be covered in depth in Part III.

[148] Wittenmark, "Adaptive Dual Control Methods: An Overview," 1995

Figure 23 shows an overview of adaptive control methods in relation to time of information acquisition. While all adaptive control methods are online methods, only dual control takes future measurements into account.

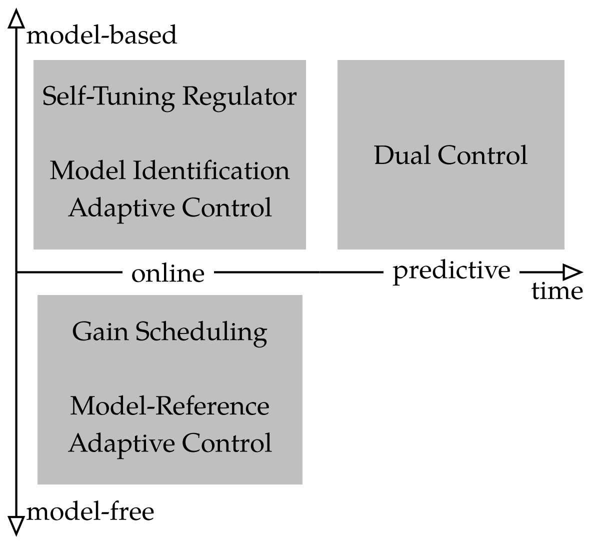


Figure 23: Classification of adaptive control methods, based on the time of information acquisition (left to right) and the use of a model (top and bottom).

Note the distinction between model-free and model-based adaptive controllers. From the above classifications, *gain scheduling* and *model-reference adaptive control* are model-free, which means that they are tuning a control-law[2] directly. The other two classes, *self-tuning regulators* and *dual control*, qualify as model-based schemes. This means that the model of the process is adapted and a controller is synthesized from this model according to some principle.

[2] A control-law is sometimes also called "policy", especially in the reinforcement learning community.

3.2 Model Identification Adaptive Control

The definition of adaptive control by Åström and Wittenmark [7] is relatively broad and can sometimes lead to confusion. Also, the distinction between adaptive and non-adaptive is much harder to make for model-free controllers than for model-based ones.[3] In the following, we therefore only consider model-based controllers, where we assume that the system is governed by a true, but unknown, dynamics function

[7] Åström and Wittenmark, *Adaptive Control*, 1994

[3] Essentially all model-free controllers (except static state-feedback controllers) are adaptive because they depend on changing parameters. This complicates the discussion quite a bit.

$$x_{t+1} = f_t(x_t, u_t) \tag{3.1}$$

that depends on the states x, the inputs u and the time instance t. The adaptive controller builds or updates an estimate $\hat{f}_t$ of the true dynamics f_t and uses it to calculate the control input

$$u_t = c(\hat{f}_t, x_t), \tag{3.2}$$

where c is the model-based controller.[4] Consequently, the controllers in this thesis can all be seen as model identification adaptive controllers in the broader context of adaptive control.

[4] In the case of optimal control, also a cost function l_t would be necessary.

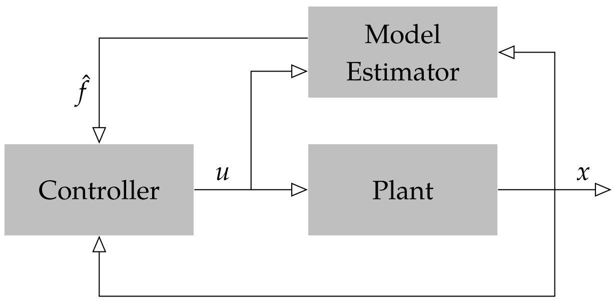


Figure 24: Structure of a model identification adaptive controller. The estimator maintains a model $\hat{f}$ of the dynamics which is used by the controller together with the state x to generate the control signal u.

Definition 3.1
A model identification adaptive controller generates control inputs using an continually estimated model of the uncertain and possibly time-varying system dynamics.

Henceforth, we use *adaptive control* as a synonym for *model identification adaptive control* to simplify reading.

3.3 System Identification and Adaptive Control

Sometimes there is confusion between system identification [86] (in combination with a regular controller) and adaptive control. This comes from the fact that both techniques use similar mathematical methods. The difference lies in the time of information acquisition: While system identification is done offline and provides a model for a controller, adaptive control is an online method. While some types of adaptive control even try to predict the process of information acquisition to the future, all adaptive methods use the measurements during runtime to adjust the controller (see also Figure 23).

[86] Ljung, *System Identification: Theory for the User*, 1999

It is clear that both system identification as well as adaptive control can "adapt" to different systems, but while an adaptive controller does the learning online while controlling, system identification is separate from the controller. Once the system identification is done, a controller is synthesized and subsequently used in the control problem. Usually parameters that do not change between the identification and the use of the control system are identified with system identification. Other parameters can only be identified online, either because they are subject to change or because the specific problem instance can not be identified in advance.

3.4 Adaptivity and Robustness

Adaptivity is not the only way to deal with uncertainties in the model or the environment. Another possible option is robust control [152]. In robust control systems, the controller is designed to be resilient against uncertainties. This means that deviations in the nominal parameters and disturbances do not endanger the system's stability. This is usually done by considering worst case examples and controlling the system in a way that all worst case examples are still stable.

[152] Zhou and Doyle, *Essentials of Robust Control*, 1998

If a control system is robust to parameter changes, one could argue that adaptivity is not necessary. But there are still reasons to design adaptive systems as well. Because robust control systems need to be compatible with multiple worst case instances of a given problem, in many situations infeasibility can arise: Not for all scenario distributions or combinations of worst cases a robust controller does exist. [24] But also when feasibility is given for the uncertain system, robust control trades performance for worst case stability. If the control system is adaptive, the performance level can be much higher after the system is learned because many of the unlikely examples can be ruled out by the learning process.

[24] Calafiore and Campi, "The Scenario Approach to Robust Control Design," 2006

Of course, adaptivity and robustness are not exclusive. A robust control system can be designed relative to the current state of an adaptive control system, combining the features of both. [62]

[62] Ioannou and Sun, *Robust Adaptive Control*, 1996

Part II

Nonparametric Disturbance Correction

4 Gaussian Processes for Periodic Error Correction

Screws and gears are not the only source of periodically recurring errors in dynamical systems. Every system that is tied to the ubiquitous day-night cycle (like building control or energy systems) or to recurring movements (like a beating heart or a satellite) suffers from periodic errors. Since these effects are often small relative to the required control precision, they are in practice usually neglected in the controller design. For high-precision control systems, however, such errors can be the dominant source of problems.

Correcting errors only after they are measured leads to a delay in the error correction. If the errors can be anticipated, the control performance can be significantly improved. While stochastically arising errors can not be preempted systematically, periodic effects are amenable for prediction: Since their future resembles their past, extrapolation is easier and more structured. Based on this idea, we present a framework for identification and control of periodic effects. Our framework continually performs identification at runtime, and is thus applicable to stochastic time varying systems.

The correction of periodic errors has repeatedly been studied. The "Very Large Telescope" uses an internal parametric model for the known error sources, and a Kalman filter [64] as an estimator for the model parameters. [41] High-precision tracking of spacecrafts on periodic trajectories was addressed by Crassidis and Markley [29], based on predictive filtering using an extended Kalman filter. To predict the beating motion of a human heart, extended Kalman filtering for state estimation was used by Yuen, Novotny, and Howe [150], allowing the nonlinear model to change over time. Concerning the use of learning based models for control, there is a wide range of literature available in the context of adaptive control. For methods based on model predictive control see, e. g., the recent works [71], [10], [137].

[64] Kálmán, "A New Approach to Linear Filtering and Prediction Problems," 1960

[41] Erm and Sandrock, "Adaptive Periodic Error Correction for the VLT," 2003

[29] Crassidis and Markley, "Predictive Filtering for Nonlinear Systems," 1997

[122] Särkkä, *Bayesian Filtering and Smoothing*, 2013

[150] Yuen, Novotny, and Howe, "Quasiperiodic Predictive Filtering for Robot-assisted Beating Heart Surgery," 2008

[71] Kocijan et al., "Gaussian Process Model Based Predictive Control," 2004

[10] Aswani et al., "Provably Safe and Robust Learning-Based Model Predictive Control," 2013

[137] Tanaskovic et al., "Adaptive Model Predictive Control for Constrained Linear Systems," 2013

In contrast to previous methods for periodic error correction, the approach presented here does not rely on a pre-specified finite-dimensional model class. Instead, we propose a nonparametric framework based on Gaussian process (GP) regression that is frequently used for system identification. It is closely related to least-squares regression, which is the most commonly employed technique in system identification, but is based on a probabilistic interpretation, which can be used to guide exploration during identification [56]. There is recent work on using GPs for state filtering [70] and on modeling and control of

[56] Hennig, "Optimal Reinforcement Learning for Gaussian Systems," 2011

[70] Ko and Fox, "GP-BayesFilters: Bayesian Filtering Using Gaussian Process Prediction and Observation Models," 2009

nonlinear systems [111], [55]. The idea of using the learned model in predictive control is conceptually similar to [71], [10], [89], with the key difference that we use a GP to predict time varying effects.

[111] Pillonetto et al., "Kernel Methods in System Identification, Machine Learning and Function Estimation: A Survey," 2014

[55] Hall, Rasmussen, and Maciejowski, "Modelling and Control of Nonlinear Systems Using Gaussian Processes with Partial Model Information," 2012

[71] Kocijan et al., "Gaussian Process Model Based Predictive Control," 2004

[10] Aswani et al., "Provably Safe and Robust Learning-Based Model Predictive Control," 2013

[89] Maciejowski and Yang, "Fault Tolerant Control Using Gaussian Processes and Model Predictive Control," 2013

A Gaussian process model is parametrized by two objects: mean and covariance function. When used in system identification, in particular the choice of covariance function has a strong effect on performance, and requires consideration of the dynamics to be identified. Although the literature knows "universal" covariance functions that can technically approximate any continuous function [134, 101], this notion only applies in the infinite limit, and can be subject to extremely slow convergence [145, 144]. Hence, the choice of covariance function is often critical in practice.

[134] Steinwart, "On the Influence of the Kernel on the Consistency of Support Vector Machines," 2002

[101] Micchelli, Xu, and Zhang, "Universal Kernels," 2006

[145] van der Vaart and van Zanten, "Rates of Contraction of Posterior Distributions Based on Gaussian Process Priors," 2008

[144] van der Vaart and van Zanten, "Information Rates of Nonparametric Gaussian Process Methods," 2011

In this chapter, the focus lies on the identification of quasiperiodic systems (Section 4.1) by using a specific model class involving periodicity (Section 4.2.1). The identification of hyperparameters is performed by exploiting the structure of the problem at hand (Section 4.2.2). We focus on the case where the only nonlinearity is the periodic effect, and use model predictive control to achieve optimal closed-loop performance (Section 4.3.1). A reformulation in the form of a tracking problem is proposed, which offers simple implementation and facilitates analysis of the control performance (Section 4.3.2). To show the qualitative properties of this framework, we apply it to a simulated, educational problem (Section 4.4). While GPs and MPC are well studied techniques, the main contribution of this work is a combination that is tailored to quasiperiodic functions, allowing for extrapolation and efficient computation, which is crucial for online identification and control.

4.1 Problem Statement

Consider the continuous dynamical system

$$\dot{x}(t) = A_c x(t) + B_c u(t) + g(t), \tag{4.1}$$

composed of a linear system with continuous dynamics matrices A_c and B_c, and a nonlinear time varying function $g : \mathbb{R}_+ \to \mathbb{R}^{n_x}$, where $x \in \mathbb{R}^{n_x}$ denotes the state and $u \in \mathbb{R}^{n_u}$ the input. For simplicity, full state measurement is assumed. The structure of (4.1) could be chosen more generally—Gaussian process models can also learn nonlinear functions of the state and input. We opt for this linear formulation with nonlinear external reference here to keep the resulting control problem conceptually clear and computationally simple. If needed, the definition can also be adapted to a nonlinear system using a nonlinear model predictive control technique [3].

[3] Allgöwer et al., "Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview," 1999

The disturbance function $g(t)$ captures nonlinear time dependent effects, in particular we focus on systems exhibiting some form of periodic behavior. Systems with time dependent errors of periodic characteristic appear in different application areas, such as building temperature control [51], beating-heart surgery [150] or electrical power grids [138]. For strictly periodic functions, there exists a constant period λ, such that $g(t + n\lambda) = g(t)$ for $n \in \mathbb{N}$. However, error sources in real systems are often not perfectly periodic in this sense, they show various forms of phase-shift, deformation and desynchronization. To address this issue, we generalize to consider locally periodic functions. These are functions for which $g(t) \approx g(t + n\lambda)$ for $n\lambda \ll \ell$ and $g(t) \not\approx g(t + n\lambda)$ for $n\lambda \gg \ell$, where ℓ is some measure of temporal locality. Intuitively speaking, local periodicity means that the periodicity is not strict, i. e. variations are allowed. In particular, this covers functions that vary on a slower time-scale, e. g., a decaying oscillation or an oscillation with long-term change in shape.

[51] Gondhalekar, Oldewurtel, and Jones, "Least-restrictive Robust MPC of Periodic Affine Systems With Application to Building Climate Control," 2010

[150] Yuen, Novotny, and Howe, "Quasiperiodic Predictive Filtering for Robot-assisted Beating Heart Surgery," 2008

[138] Taylor and McSharry, "Short-term Load Forecasting Methods: An Evaluation Based on European Data," 2007

We consider the case where a linear model (matrices A_c and B_c) is available and the goal is to infer the disturbance function $g(t)$ online from measurements. This is motivated by the fact that often a nominal model is derived either from physical considerations or an offline system identification step.

At every measurement time t_k, the system goes through the following process:

1. Measure $x(t_k), \dot{x}(t_k)$
2. Construct observation for $g(t_k)$, update model for $g(t)$
3. Compute control input $u(x(t_k), g(t_k))$ and apply to the system

4.2 GPs for Quasiperiodic Functions

GP regression is a general framework for nonlinear regression, see Chapter 1 for an introduction. Note that in the current chapter we use the time t as the function argument, since this chapter is concerned with time-series forecasting.

As mentioned above, in the context of our particular setup, the GP framework may in fact also be used to construct probabilistic models for fully nonlinear systems $g(x, u, t)$, without major changes. However, we focus on the simpler case of $g(t)$, allowing for direct incorporation in stochastic predictive control techniques.

This section proposes a covariance function suitable for the identification of quasiperiodic functions and presents an efficient technique for hyperparameter optimization.

4.2.1 Quasiperiodic Covariance Functions

The way to construct a periodic hypothesis class, and the central idea of this chapter, is to construct a covariance function that focuses prior probability mass on locally periodic functions. Among the most popular kernels for regression purposes is the square exponential[1] kernel

$$k_{\text{SE}}(t, t'; \ell_{\text{SE}}) = \exp\left(-\frac{|t-t'|^2}{2\ell_{\text{SE}}^2}\right), \tag{4.2}$$

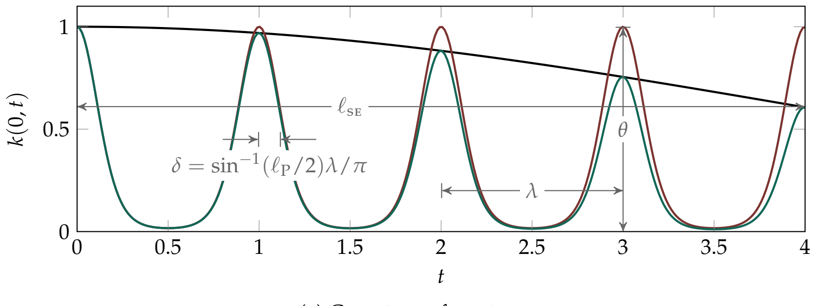
[1] The square exponential kernel is also known as *squared exponential kernel*, *Gaussian kernel*, or *radial basis function*.

with length-scale ℓ_{SE}. This kernel gives a stationary model which does not allow for structured extrapolation. MacKay proposed a periodic covariance function, based on a sine-transformation of the input [92]

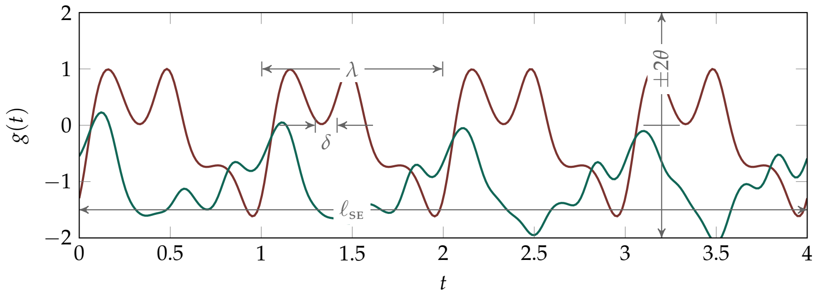
[92] MacKay, "Introduction to Gaussian Processes," 1998

$$k_{\text{P}}(t, t'; \ell_{\text{P}}, \lambda) = \exp\left(-\frac{2\sin^2\left(\frac{\pi}{\lambda}(t-t')\right)}{\ell_{\text{P}}^2}\right), \tag{4.3}$$

with length-scale ℓ_{P} and period-length λ. Function values $g(t), g(t')$ jointly sampled from Gaussian process priors with this covariance function are perfectly correlated if $|t-t'| = \lambda$, resulting in identical function values for points that are one period length apart. Thus, sampled functions are perfectly periodic with period λ. Within this period length, samples vary on a typical regularity length scale of $\delta = \sin^{-1}(\ell_{\text{P}}/2)\lambda/\pi$, over a range with standard deviation $\theta = 1$ (Figure 25).



(a) Covariance functions



(b) Samples from GPs with these covariance functions

Figure 25: Kernel combination, covariance function and samples. **Top:** The compound kernel k_{C} (——,(4.4)) is the product of k_{SE} (——, (4.2)) and k_{P} (——, (4.3)). **Bottom:** Samples drawn from Gaussian process priors using these covariance functions (same colors). Samples using the periodic kernel are perfectly periodic, while samples using the compound kernel are only periodically similar on a scale controlled by the parameter ℓ_{SE} of (4.4). This local periodicity can be used to increase modeling flexibility.

For many systems, strict periodicity is too strong an assumption. For example, the weather is periodic, but also stochastic, which has an effect on the periodic temperature in a building. Therefore, modeling external errors with perfect periodicity can lead to severe overfitting and low extrapolation performance. To weaken the perfect correlation, we use the fact that the kernel property is closed under multiplication and addition (i. e. kernels form a semiring [115, §4.2.4]): Although the product of two Gaussian processes is not a Gaussian process, the product of two kernel functions is also a kernel and therefore a valid covariance function for a Gaussian process.

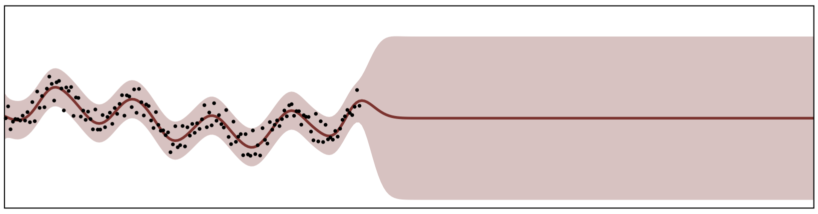
[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

This property of covariance functions is useful to construct composite covariance functions, either automatically from data [39] or manually. Since we assume access to physical knowledge about the system, we constructed a suitable kernel in a qualitative manner. Multiplying the periodic kernel with a broad square exponential gives another kernel whose corresponding Gaussian process condenses mass at periodic functions that change over time

[39] Duvenaud et al., "Structure Discovery in Nonparametric Regression through Compositional Kernel Search," 2013

$$k_{\mathrm{C}}(t, t'; \theta^2, \ell_{\mathrm{SE}}, \ell_{\mathrm{P}}, \lambda) = \theta^2 \cdot k_{\mathrm{SE}}(t, t'; \ell_{\mathrm{SE}}) \cdot k_{\mathrm{P}}(t, t'; \ell_{\mathrm{P}}, \lambda), \tag{4.4}$$

with signal variance θ^2 and the other parameters as stated above. This kernel considers two input times similar if they are similar under both the square exponential *and* the periodic kernel. If $\ell_{\mathrm{SE}} \gg \lambda$, this allows encoding a decay in the covariance over several oscillations. The different covariance functions are shown in Figure 25 (a), exemplary randomly sampled functions from Gaussian processes with those covariance functions are shown in Figure 25 (b). The posteriors of GPs with aperiodic and periodic covariance, trained on periodic data, are shown in Figure 26. In the region far away from data points the predictions are equal, whereas close predictions show significantly more structure with the locally periodic kernel.


(a) SE kernel

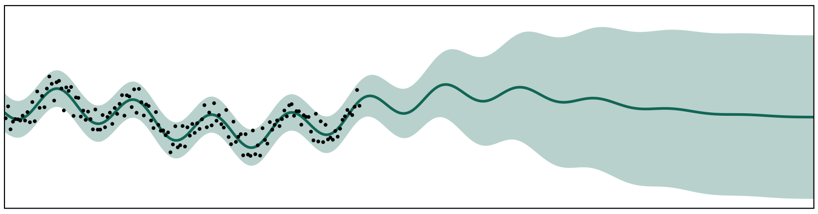

(b) Locally periodic kernel

Figure 26: Comparison of Gaussian process posteriors (posterior mean function as thick line, shaded region covers two marginal standard deviations) arising from the same periodic data (···) for the square exponential kernel (—/▨) and a product of periodic and square exponential kernel (—/▨). The locally periodic kernel provides a richer extrapolation, which can be used for improved predictive control.

Figure 26 (b) illustrates the key benefit of this approach: With increasing distance from data, the prediction degrades gracefully back to the zero mean. If a not perfectly periodic function would be predicted with a purely periodic kernel, prediction and reality could run out of phase over time, leading to bad predictive behavior. The proposed locally periodic covariance function circumvents this problem.

4.2.2 Custom Parameter Optimization

The combined kernel (4.4) has four hyperparameters, which will henceforth be subsumed in the vector $\boldsymbol{\eta} := (\theta^2, \ell_{\text{SE}}, \ell_{\text{P}}, \lambda)$. This includes the period length λ of the periodicity. Inferring good values for $\boldsymbol{\eta}$ is important for good modeling performance. The parameter optimization is done by type-II maximum likelihood / maximum a-posteriori, as described in Section 1.5. However, using this optimization scheme in a non-modified ("vanilla") fashion usually does not work in periodic or quasiperiodic settings.

The log-likelihood surface for $\boldsymbol{\eta}$ is, especially for periodic likelihoods, not convex. Figure 27 shows a slice through this surface along the periodicity parameter λ for a quasiperiodic dataset. Standard numerical optimizers will, thus, usually return suboptimal local extrema of this function. An interesting observation in our specific context is that the periodic structure of the covariance function and the data is reflected in this hyperparameter likelihood as well. The reason for this is a harmonic effect: If the data has a true period of λ, then periodic functions whose periodicity is an integer multiple of λ also fit the data well, resulting in low values in the log likelihood

$$\log p(\mathbf{y}|\mathbf{t}, \boldsymbol{\eta}) = -\frac{1}{2}\mathbf{y}^{\intercal} K(\boldsymbol{\eta})^{-1}\mathbf{y} - \frac{1}{2}\log|K(\boldsymbol{\eta})| - \frac{N}{2}\log 2\pi. \tag{4.5}$$

Intuitively, this can be compared to a function with periodicity λ, which could also be considered as a function with period length 2λ. To see this, recall from Figure 25 that the periodic functions can have an arbitrary recurring pattern in each repetition. By inspecting Equation (4.5), we can gain intuition and notice that the likelihood is a nonlinear function of terms of the form (4.3). Since each of these terms is periodic in λ^{-1}, the overall function will show periodicity in that term.

This harmonic structure can be exploited by means of a heuristic to increase numerical stability of the optimizer. We designed a customization of a numerical optimizer, outlined in Algorithm 1, that, after convergence to a local minimum $\boldsymbol{\eta} = (\theta^2, \ell_{\text{SE}}, \ell_{\text{P}}, \lambda)$, also evaluates the function value at $\boldsymbol{\eta}' = (\theta^2, \ell_{\text{SE}}, \ell_{\text{P}}, \frac{\lambda}{2})$. If the negative log likelihood at this location is lower, the optimal value and its location is updated, and the bisection is repeated. The locations that are iteratively proposed during the loop (Line 4 of Algorithm 1) are shown as vertical lines (——) in Figure 27. This approach uses the otherwise problematic harmonic structure in the hyperparameter optimization to find better optima.

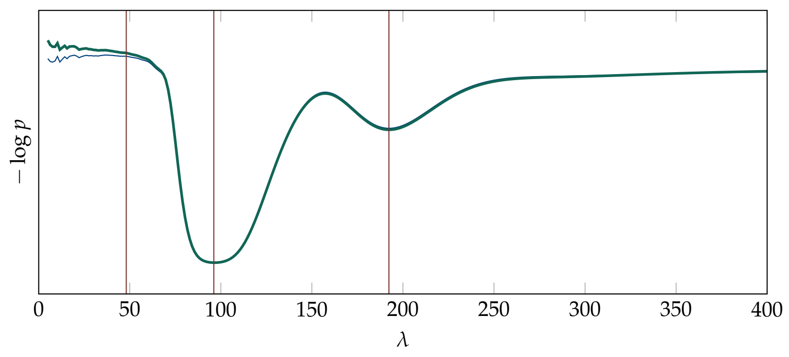


Figure 27: Slice through the likelihood surface along the dimension of the hyperparameter λ, defining the period length of g. Shown are the logarithm of the type-II marginal likelihood (——) and the logarithm of the posterior distribution (——). The shape of the log posterior is dominated by the likelihood, indicating that most prior assumptions are dominated by the observed data. The meaning of the vertical lines is explained in Section 4.2.2.

```
 1: $\eta := (\theta^2, \ell_{\mathrm{SE}}, \ell_{\mathrm{P}}, \lambda)$          ▷ initial guess
 2: $\eta \leftarrow$ LOCALLY_OPTIMIZE($\eta$)          ▷ use std. optimizer
 3: loop
 4:     $\eta' \leftarrow (\theta^2, \ell_{\mathrm{SE}}, \ell_{\mathrm{P}}, \frac{1}{2}\lambda)$          ▷ make proposal
 5:     if nll($\eta'$) > nll($\eta$) then          ▷ compare neg. log lik.
 6:         break          ▷ reject and leave loop
 7:     else
 8:         $\eta \leftarrow \eta'$          ▷ accept proposal
 9:     end if
10: end loop
11: return $\eta$
```

Algorithm 1: Customized parameter optimization. After the convergence of the standard optimizer, the likelihoods for halved period lengths are evaluated. The period length with the lowest negative log likelihood is accepted.

4.2.3 Sampling Methods for Parameter Identification

Instead of using only a maximum likelihood or maximum a-posteriori point estimate, theoretically the goal would be to compute the marginal

$$p(g) = \int p(g|\boldsymbol{\eta})p(\boldsymbol{\eta})d\boldsymbol{\eta} \tag{4.6}$$

using a prior density $p(\boldsymbol{\eta})$ [115, §5.2], which can only be performed approximately at high computational expense. One comparably elaborate and precise way to approximate the posterior distribution over $\boldsymbol{\eta}$ and to marginalize over the unknown parameters $\boldsymbol{\eta}$ is sampling, using a Markov chain Monte Carlo (MCMC) method. We found *shrinking-rank slice sampling* [139] to be particularly well-suited for this task and implemented the method for a comparison between the MAP point-estimate and MCMC sampling.

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

[139] Thompson and Neal, "Slice Sampling with Adaptive Multivariate Steps: The Shrinking-Rank Method," 2010

The left column of Figure 28 shows the resulting marginals on the function g at two different points in the learning process for a quasiperiodic system. As expected, the posterior uncertainty is high after only a few observations, in particular after less than one full period, but collapses to a highly confident distribution after several periods of observations have been collected.

The right column of Figure 28 shows the Gaussian process point estimates for g resulting from MAP inference. From the figure, it is clear that point estimation leads to a more limited, and generally overly confident extrapolation model, especially in early phases of learning, when the dataset does not yet cover several periods. However, MAP offers two advantages that make it attractive from an applied perspective: The first one is computational cost—MCMC sampling can be orders of magnitude more expensive than optimization for a MAP estimate. The second one is an algebraic one: MCMC estimates are mixtures of Gaussian process models (see Figure 28). This means the overall model for the regression function defined by these models is an object challenging to work with. Therefore, applications often use the computationally much less taxing MAP inference.

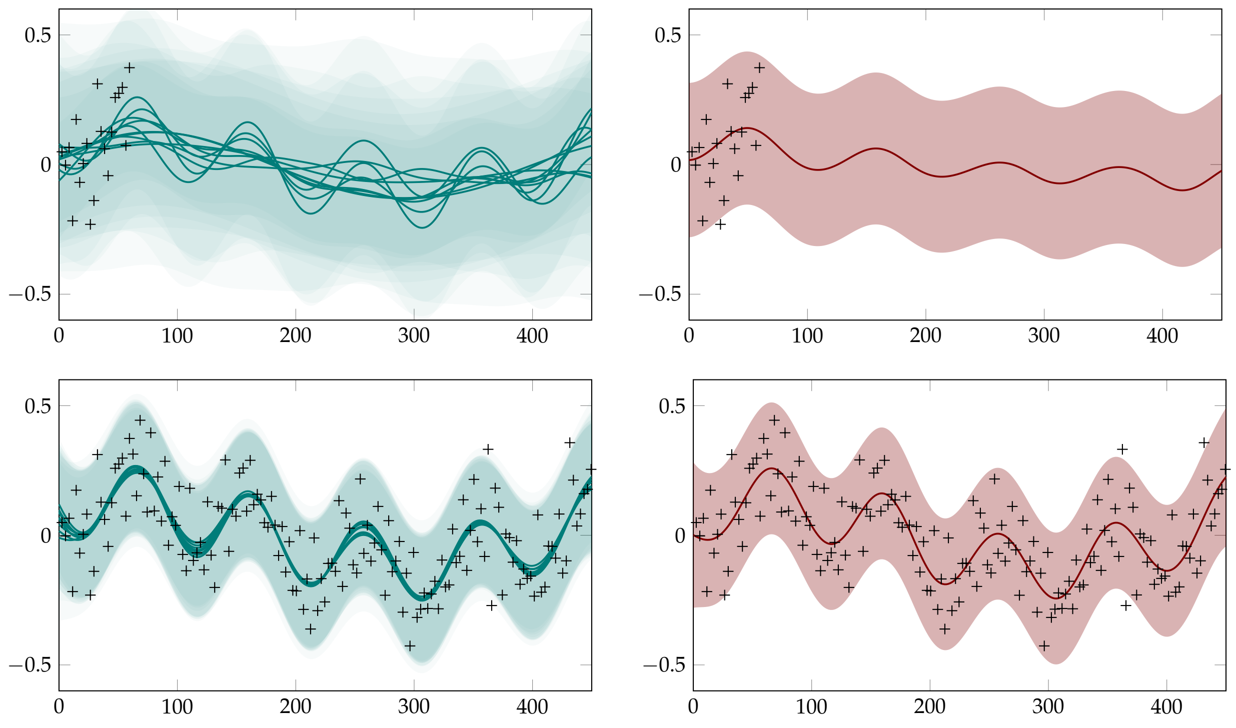


Figure 28: Comparison of Markov chain Monte Carlo inference (left column) with the maximum a-posteriori point estimate (right column) on hyperparameters of the Gaussian process model. **Top:** Initial phase of learning, after only a few observations. **Bottom:** Convergence after observation of several periods.

4.3 GP Predictions in Model Predictive Control

Popular control frameworks supporting the incorporation of feedforward model predictions include linear quadratic regulator (LQR) techniques [104, §8.3], or model predictive control (MPC) [117]. See Chapter 2 for an introduction to these methods.

[104] Ogata, *Discrete-time Control Systems*, 1995

[117] Rawlings and Mayne, *Model Predictive Control: Theory and Design*, 2009

For periodic error correction, we employ an online MPC framework, computing the optimal control input by solving an optimization problem for each measured state. This allows for direct incorporation and updating of the GP model as well as system constraints, such as input constraints.

4.3.1 Discrete-time MPC formulation

A discrete-time MPC approach is used based on the discretization of the model (4.1)

$$x_{k+1} = Ax_k + Bu_k + a_k, \tag{4.7}$$

where A and B are obtained from a zero-order-hold discretization and a_k is a discretization of choice of $g(t)$ at time t_k.

Since the optimal control input is computed at each sampling time based on the current measured state, the model can be updated online. An important aspect and advantage of combining online learning of a continuous time function with MPC is the possibility to decouple the discretization from the sampling time. While in a standard MPC setup, unmodeled effects only become apparent through state measurements and therefore require fast sampling rates, the GP model captures these effects and provides a continuous prediction of their evolution in the future. As a result, the sampling time can be chosen as a multiple of the discretization or prediction interval without sacrificing performance by using the sequence of control inputs in between state measurements. It is clear that an upper bound on the sampling time is imposed by the prediction horizon.

Since the prediction from the Gaussian process model is stochastic and provides a distribution over future function values rather than one particular sequence, stochastic MPC methods offer a natural framework to incorporate the GP model and make use of the posterior model uncertainty. For an overview of recent stochastic MPC methods, see e. g., [74], [97] and the references therein. The model uncertainty is an important advantage over other nonparametric methods like kernel ridge regression or regularized least squares, which do not provide posterior uncertainty.

[74] Kouvaritakis and Cannon, "Stochastic Model Predictive Control," 2014

[97] Mayne, "Model Predictive Control: Recent Developments and Future Promises," 2014

A common cost function in stochastic MPC for regulating the system state to the origin is the expected value of the sum of stage costs

$$J(x_0, p(g), \mathbf{u}) := \mathbb{E}\left[\sum_{i=0}^{T} l(x_i(p(g)), u_i)\right] \tag{4.8}$$

where $p(g) = \mathcal{GP}(g; m^{|\mathbf{t},\mathbf{y}}, k^{|\mathbf{t}})$ denotes the posterior distribution over the function g. If the stage cost l is chosen to be quadratic (which is the case for many practical MPC problems) and since the inputs are deterministic and the GP posterior is Gaussian, the expected value is equivalent to using the mean of the state evolution, i.e. the mean GP prediction of $g(t)$ in dynamics (4.1). The most common stochastic MPC problem hence results in a deterministic formulation by using the GP posterior mean $\tilde{g}(t) = m^{|\mathbf{t},\mathbf{y}}(t)$, and reduces to the certainty equivalent controller.

We consider the case of quadratic stage cost in the following. Since the true function $g(t)$ is unknown, a discrete-time system describing the mean of the state trajectory is approximated by replacing a_k in (4.7) with $\tilde{a}_k$, obtained by discretizing the mean of the GP prediction $\tilde{g}(t)$. $\tilde{\cdot}$ denotes the estimate of the true function inferred from data, and emphasizes that these are generally not the same. The resulting discrete-time system can directly be used in a standard MPC formulation.

4.3.2 Tracking MPC

We propose a different formulation in the following, which simplifies implementation by transforming the regulation problem into a linear tracking problem. The state of the mean discrete-time system at prediction time step $k+n$, starting from state x_k at time step k is given by:

$$\tilde{x}_{k+n} = A^n x_k + \sum_{m=1}^{n} A^{m-1} B u_{k+n-m} + \sum_{m=1}^{n} A^{m-1} \tilde{a}_{k+n-m}, \tag{4.9}$$

where $\tilde{a}_k$ is the discretization of the GP prediction $\tilde{g}$. This is the sum of the linear system and a system driven by $\tilde{a}_k$. The MPC problem for regulating system (4.7) to the origin can be reformulated as a tracking problem for the linear system, tracking a nonlinear reference.

The reference signal is generated from the GP prediction according to (4.9):

$$\mathbf{x}^{\text{ref}} = \begin{bmatrix} \tilde{a}_{k+1} & \cdots & \sum_{m=1}^{n} A^{m-1} \tilde{a}_{k+n-m} \end{bmatrix}. \tag{4.10}$$

The resulting MPC problem is given by

$$\begin{aligned}
(\mathbf{x}^*, \mathbf{u}^*) = \arg\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{n=0}^{T} l(x_n - x_n^{\text{ref}}, u_n) && \text{(4.11a)}\\
\text{s.t.} \quad & x_{n=0} = x(t_k) && \text{(4.11b)}\\
& x_{n+1} = A x_n + B u_n && \text{(4.11c)}\\
& u \in \mathbb{U} && \text{(4.11d)}
\end{aligned}$$

where $\mathbb{U} \subseteq \mathbb{R}^{n_u}$ is a polytopic set defining the input constraints, $\mathbf{x} := [x_0, \ldots, x_T]$ is the state trajectory and similarly for $\mathbf{u}$. The resulting problem is a quadratic program that can be solved efficiently using available optimization software (e. g., [52]) or fast MPC techniques proposed in recent years, such as code generation (e. g., [36]). Because this is an instance of a basic MPC technique, the standard properties of MPC apply. It also allows for a more principled analysis of the closed-loop properties: Extensions in the field of tracking MPC can be applied to ensure stability, such as reference governors, or the periodic MPC approach in [85]. Applying the modified tracking formulation in [85], convergence can be guaranteed if the model $\tilde{g}(t)$ converges to a periodic function.

[52] Grant and Boyd, *CVX: Matlab Software for Disciplined Convex Programming, Version 2.1*, 2014

[36] Domahidi, *FORCES: Fast Optimization for Real-time Control on Embedded Systems*, 2012

[85] Limon et al., "MPC for Tracking Periodic Reference Signals," 2012

Remark 4.1
The discrete-time model (4.7) with the predicted nonlinear term $\tilde{a}_k$ can in principle be used in any linear or linearized state estimation or control method based on state prediction. One example is the Kalman filter. The nonlinear prediction from the GP can be incorporated into the state prediction without complicating the Kalman filter equations. The measurement update of the Kalman filter remains unchanged. The GP predictions increase the performance by providing a better state estimate. This leads to smaller correction terms and smaller posterior variance.

Remark 4.2
Because the point-wise posterior of a Gaussian process is a Gaussian distribution, state constraints can be included in the form of soft constraints, penalizing the amount of constraint violation, or chance constraints, ensuring constraint satisfaction with a certain probability [127, 108].

[127] Schwarm and Nikolaou, "Chance-constrained Model Predictive Control," 1999

[108] Ono and Williams, "Iterative Risk Allocation: A New Approach to Robust Model Predictive Control with a Joint Chance Constraint," 2008

Remark 4.3
The approach can also be applied to stage cost functions and constraints that do not allow for a deterministic representation using the GP model, e. g., a value-at-risk formulation involving the variance of the cost by using sample-based methods to approximate the stochastic MPC problem [25, 124]. GPs fit well in this framework by being generative models from which sample trajectories can be easily drawn.

[25] Campi, Garatti, and Prandini, "The Scenario Approach for Systems and Control Design," 2009

[124] Schildbach et al., "The Scenario Approach for Stochastic Model Predictive Control with Bounds on Closed-Loop Constraint Violations," 2014

Remark 4.4
Depending on the discretization, a trade-off between the mean and variance of a quadratic cost function can be formulated as a deterministic optimization problem using the posterior GP prediction:

$$\mathbb{E}\left[\frac{1}{2} x_n^\intercal Q_n x_n\right] = \mu_n^\intercal Q_n \mu_n + \operatorname{tr}(Q_n \Sigma_n), \tag{4.12a}$$

$$\mathbb{V}\left[\frac{1}{2} x_n^\intercal Q_n x_n\right] = \mu_n^\intercal Q_n \Sigma_n Q_n \mu_n + \frac{1}{2} \operatorname{tr}(Q_n \Sigma_n Q_n \Sigma_n). \tag{4.12b}$$

This is the case whenever the distribution $\mathcal{N}(\mu_n, \Sigma_n)$ of a_k can be directly obtained from the distribution of $g(t)$, e. g., in the case of Euler discretization.

4.4 Numerical Results

The presented method was implemented for a simple problem and evaluated in simulation to show how the state evolution is anticipated by the use of the GP prediction. In Chapter 5, this method will also be evaluated on an experimental application in hardware.

4.4.1 Implementation Details

Since in practice the state and derivative can not be measured directly, they have to be approximated from potentially noisy measurements. A Kalman filter [64] can be used to estimate the state, which increases the performance, especially when the measurement noise is high.

[64] Kálmán, "A New Approach to Linear Filtering and Prediction Problems," 1960

Since the function to be inferred acts as additional input to the dynamics, observations of the disturbance are constructed from observations of the *change* in state, $\Delta x = x_{k+1} - x_k$. We therefore assume $g(t)$ to be piece-wise constant, here a zero-order-hold linearization is used, chosen to be at $t + \frac{1}{2}\Delta t$

$$a_k = Gg(t + \frac{1}{2}\Delta t), \qquad G = \int_0^{\Delta t} e^{A\tau} d\tau, \tag{4.13}$$

where Δt is the sampling time. Note that this is only used to generate the data point, the resulting GP model is still a continuous function. The value $g(t + {}^1\!/_2\Delta t)$ is obtained as solution of the linear system

$$Gg(t + \frac{1}{2}\Delta t) \approx (x_{k+1} - Ax_k - Bu_k)\,, \tag{4.14}$$

which is the data point used to train the GP.

In the experiments presented in the following, the discretization of the mean prediction $\tilde{g}(t)$ is obtained from the exact discretization

$$\tilde{a}_k = \int\limits_0^{\Delta t} e^{A_c \tau} \tilde{g}((k+1)\Delta t - \tau) d\tau, \tag{4.15}$$

evaluated with a standard ODE-solver [37].

[37] Dormand and Prince, "A Family of Embedded Runge-Kutta Formulae," 1980

4.4.2 A Simple Example

As a simple problem for providing intuition, consider the following linear double integrator system with an additive time-periodic component $g(t)$:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + g(t), \qquad g(t) = \begin{bmatrix} \sin(t) \\ \cos(1.3t) \end{bmatrix}. \tag{4.16}$$

The goal is to control the first state of the system to the origin. We use the quadratic cost

$$l(x_n, u_n) = \frac{1}{2} x_n^\intercal Q x_n + \frac{1}{2} u_n^\intercal R u_n \tag{4.17}$$

with diagonal state weight $Q = \operatorname{diag}(100, 0)$. The weight on the control input is set to $R = 1$, allowing for aggressive control behavior. The horizon length of the MPC is set to $T = 15$. State and input constraints are omitted for simplicity.

The system was simulated numerically with a sampling rate of 1 Hz. Figure 29 shows control inputs and resulting state trajectories for a model predictive controller without information about the disturbance g, and a model predictive controller using the posterior mean functions of two periodic Gaussian process regressors as a model for g. One GP is trained for each dimension of the disturbance.

After an identification phase in the first 5 s of the experiment, the GP based controller shows a drastic performance improvement. Omitting this identification phase, the root-mean-square (RMS) error, measured with respect to the origin, drops by 90 %, from 0.94 for the linear model to 0.097 for the GP based controller. Speaking more qualitatively, Figure 29 also shows less residual structure in the controlled state x_1. It is visible that control signals are applied earlier when the prediction is used.

While the GP based controller is effective at reducing the periodic structure from the first controlled state, the regression model itself remains able to predict the periodic error correctly into the future, even when trained exclusively on controlled states. This is possible because the regression model is obtained from the controlled dynamics, so it can account for the shift of periodicity from the states to the control input. This feature of the framework is crucial for identifying controlled systems.

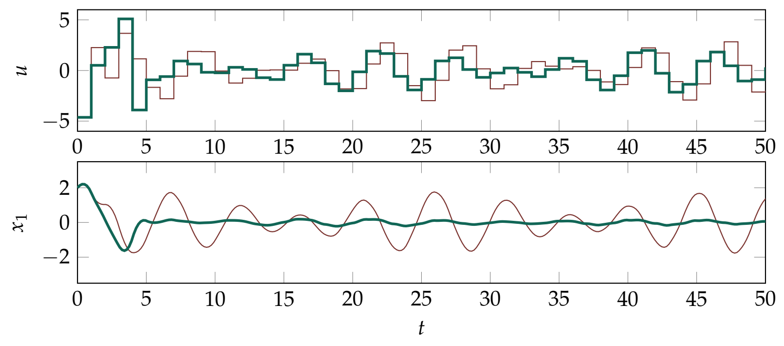


Figure 29: Closed-loop input and output trajectories with MPC control. MPC using a linear model (——) is compared to the GP based MPC controller (——).

4.5 Conclusion

High-precision control of dynamical systems requires precise models, even of minor external error sources. Where analytic models are not available, they can only be constructed numerically from measurements of the system. Periodic error sources are an especially promising domain in this regard, as they can be extrapolated well into the future. We have studied a nonparametric modeling framework based on a carefully crafted Gaussian process prior exhibiting a weak, localized form of periodicity. Because Gaussian regression returns models in the form of stochastic differential equations, they can be combined directly with existing control frameworks. Integration into a model predictive control scheme was investigated, which can evaluate the prediction at a desired temporal resolution.

5 Periodic Error Correction for Telescope Tracking

Our planet is rotating, performing one full turn relative to the sky in a stellar day of about 23 hours and 56 minutes [2, §12.2]. This poses a problem to photographers: Due to the low light intensity of most astronomical objects, the exposure times for astronomical images range from seconds to several hours. Without counteracting the Earth's rotation, the stars travel long distances across a camera's image sensor during exposure. Figure 30 shows an intentional example of this effect.

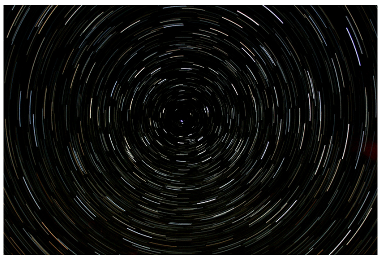


Figure 30: Star motion around the northern celestial pole. The image is composed of several sub-frames, the total exposure time is about half an hour. Image by Robert Vanderbei [146].

[146] Vanderbei et al., *Roque de los Muchachos Observatory*, 2012

[2] Allen and Cox, *Allen's Astrophysical Quantities*, 2000

Efforts to tackle this problem have lead to the development of specialized mechanical devices, actuated *telescope mounts*, that follow the stars' motion. However, mechanical devices are never perfect. A slight imperfection in the shape of a cog or a worm in the gear of a telescope mount leads to pointing errors of usually several arcseconds[1]. Taking into account the fine angular resolution of typical imaging setups, this means several pixels of blur during a long-exposure image.

[1] One arcsecond ($''$) is the 3600th fraction of a degree.

The periodic error correction method based on Gaussian processes (Chapter 4) is suitable to address periodic errors in telescope tracking. In this chapter we first describe the telescope guiding problem (Section 5.1), and then apply the method on both simulation and hardware experiments (Section 5.2).

5.1 The Telescope Problem

Amateur telescope mounts for astronomical imaging are usually built in equatorial design [110, 18], one such mount is shown in Figure 31. The mechanics is designed in a way that the right ascension (RA) axis of the telescope mount is aligned with the Earth's rotational axis when set up properly. This way, only this axis has to be constantly in motion to follow the stars, which simplifies the control problem. The telescope can now be modeled in the form of a linear scalar model

$$\dot{x}(t) = a_c x(t) + b_c u(t) + g(t), \tag{5.1}$$

where a_c is the scalar dynamics, b_c the scalar input gain and $g(t)$ a scalar disturbance. As described in detail in Chapter 4, the disturbance is modeled with a quasiperiodic Gaussian process regression model. It should be noted that in practice, measurements of $\dot{x}(t)$ are generally not available and will be approximated numerically, see also the experimental details in Section 5.2.

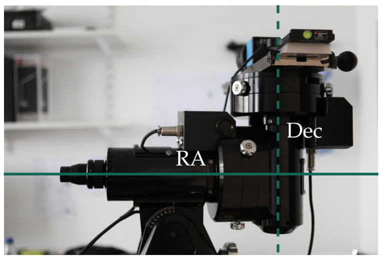


Figure 31: Typical German equatorial mount. When used for astronomical imaging, the right ascension (RA) axis is aligned with the Earth's rotational axis. The declination axis (Dec) is necessary to be able to point at arbitrary positions.

[110] Parker, *Making Beautiful Deep-sky Images: Astrophotography with Affordable Equipment and Software*, 2007

[18] Beish, *Design a German Equatorial Mount for the Planetary Telescope*, 2001

Existing *periodic error correction* systems require careful system identification by the user of the telescope, and still regularly lead to unsatisfactory performance.

A challenge specific to this astronomical application is that state measurements are performed by taking images of guiding stars [110, §1], which requires relatively long exposure times, resulting in the measurement interval reaching the order of magnitude of the error periodicity. This is precisely the domain in which we expect to see utility from a periodic model.

[110] Parker, *Making Beautiful Deep-sky Images: Astrophotography with Affordable Equipment and Software*, 2007

The performance gain one can expect from the use of a periodic model for feed-forward compensation depends on the sampling rate of the control system: If the external error is slow compared to the measurement rate, a locally linear model is sufficient. But if the external error is on the same time scale as the measurement, it helps to use feed-forward control based on GP predictions. With the presented approach it is even possible to choose the control interval smaller than the actual measurement interval. See Section 5.2.1 for a more detailed discussion.

5.2 Experiments

The presented method was implemented for the telescope problem and evaluated on different problems. After testing on a simulated telescope system, where the performance under different measurement frequencies and under sensor failure was analyzed, the proposed method was evaluated in an experiment on a real telescope system, showing substantial improvements in control performance.

5.2.1 Simulated System

The period length of the periodic error in telescopes is relatively long. To allow rapid prototyping, we designed a simulation system with dynamics similar to a real telescope. Experiments have shown that the model can be simplified by considering only the angular pointing error as state, measured relative to the desired state. The pointing error can be influenced by an input velocity. The resulting model is

$$\dot{x}(t) = u(t) + g(t), \tag{5.2}$$

with an unknown function $g(t)$ that is observed to be quasiperiodic.

Figure 32 shows simulation results that empirically confirm the intuition from Section 5.1 that the benefit of periodic prediction in control depends on the sampling rate. Using the numerical simulation, we compare, for various sampling rates of the state,

- an MPC controller using the linear model (4.1) with $g(t) = 0 \quad \forall t$
- two MPC controllers, both using a nonparametric, but fully stationary (i.e. not periodic) GP model for g with the square exponential covariance function (4.2); one of these models uses a length scale smaller than the periodicity (i. e. it can extrapolate periodic swings locally, but not beyond one period); the other a length scale longer than the periodicity (i. e. it averages over the periodic variations)
- two MPC controllers using instances of the periodic model for g; one in which the hyperparameters are fixed to a good value a priori (amounting to the assumption that the period of g is known), the other using the full setup described in Section 4.2, in which the periodicity hyperparameter is learned by type-II maximum likelihood during identification

Since we are only interested in the performance in the limit in this experiment, all the controllers were run for an identification phase of 10 period lengths to avoid artifacts from identification. Figure 32 shows RMS error, i.e. deviation of the state from the origin, as a function of the sampling time. The RMS error is measured over 10 period lengths, starting after the identification phase. The discretization time for the MPC is always set to 1/100 of the period length, i. e. to 1 s. Between measurements, the MPC controllers are operated in open-loop mode, i.e. the control actions are obtained from the sequence of the last MPC optimization.

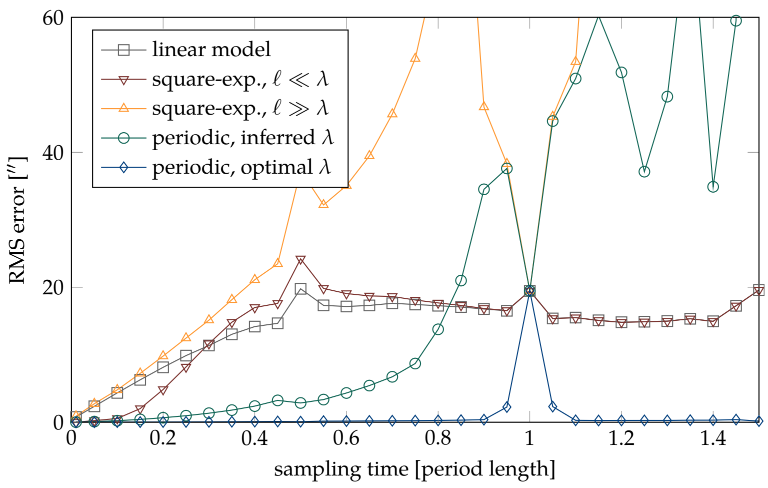


Figure 32: Comparison of the RMS error at different sampling times (in simulation), for five different prediction models: linear (—□—), square exponential with $\ell \ll \lambda$ (—▽—), square exponential with $\ell \gg \lambda$ (—△—), periodic with inferred λ (—○—) and periodic with optimal λ (—◇—). MPC control inputs are computed at the indicated sampling times, shown as a fraction of the period length. The control rate is 1 Hz for every plot. The MPC parameters were set to $Q = 10^2$ and $R = 10^1$. The horizon length was chosen such that the horizon covers the time until the next measurement. Between measurements, the MPCs are operated in open-loop mode. $g(t)$ is set to be a sine with fixed period $\lambda = 100\,\mathrm{s}$.

The results demonstrate the intuition: For sampling times much smaller than λ, the dynamics are locally linear, and all models achieve an error close to zero. Their performance difference is only marginal (lower left in Figure 32). For sampling times between about 10 % and 80 % of λ, the periodic model offers considerable benefits. When the sampling times are close to, or larger than, the periodicity, the Nyquist rate imposes limits on identifiability of the system, which adversely affects the performance of the periodic nonparametric model. This shows that a broad prior can lead to bad performance if only little data is available. On the other hand, if λ is known precisely, very good control is possible even for sampling rates lower than λ. The parameter-inferring model (—○—) in Figure 32 represents the performance of a system almost ignorant of λ in the beginning: the prior is very broad. One can expect prior information about λ of varying vagueness to give performance somewhere in the region between the parameter-inferring model (—○—) and the optimal-parameter model (—◇—) in Figure 32.

The case where sampling rate and λ are equal is special, since then g appears to be constant in the measurements, and even the informed periodic model can not learn the behavior of g. A weaker version of this effect is also visible in the plot at a sampling rate of $\lambda/2$. This "selection bias" affects all regression models, including the aperiodic ones.

5.2.2 Evaluation of Fault Tolerance

A similar experiment was conducted to investigate the effect of missing measurements on the performance of the controlled system. Figure 33 shows the empirical results. The setup is the same as in the experiment described before, but now the sampling time is fixed to a value of 5 s, while the period length is 100 s. The length of the horizon is set to 41 time steps (i. e. 205 s), covering more than two full periods of the periodic effect, which is a realistic setting for a real telescope.

After giving the system enough time to learn under regular output measurements, no new data is used to update the GP model of $g(t)$, and no new state estimate is available to the controller. This corresponds to a fault in the sensor (or clouds in front of the camera in the telescope setting). The sensor does not recover within the simulation time. Figure 33 shows the performance of the different controllers, measured in terms of the RMS error since the beginning of the fault, here at time 0. In all controller setups, the MPC control sequence can not be updated as there is no state estimate available. For the following time steps without measurement, the MPC therefore is operated in open-loop mode, i. e. the control actions from the control sequence computed at the time before the fault are used.

At the beginning of the failure (lower left of Figure 33), the performance of all methods is good, since the effect of the periodic disturbance is still small. Over time, we see the simple controller without prediction (—□—) slowly degrading. Interestingly, the performance of the GP with square exponential kernel with too long a length scale

(—△—) performs even worse than not predicting $g(t)$ at all. This illustrates how critical the choice of the hyperparameters is and that a wrong choice can even degrade the performance. The model with sensible length scale (—▽—) performs significantly better initially, but the extrapolation of the SE kernel degrades quickly (see also Figure 26 (a)) resulting in an overall performance that is only slightly better than for the linear model.

With periodic predictions, in contrast, the controllers perform significantly better during the fault. The periodic GP (—⊖—) is better than the SE with short length scale, even if the period length is inferred and not fixed at a good value. The RMS error for the GP with optimal parameter λ (—◇—) is virtually zero and even a controller having access to the true function $g(t)$ would therefore only show marginal improvement. This analysis shows that the proposed combination of a periodic GP model and MPC control is able to compensate temporary sensor failures and maintain high control performance for locally periodic dynamic effects.

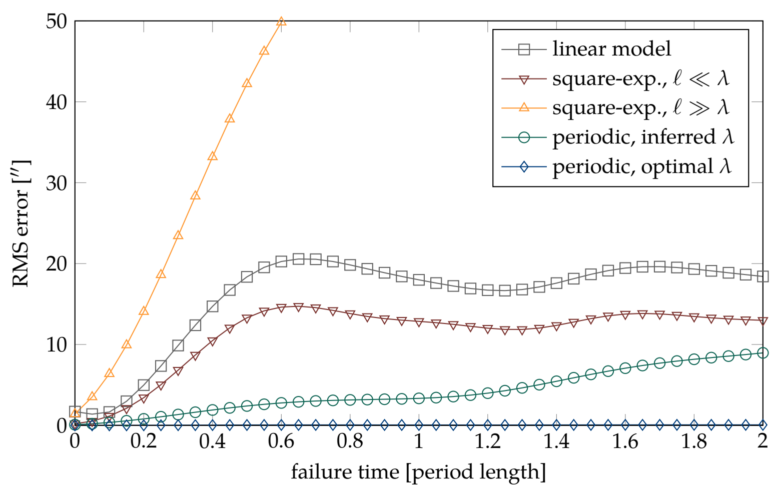


Figure 33: Comparison of the RMS error after sensor failure (in simulation), for five different prediction models: linear (—□—), square exponential with $\ell \ll \lambda$ (—▽—), square exponential with $\ell \gg \lambda$ (—△—), periodic with inferred λ (—⊖—) and periodic with optimal λ (—◇—). Sampling time and discretization time of the MPC are both 5 s. The MPC parameters are set to $T = 41$, $Q = 10^2$ and $R = 10^1$. After a training period, no new measurements are available ("dark phase"). The MPC is operated in open-loop mode. $g(t)$ is set to be a sine with fixed period $\lambda = 100$ s.

5.2.3 Hardware Experiment

We have tested our implementation on a physical system, a commercially available *Vixen* Sphinx telescope mount (Figure 34). Without closed-loop control, this mount shows about $7''$ of RMS error after correction for static drift. The error arises from the imperfect shape of the cogs in the gear of this mount (Figure 35). The imperfect shape is not visible to the naked eye, see Figure 37 (a) for an uncontrolled measurement.

Because outdoor measurements are subject to random, time-varying effects like weather conditions, we constructed a more reproducible experimental setup using a second, high-precision gearless *ASA* DDM60

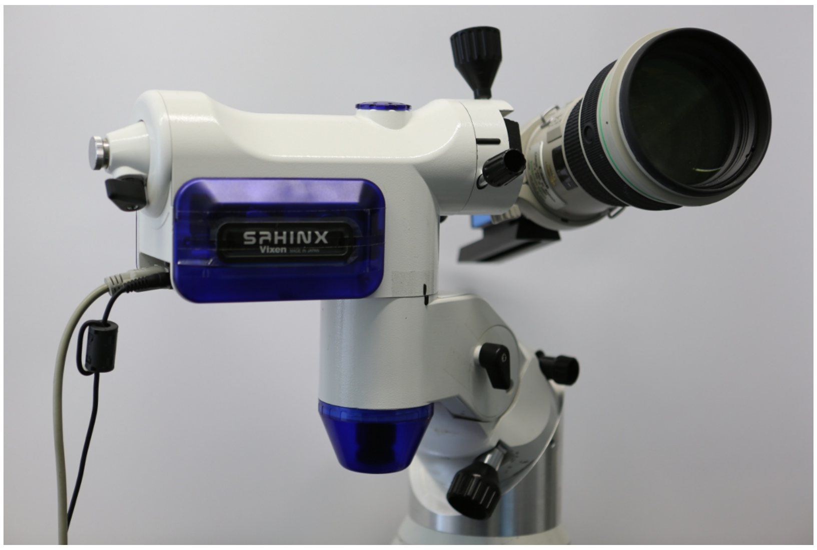


Figure 34: The telescope mount used for the tracking experiments. On the right side is the camera lens used as guiding telescope. A main telescope is not used for the tests.

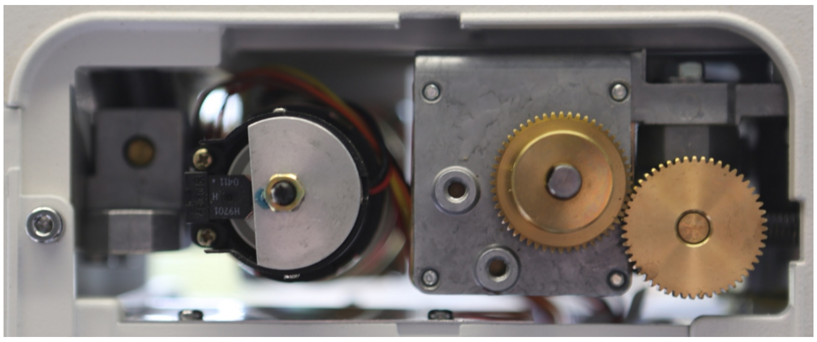


Figure 35: The gearbox of the telescope mount. One of the motors is visible on the left. The two cogs on the right transmit the motor's rotation to a worm gear (not visible), which sits on the same axis and thus has the same period. These cogs and the worm gear are the likely source of the periodic error.

Pro telescope mount equipped with a laser "star" as tracking reference (Figure 36). It has a typical tracking RMS error of about 0.4″. The measurement is done with a *Canon* EF400DO lens on a *The Imaging Source* DMK 41AU02.AS camera.

For the hardware interaction, the open source "PHD Guiding" software package is used. In the original implementation this software uses a proportional controller with hysteresis to prevent direction switching. The telescope is connected to the computer with a *Shoestring Astronomy* GPUSB, a device that sends pulse-width modulated signals to telescopes over a commonly used 6-wire interface.

We altered the software to gain access to the measured displacement of the camera image. The value is sent through a network socket to MATLAB, with which the proposed controller was implemented to calculate the optimal control signal. The control signal is sent back to the guiding software, which then passes it on to the telescope hardware. For plotting and calculation of the RMS error, the measured displacement is converted from pixels into arcseconds (″) with an empirically determined conversion factor[2].

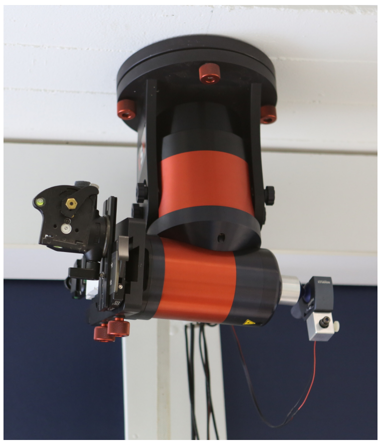


Figure 36: The gearless *ASA* DDM60 Pro telescope mount, attached to the ceiling of our laboratory.

[2] The pixel resolution of our lens and camera combination was determined with `nova.astrometry.net` [58].

[58] Hogg et al., "Automated Astrometry," 2008

For real-time implementation, algorithmic complexity is relevant. The computational cost of the GP prediction scales cubically in the number of data points. To bound computational cost, we limit the number of used data points to 90 in a moving window fashion. This gives a sufficient coverage of 270 s, or about 3 periods of the short periodic component. Since inference continuously runs in an extrapolation setting (see also Figure 26 (b)), this is sufficient for precise inference and control.

For the prediction of the dynamics in the MPC framework, an ODE-solver is employed to predict the tracking reference from the mean of the GP prediction. This has manageable computational cost because the inference of a Gaussian process is dominated by the initial one-time operation of inverting the Gram matrix K in $\mathcal{O}(N^3)$ time, while evaluating the mean function at M times only has cost $\mathcal{O}(MN)$.

The optimization of the hyperparameters is also an expensive part of this algorithm. As the kernel Gram matrix has to be filled and inverted at every evaluation of the objective function, the number of evaluations has to be kept small at every sampling time. We use a numerical optimizer based on the BFGS update [23, 45, 50, 128], a quasi-Newton algorithm that updates an estimate of the inverse Hessian in each iteration. In the standard implementation, the estimate of the Hessian obtained at one time step is discarded after each individual call to the optimizer. For the use in the control setting, we altered the algorithm so that the estimate of the inverse Hessian is stored and used to initialize the optimizer's estimate at the next sampling time. This makes it possible to do one iteration per sampling interval (in a sense, threading the optimization algorithm into the learning algorithm). This significantly reduces computation time.

[23] Broyden, "A New Double-rank Minimization Algorithm," 1969

[45] Fletcher, "A New Approach to Variable Metric Algorithms," 1970

[50] Goldfarb, "A Family of Variable Metric Updates Derived by Variational Means," 1970

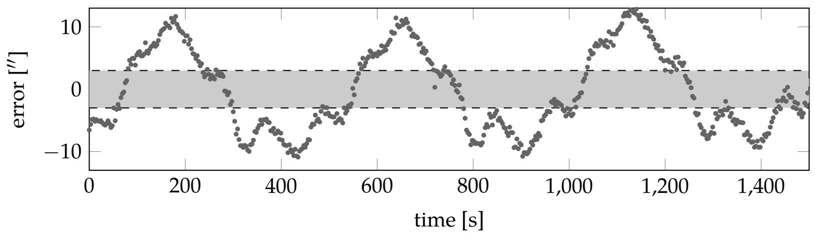
[128] Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization," 1970

The presented method was tested on two different setups, one with an MPC based on the linear model only and one with the GP prediction for the periodic error g(t). The test was run 3 times for 25 min each. Both the sampling and the discretization time were set to 3 s. The horizon length was $T = 10$; the state and control weights were set to $Q = 10^2$ and $R = 10^3$. The results of these runs are shown in Table 1.

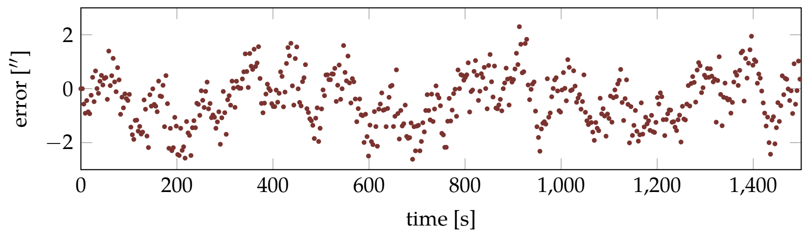
	Run 1	Run 2	Run 3	**Mean**
Plain MPC	0.9839	1.0234	0.9353	**0.9809**
GP-MPC	0.7365	0.7792	0.7605	**0.7587**

Table 1: Experimental results from indoor telescope tracking (RMS error, $''$).

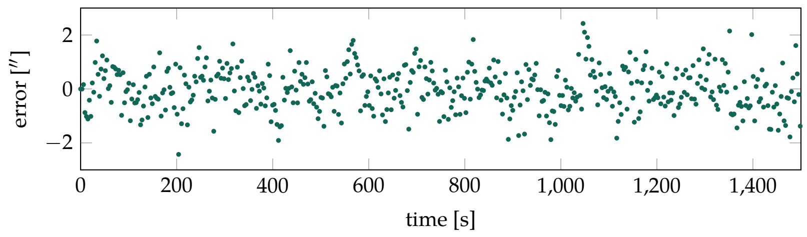
The RMS error drops by 22.64 % through the use of GP predictions in this hardware setup. Baseline measurements without movement showed about 0.25 to 0.35$''$ of noise. The noise introduced by the stepper motor could not be quantified with the current measurement system. Overall, the presented method eliminated at least a third of the controllable error, after subtracting baseline noise but without taking the stepper motor into account. This is a good result in this domain, but could probably be further improved, which is also visible through the weak residual structure visible in Figure 37 (c).



(a) Uncontrolled, RMS(e) = 6.533″



(b) Plain MPC, RMS(e) = 1.023″



(c) GP-MPC, RMS(e) = 0.779″

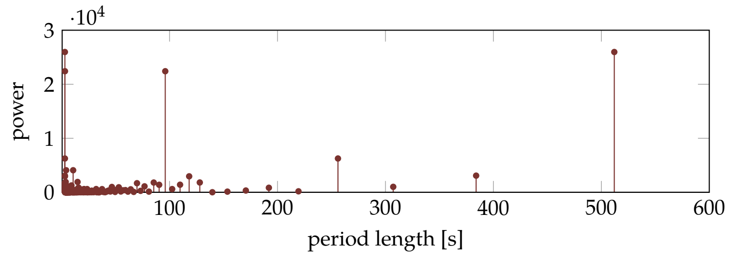
Figure 37: State measurements from the indoor tracking experiment: Without controller (•••), where the highlighted area (▨) shows the vertical range of the two other plots; for plain MPC using only a linear model (•••); and for the periodic Gaussian process based MPC (•••). The controlled measurements' data are from run 2 of our experiments, which resulted in the highest (worst) RMS error for both models (Table 1).

Figure 38 shows the power spectra of the measurements, obtained from the Fourier transform. It is noticeable that the strong periodic components near 100 s and near 500 s, as well as the constant component are highly damped with the presented method.

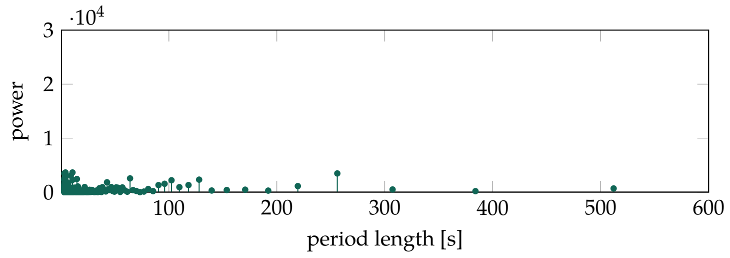
5.3 Conclusion

Telescope tracking requires high precision with low sampling rates. Therefore, Gaussian process regression with a quasiperiodic prior is a well-suited prediction framework for this application.

Numerical and hardware experiments confirm the intuitive result that the benefit of periodic models depends on the relative size of state sampling and disturbance frequencies. We showed that, even in cases where the gain of a periodic prediction is only marginal during normal operation, these models are beneficial when sensors fail temporarily. The presented method also shows considerable increases in control performance, confirming the practical utility of this framework.



(a) Plain MPC



(b) GP-MPC

Figure 38: Power spectra of the measurements of Figure 37, plain MPC using only a linear model (●), and periodic Gaussian process model based MPC (●).

6 Software Implementation: PHD2 Guiding

THE METHOD described in Chapters 4 and 5 was implemented as part of the telescope guiding software *PHD2 Guiding* [132]. PHD2 Guiding is an open-source telescope guiding software used by many amateur astronomers. Developing the adaptive periodic error correction algorithm as a part of PHD2 Guiding makes it possible to reach the astronomers that already use this software. Also, by using the mature and well-engineered PHD2 Guiding platform, we do not need to take care of hardware interactions and can instead focus on the guiding algorithm itself.

[132] Stark, McKee, Galasso, et al., *PHD2 Guiding*, 2016

After introducing the PHD2 Guiding framework (Section 6.1), we describe several modifications to the algorithm (Section 6.2). These changes were necessary to increase robustness for the use by unexperienced users. We then briefly present a guiding method for the second telescope axis (Section 6.3) before discussing the results of experiments with this predictive telescope guiding framework (Section 6.4).

6.1 The PHD2 Guiding Framework

PHD2 Guiding is a GUI-based software for telescope guiding optimized for easy usage. Figure 39 shows the main window of the software with star display and residual error curves. The software includes drivers for many guiding cameras as well as telescope mounts, and handles the hardware interactions. PHD2 Guiding is open-source software, so it is possible to integrate additional features. It is under active development and is downloaded several thousand times per month.

The current state-of-the-art guiding algorithm in PHD2 Guiding is a proportional controller with hysteresis to prevent changes in the control direction. This algorithm does not make use of extrapolation. The lack of a predictive control scheme results in residual structure in the pointing error when the sampling frequency is low. It is also not robust, e. g., when a cloud occludes the guide star for some time. In such a case the guide star is usually lost because it moved too far away from its original position during the period without measurements.

These two issues can be tackled with the GP framework: Structured extrapolation together with predictive control reduces residual error and residual structure. Also, since extrapolation and predictive control still offer a control input even when no measurements are available, it is possible to issue control actions in those cases. This results not only in a decrease in pointing error, but also increases robustness because the guide star remains closer to the target position.

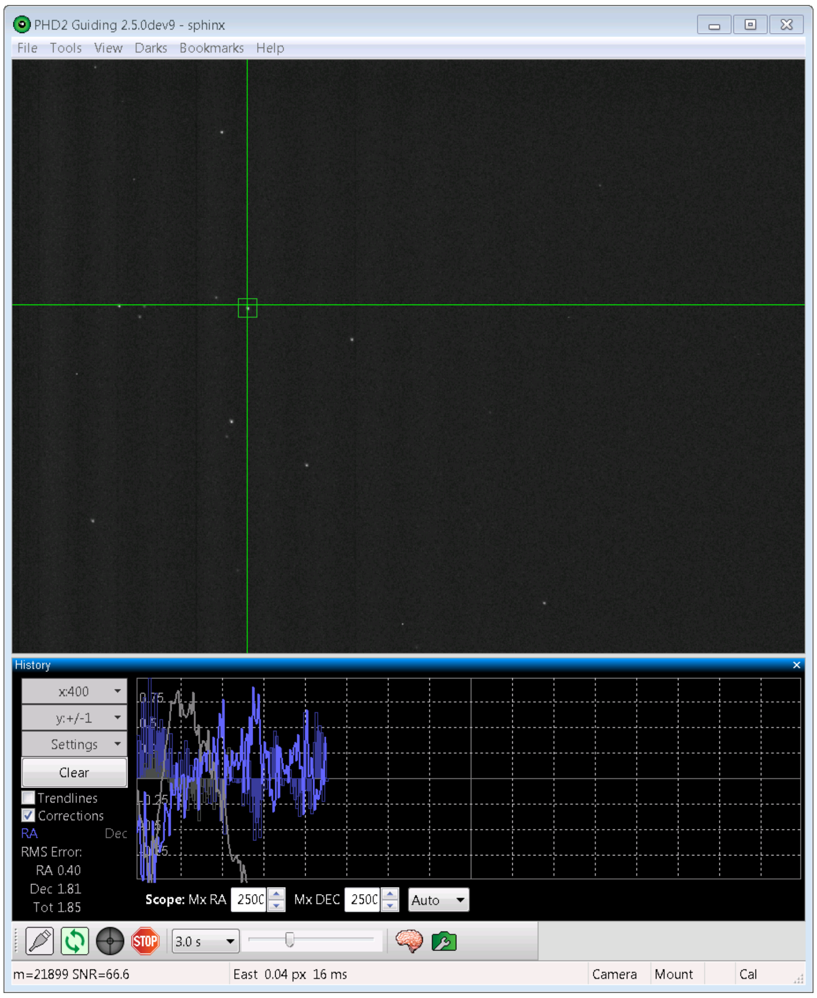


Figure 39: GUI of PHD2 Guiding in version 2.5.0. The upper part is the star display with the guide star highlighted by the crosshair. The search region for the guide star is the small box. The lower part shows the residual error curves and the controls.

6.2 Periodic Error Correction for PHD2 Guiding

Despite the fact that there are libraries for Gaussian processes in C++, we chose to implement the GP algorithm ourselves. This decision was made because some crucial features, such as the conditioning on a subset of the kernel combination, are missing in the readily available libraries.

Much of the GP infrastructure is based on linear algebra. Thus, we selected to use the Eigen matrix library [53] to make use of fast and well-tested linear algebra code. Using the Eigen library also increases the readability of the overall implementation. The Eigen library itself

[53] Guennebaud, Jacob, et al., *Eigen v3*, 2010

has no external dependencies and is therefore relatively easy to include in the existing project.

6.2.1 Data Acquisition

Instead of retaining all collected data, it is necessary to limit the number of data points to keep the computational effort bounded and the runtime predictable. We use a FIFO buffer to store data points, where, once the buffer is full, the oldest entry is dropped whenever a new data point is added. This amounts to a windowing of the dataset.

Using the measured pointing error directly to identify the dynamics function turned out to be not robust enough during our intensive testing. Therefore, we model the *accumulated* gear error

$$a_k = e_k + \sum_{i=0}^{k-1} u_i \tag{6.1}$$

where e_k is the measured pointing error at time step k and u_i are the control signals at time i. This model is more robust because the error introduced by the gear accumulates, while random effects caused by, e. g., the stepper motor, average out. Overall, this leads to a better signal-to-noise ratio for the accumulated gear error compared to the absolute pointing error.

Additionally, the measurement process does not have uniform noise. Since the location of the guide star is determined from pixel intensities via a weighted centroid of the star's pixels, the accuracy of the centroid calculation depends on the signal-to-noise ratio (SNR) of the image. In PHD2 Guiding, the SNR is calculated according to the method described by Simonetti [129]; the relation between the SNR and the measurement noise variance was determined empirically and modeled by the linear regression model

[129] Simonetti, *Measuring the Signal to Noise Ratio of the CCD Image of a Star or Nebula*, 2004

$$\sigma(\rho) = w_{\sigma,1}(\rho - \rho_0)^{-1} + w_{\sigma,0}, \tag{6.2}$$

where ρ is the SNR, ρ_0 is an input offset and w_σ are the parameters for the regression model. Figure 40 shows this relationship and the data used to determine it.

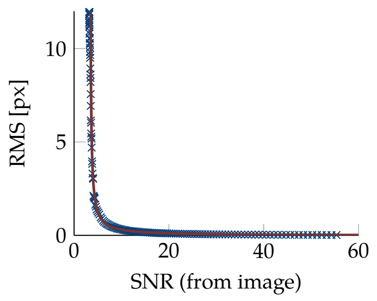


Figure 40: Relationship between the signal-to-noise ratio and measurement variance. Shown are RMS errors from simulated exposures for different SNRs (×) and the regression line (——).

The value of the estimated noise variance is stored along with the measurements of the signal. It is then used in a heteroscedastic noise model, as described in Section 1.7.1.

6.2.2 Modeling

In real astronomical imaging setups (see Figure 41 for an example), it is almost impossible to align the axis of the telescope mount with the

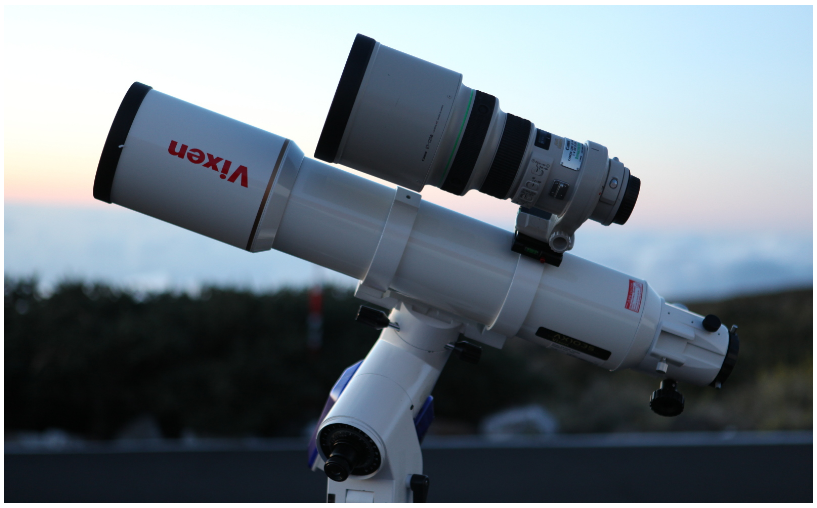


Figure 41: Telescope setup with main and guiding telescope. The guiding telescope is mounted "piggyback" on the main imaging telescope. The cameras are not yet attached.

Earth's rotational axis perfectly. Thus, in most cases there will be a moderate drift that needs to be modeled and predicted in addition to the periodic component. As it is hard to find the linear drift component independently of the periodic gear error, we employ a joint model as described in Section 1.7.2, consisting of fixed features for constant and linear components and a Gaussian process to model the nonlinear part:

$$a(t) = [\beta_0 \quad \beta_1] \begin{bmatrix} 1 \\ t \end{bmatrix} + f(t) \tag{6.3}$$

with $f(t) \sim \mathcal{GP}$. We assume an uninformative prior where $\beta \sim \mathcal{N}(0, B)$, $B^{-1} \rightarrow 0$, as detailed in [115, §2.7].

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

The GP itself consists of three different components that were identified during experimentation. It is also possible to automatically identify kernel components [39], but for this application the hand-crafted kernels were more compact and robust. The different components are:

[39] Duvenaud et al., "Structure Discovery in Nonparametric Regression through Compositional Kernel Search," 2013

slowly-varying SE Not all mechanical effects are periodic. In order to allow for mechanical deviations, e. g., deformations due to load shift in the mechanics, we include a long-length-scale square exponential kernel. As a result of the long length scale, this part of the model is also amenable to extrapolation.

$$k_{\text{SE},0}(t, t'; \theta_0, \ell_0) = \theta_0^2 \exp\left(-\frac{|t-t'|^2}{2\ell_0^2}\right) \tag{6.4}$$

periodic The periodic component is the important part of the model. It captures the recurrent disturbances from the gear. This component was chosen to be periodic to enable long-term predictions.

$$k_{\mathrm{P}}(t, t'; \theta_{\mathrm{P}}, \ell_{\mathrm{P}}, \lambda) = \theta_{\mathrm{P}}^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi}{\lambda}(t - t')\right)}{\ell_{\mathrm{P}}^2}\right) \tag{6.5}$$

quickly-varying SE Since the actual data is not perfectly periodic, we allow for fast variations with a square exponential kernel with short length-scale. This kernel captures deviations due to mechanical roughness.

$$k_{\mathrm{SE},1}(t, t'; \theta_1, \ell_1) = \theta_1^2 \exp\left(-\frac{|t - t'|^2}{2\ell_1^2}\right) \tag{6.6}$$

The combined kernel function is the sum of the three individual kernel functions

$$k_{\mathrm{C}}(t, t'; \boldsymbol{\eta}) = k_{\mathrm{SE},0}(t, t'; \theta_0, \ell_0) + k_{\mathrm{P}}(t, t'; \theta_{\mathrm{P}}, \ell_{\mathrm{P}}, \lambda) + k_{\mathrm{SE},1}(t, t'; \theta_1, \ell_1), \tag{6.7}$$

where we have subsumed parameter dependencies into the parameter vector $\boldsymbol{\eta} = [\theta_0, \ell_0, \theta_{\mathrm{P}}, \ell_{\mathrm{P}}, \lambda, \theta_1, \ell_1]$. A sample from the compound model, combining the linear part and the GP with kernel function (6.7) is shown in Figure 42.

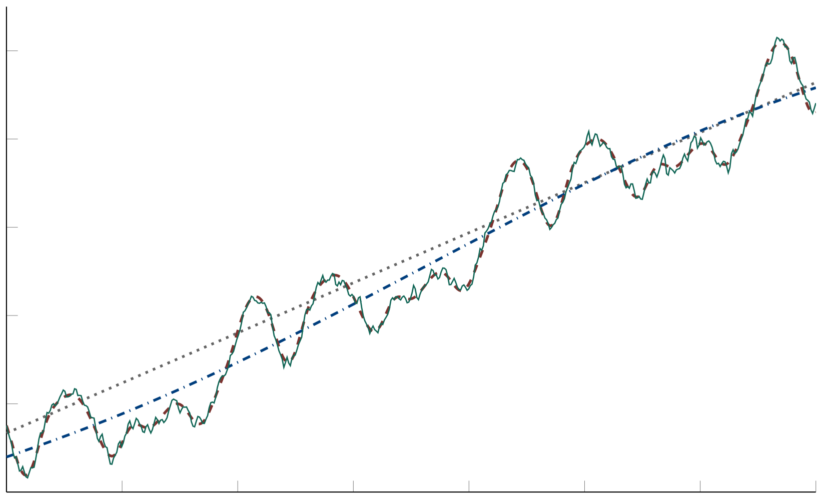


Figure 42: Combining the components of the model: Linear drift (⋯⋯), plus long length-scale SE (-·-·-), plus periodic (- - -), plus short length-scale SE (——). One sample of the model is generated from the individual components.

The overall model is expressive enough to capture the different aspects of the problem. While linear drift, slowly-varying SE and the periodic component are features that can be extrapolated, the quickly-varying SE can not. It regresses back to the mean quickly after the last data point, which disturbs the controller. Therefore, while inference is done with the full model, the predictions are conditioned on the model without the quickly-varying SE. See Section 1.4.3 for the details on the output conditioning.

6.2.3 Hyperparameter Identification

While many of the hyperparameters included in the model can be estimated by mechanical considerations without affecting performance too much, one parameter can not: the period length λ. It is crucial for the predictive power that this parameter closely matches the mechanical period length; there is little robustness regarding the choice of this parameter.

Instead of relying on the maximization of a probabilistic criterion, the identification of the main periodic component is done with classic spectrum analysis. After subtraction of a polynomial fit of order 1 to remove the linear trend, a Hamming window [109, §7.2.1] is applied to reduce spectral leakage effects due to the finite-length signal.

[109] Oppenheim, Schafer, and Buck, *Discrete-time Signal Processing*, 1999

In order to obtain the spectral density of the measured signal, the discrete Fourier transform (DFT) [109, §8.1] is calculated on the preprocessed data

$$s_k = \sum_{n=0}^{N-1} a_n e^{-\frac{2\pi\imath}{N}nk}, \tag{6.8}$$

where $\mathbf{s} = [s_0, \ldots, s_{N-1}]$ is the spectrum, $\mathbf{a} = [a_0, \ldots, a_{N-1}]$ is the data and $\imath$ is the imaginary unit. The period length of the strongest frequency component is subsequently calculated from the power spectrum $\mathbf{s}$. In order to ensure constant frequency resolution, the data vector is zero-padded up to the maximum number of data points in the FIFO buffer.

In the software implementation, we use the fast Fourier transform (FFT) [47, 28] for the spectrum analysis, since the naïve implementation of (6.8) is not viable for the online use.

[47] Gauss, "Nachlass: Theoria Interpolationis Methodo Nova Tractata," 1866

[28] Cooley and Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," 1965

6.2.4 GP Approximation

It is important to note that in the case of periodic covariance functions, it is not enough to do a simple windowing because it is necessary to cover multiple periods with the data for decent predictive performance. Depending on the choice of sampling frequency, it is often too expensive to use a full dataset of consecutive measurements to cover multiple periods. Instead, the FIFO buffer holds multiple periods worth of measurements and we use a subset-of-data (SD) approximation to keep inference cheap. See Section 1.6.1 for the details of the SD method.

The fact that the extrapolation position is known in advance is beneficial in this case. In order to obtain a computationally cheap method, we use the covariance between the prediction point and the data to select the data points for the approximation:

$$\mathbf{x}_{\text{SD}} = \texttt{sort}\,\{\mathbf{x}; k(\mathbf{x}, x)\}_{0:M-1}\,, \tag{6.9}$$

where $\mathbf{x}_{\text{SD}}$ is the selected dataset, `sort` sorts the data $\mathbf{x}$ according to the values of the covariance vector $k(\mathbf{x}, x)$ and $\{\cdot\}_{0:M-1}$ selects the first M elements.

For each control step, we select a new subset from the large dataset in the FIFO buffer, depending on the current prediction point. This way we make sure that there is always relevant data from past periods available to the GP, while keeping the inference cost low.

6.2.5 Control

The control algorithm used in the guiding software was designed to avoid the problems that arise from the low sampling frequency, while keeping the complexity of the algorithm low. Ideally, one would implement an optimal control scheme as introduced in Section 4.3, but for the simplicity of the algorithm, only a "deadbeat" controller[1] was considered. Due to the low sampling frequency and non-uniform sampling intervals, this deadbeat control often led to overshoots and instabilities. Reducing the control gain, on the other hand, led to performance loss due to the periodic error.

[1] A deadbeat controller is a predictive controller that is tuned to eliminate the entire error in one step, i. e. the error-feedback gain is 1.

The implemented algorithm, therefore, is a hybrid controller: Deadbeat control is applied for the compensation of the periodic error, and proportional control is used to drive the residual error to zero. The overall control signal at time k is given by

$$u_k = -(\tilde{a}_{k+1} - \tilde{a}_k) - g_{\text{P}} e_k, \tag{6.10}$$

where $\tilde{a}_k$ is the mean prediction of the Gaussian process modeling the accumulated gear error a_k, and g_{P} is the control gain for the proportional controller. The structure of this hybrid control algorithm is shown in Figure 43.

6.3 The Declination Axis

When the alignment between the right ascension (RA) axis and the Earth's rotational axis is not perfect, the offset introduces a linear drift in the image plane for *both* axes. Therefore, also the declination (Dec) axis needs to be controlled.

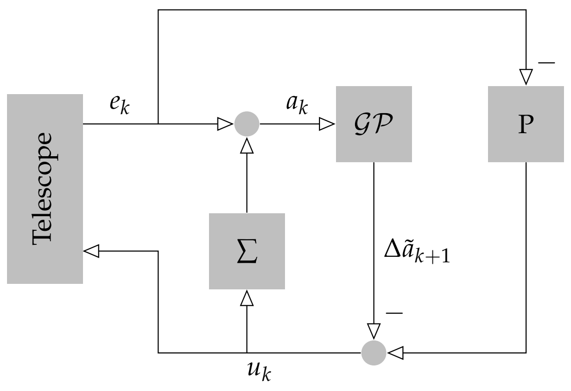


Figure 43: Structure of the telescope controller for the RA axis. While the residual error is handled with proportional control, the GP prediction is accounted for in a deadbeat fashion.

However, since the declination axis is almost still and only needs to compensate small drift errors, the regression problem is different from the RA axis: Due to the low signal-to-noise ratio and the small drift, the control signal of a proportional control often changes its sign. This leads to gear backlash effects, disrupting even usual robust regression models, such as ℓ_1-regression or Huber-regression [59].

[59] Huber, "Robust Estimation of a Location Parameter," 1964

Since the underlying dynamical model is simple, and the estimation only needs to find a linear drift, we implemented a trimmed mean estimator for the linear drift:

$$\Delta\tilde{a}^{\text{Dec}}_{k+1} = \frac{1}{k-2n_T} \sum_{i=n_T}^{k-n_T-1} \texttt{sort}\left\{\mathbf{\Delta a}^{\text{Dec}}_{1:k}\right\}_i, \tag{6.11}$$

where $\mathbf{\Delta a}^{\text{Dec}}_{1:k}$ is the collection of finite differences of the accumulated gear errors a^{Dec}_k and n_T is the number of elements to trim on either side. The `sort`-operator sorts the collection according to the values, and $\{\cdot\}_i$ selects the i-th element.

The control structure, which is similar to (6.10), is shown in Figure 44. The estimated drift $\Delta\tilde{a}^{\text{Dec}}_{k+1}$ is corrected for in a deadbeat fashion, while the error term is driven to zero with a conservatively-tuned PD controller [8].

[8] Åström and Hägglund, *PID Controllers: Theory, Design and Tuning*, 1995

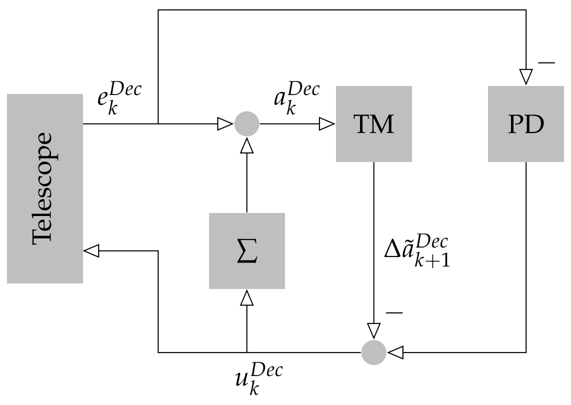


Figure 44: Structure of the telescope controller for the Dec axis. While the residual error is handled with PD control, the prediction from a trimmed-mean estimator is accounted for in a deadbeat fashion.

6.4 Experiments

The demonstration of the implemented guiding algorithm was done for the use-case of tracking stars in the night sky (as opposed to the laboratory setup of Section 5.2.3). The measurements were made in Tübingen, Germany, using a *Vixen* Sphinx telescope mount with a *Canon* EF 400 DO IS USM lens and a *The Imaging Source* DMK 41AU02.AS camera, the same equipment as in the laboratory setup.

6.4.1 Guiding Performance

The performance of GP guiding is shown in comparison to the state-of-the-art algorithm[2] that is used in PHD2 guiding. Figure 45 shows the exemplary comparison of the two algorithms under otherwise identical conditions. The angular pointing error between target position of a guide star and its measured position is plotted over time. It is visible that the GP guiding shows less residual structure after the initial learning phase (up to around 1000 s). Since the GP guider starts with zero information, the effect of learning is visible in the data by the structural change in the error residuals after 1000 s.

[2] In PHD2 Guiding, the standard algorithm is "Hysteresis", which is a P-controller with hysteresis to prevent gear backlash effects.

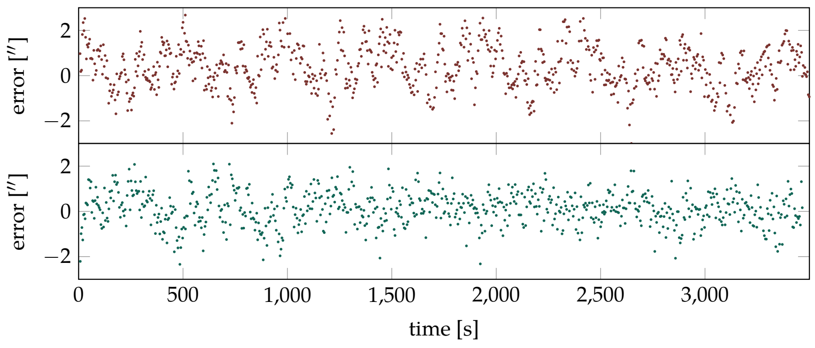


Figure 45: Comparison between state-of-the-art tracking of PHD2 Guiding 2.5.0 (top, ···) and GP-based tracking (bottom, ···).

When analyzing the guiding performance, the initial learning phase has to be compared separately. We report the mean, standard deviation and RMS error of the two algorithms for the different phases. The GP guiding consistently outperforms the standard algorithm.

	overall			for $t < 1000$			for $t > 1000$		
	mean	SD	RMS	mean	SD	RMS	mean	SD	RMS
Hy	0.357	0.968	1.031	0.427	0.917	1.010	0.331	0.985	1.039
GP	0.122	0.751	0.761	0.088	0.893	0.895	0.136	0.685	0.697

Table 2: Experimental results from outdoor telescope tracking (all numbers in "). "Hy" is the standard "Hysteresis" algorithm and "GP" is the GP guiding algorithm developed in this thesis. In addition to the overall performance, we report the performance for the initial phase ($t < 1000$) and for the later phase ($t > 1000$) separately.

6.4.2 Dark Guiding

In order to show the improved robustness under conditions where the measurement is unavailable for some time, we implemented a dark guiding mode. When there is no measurement, the guiding is done with the GP predictions alone. This prevents the telescope from moving away from the guide star too quickly, increasing chances of finding the guide star after recovery of the measurement process.

Figure 46 shows the benefit of the GP guider under occlusion conditions. With the GP guider the linear and periodic motion is predicted, keeping the pointing error comparably small. The state-of-the-art algorithm does not predict the motion of the telescope, leading to larger pointing errors after an occlusion period.

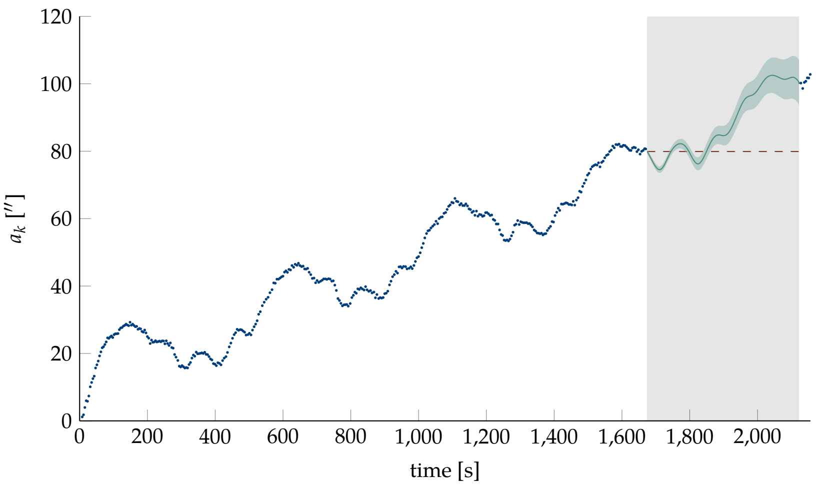


Figure 46: Tracking under cloudy conditions. After enough time for learning, the measurements were dropped for a few minutes, simulating the effect of a cloud covering the guide star. The plot shows the accumulated gear error a_k from measurements (···), and the predictive mean (——) and uncertainty (🟩) for the time without measurement (indicated by ▨). The "prediction" of the state-of-the-art algorithm is flat (– – –).

6.5 Conclusion

Making the Gaussian process based periodic error correction algorithm available to many practitioners in astrophotography was an important goal of this project. In order to make the algorithm work on a broad range of telescopes, a number of changes to the algorithm were developed and implemented, such as a hybrid control strategy and the parameter identification via FFT.

In experiments under realistic conditions we showed that the implemented algorithm provides improved tracking performance and can even deal with the case of an unreliable measurement process due to clouds.

Part III

Nonlinear Dual Control

7 Introduction to Dual Control

The idea of performing simultaneous identification and control by applying optimal control to both states and parameters of uncertain dynamical systems is today also known as *Bayesian reinforcement learning* [112] in the machine learning community. Originally it was termed *dual control* by Fel'dbaum [42], who already noted that dual control algorithms will be computationally expensive due to the necessary numerical integration and optimization. Both algorithmic and—by the standards of the time—computational complexity have limited its widespread application.

[112] Poupart et al., "An Analytic Solution to Discrete Bayesian Reinforcement Learning," 2006

[42] Fel'dbaum, "Dual Control Theory I-IV," 1960–1961

While adaptive control only considers past observations, dual control also takes future observations into account. This approach mitigates a number of drawbacks of other techniques for dealing with uncertain parameters. Robust controllers [152], for example, may limit performance due to their worst-case design; adaptive controllers based on *certainty equivalence* [76, §11.2] can lead to failure in cases of high uncertainty because the uncertainty of the parameters is not taken into account but only their mean estimates. One approach to incorporate the uncertainty is stochastic optimal control [6], where the uncertain parameters are marginalized out while calculating the optimal controller. This leads to smaller control signals when facing uncertainty ("cautious control"), but it can also result in the so-called "turn-off phenomenon" [60] if the uncertainties are large: The control is scaled down towards zero, and, as a result, the system never acts or learns.

[152] Zhou and Doyle, *Essentials of Robust Control*, 1998

[76] Kumar and Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*, 1986

[6] Åström, *Introduction to Stochastic Control Theory*, 1970

[60] Hughes and Jacobs, "Turn-off, Escape and Probing in Nonlinear Stochastic Control," 1974

For many systems, more excitation leads to better estimation, but also to worse control performance. Dual control attempts to find a compromise between exploration and exploitation by taking the future effect of current actions into account. In episodic settings, where the control problem is re-instantiated repeatedly with unchanged system dynamics, comparably simple notions of exploration can succeed. For example, assigning an *exploration bonus* to uncertain options [149], or acting optimally under one sample from the current probabilistic model of the environment [26], can perform well [32, 72, 131]. Such approaches, however, do not model the effect of actions on future beliefs, which limits the potential for the balancing of exploration and exploitation. This issue is most drastic in the non-episodic case, the control of a single trial. Here, the controller can not hope to start over, and exploration must be carefully controlled to avoid disaster.

[149] Wittenmark, "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters," 1975

[26] Chapelle and Li, "An Empirical Evaluation of Thompson Sampling," 2011

[32] Dearden, Friedman, and Andre, "Model Based Bayesian Exploration," 1999

[72] Kolter and Ng, "Near-Bayesian Exploration in Polynomial Time," 2009

[131] Srinivas et al., "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," 2010

A principled solution to this problem is offered by the dual control framework: A probabilistic belief over the dynamics and the environment can be used not just to simulate and plan trajectories, but also to reason about changes to the belief from future observations and their influence on future decisions. An elegant formulation is to combine the physical state with the parameters of the probabilistic model into an augmented dynamical description, which is then controlled jointly. Due to the inference, the augmented system invariably has nonlinear and uncertain dynamics, causing the optimal controller to have prohibitive computational cost—even for finite state spaces and discrete time [112], all the more for continuous space and time [56].

[112] Poupart et al., "An Analytic Solution to Discrete Bayesian Reinforcement Learning," 2006

[56] Hennig, "Optimal Reinforcement Learning for Gaussian Systems," 2011

When the learning as response to current actions is taken into account, the turn-off characteristic vanishes in favor of explorative behavior. [42] It was shown that this kind of problem is intractable [4, §III.3], except for a few comparably simple systems, e. g., [135]. Therefore, a variety of approximate formulations of the dual control problem have been developed. This includes the introduction of perturbation signals [63], exploration bonuses [149], series expansion of the loss function [142] or modifications of the loss function [43, §4]. A comprehensive overview of dual control methods is given by Wittenmark [148]. A historical side-effect of these numerous treatments is that the meaning of the term "dual control" has evolved over time, and is now applied both to the fundamental concept of optimal exploration, and to methods that only approximate this notion to varying degree.

[42] Fel'dbaum, "Dual Control Theory I-IV," 1960–1961

[4] Aoki, *Optimization of Stochastic Systems*, 1967

[135] Sternby, "A Simple Dual Control Problem with an Analytical Solution," 1976

[63] Jacobs and Patchell, "Caution and Probing in Stochastic Control," 1972

[149] Wittenmark, "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters," 1975

[142] Tse, Bar-Shalom, and Meier, "Wide-sense Adaptive Dual Control for Nonlinear Stochastic Systems," 1973

[43] Filatov and Unbehauen, *Adaptive Dual Control*, 2004

[148] Wittenmark, "Adaptive Dual Control Methods: An Overview," 1995

However, many approximate methods are too simple to retain all features of dual control: *caution*, the downscaling of control signals when facing high uncertainty; *exploration*[1], the excitation of the system when cautious control does not learn fast enough; and the *value of information*, the selective exploration of system parameters that are important for future performance of the system. [15] An approximation derived by Tse, Bar-Shalom, and Meier [142, 100] is conceptually close to optimal dual control and retains all three of the aforementioned features of dual control. In the following sections we provide a brief introduction to this method.

[1] This feature is sometimes also called "investigation" or "probing" in the dual control literature.

[15] Bar-Shalom and Tse, "Caution, Probing, and the Value of Information in the Control of Uncertain Systems," 1976

[100] Meier, *Combined Optimal Control and Estimation Theory*, 1966

Dual control has regained attention over the past few years, but in many cases exploration bonuses are used and explicitly added to the control cost. Other methods include constraining the minimal information gain [78, 116] and persistent excitation [48, 94] to maintain the parameter knowledge. The approach that will be proposed in the following chapter focuses on maintaining the value of information, which can not be expressed through exploration bonuses or excitation signals, but is one of the key benefits of dual controllers.

[78] Larsson, "Application-oriented Experiment Design for Industrial Model Predictive Control," 2014

[116] Rathouský and Havlena, "MPC-Based Approximation of Dual Control by Information Maximization," 2011

[48] Genceli and Nikolaou, "New Approach to Constrained Predictive Control with Simultaneous Model Identification," 1996

[94] Marafioti, Bitmead, and Hovd, "Persistently Exciting Model Predictive Control," 2014

In this chapter, we review the algorithm for approximate dual control introduced by Tse and Bar-Shalom [141] in a mildly simplified

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

fashion: While the original algorithm is based on differential dynamic programming (DDP) [96], we view the algorithm in the light of the more recent iterative linear quadratic Gaussian (iLQG) framework [140], where applicable. We first introduce the general stochastic optimal control problem (Section 7.1), present the dual control approach and show how the algorithm works (Section 7.2). Finally, we show the effect of the dual control approach on the cost function (Section 7.3).

[96] Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," 1966

[140] Todorov and Li, "A Generalized Iterative LQG Method for Locally-optimal Feedback Control of Constrained Nonlinear Stochastic Systems," 2005

7.1 Model and Notation

In this chapter, we consider the discrete-time, finite-horizon stochastic optimal control problem of the form

$$x_{t+1} = A_t x_t + B_t u_t + \xi_t \quad \text{(state dynamics)} \tag{7.1a}$$

$$y_t = C x_t + \gamma_t \quad \text{(observation model),} \tag{7.1b}$$

with dynamics matrices $A_t \in \mathbb{R}^{n_x \times n_x}$ and $B_t \in \mathbb{R}^{n_x \times 1}$. At time $t \in \{0, \dots, T\}$, $x_t \in \mathbb{R}^{n_x}$ is the state and $\xi_t \sim \mathcal{N}(0, D)$ is a Gaussian disturbance. The control input is denoted u_t; for simplicity we assume scalar $u_t \in \mathbb{R}$ throughout. Measurements $y_t \in \mathbb{R}^{n_y}$ are observations of x_t, corrupted by Gaussian noise $\gamma_t \sim \mathcal{N}(0, W)$. The generative model thus reads $p(x_{t+1} \,|\, x_t, u_t) = \mathcal{N}(x_{t+1}; A_t x_t + B_t u_t, D)$ and $p(y_t \,|\, x_t) = \mathcal{N}(y_t; C x_t, W)$, with a linear map $C \in \mathbb{R}^{n_y \times n_x}$. Trajectories are vectors $\mathbf{x} = [x_0, \dots, x_T]$, and analogously for $\mathbf{u}, \mathbf{y}$. We occasionally use the subset notation $\mathbf{y}_{i_1:i_2} = [y_{i_1}, \dots, y_{i_2}]$.

We further assume that dynamics matrices A_t, B_t are not known, but are described by a Gaussian distribution over their elements. To simplify notation, we reshape the elements of A_t and B_t into a parameter vector $\theta_t = [\mathrm{vec}(A_t); \mathrm{vec}(B_t)] \in \mathbb{R}^{(n_p+1)n_x}$, and define the reshaping transformations $A(\theta_t) : \theta_t \mapsto A_t$ and $B(\theta_t) : \theta_t \mapsto B_t$.

The key observation in dual control is that both the states x and the parameters θ are subject to uncertainty and can therefore be subsumed in an *augmented state* $z_t^\intercal = [x_t^\intercal, \theta_t^\intercal] \in \mathbb{R}^{n_x + n_\theta}$ [42]. Even though the source of uncertainty is different for states and parameters (the states are uncertain due to stochasticity, while the parameters are uncertain due to ignorance), both can then be dealt with in the form of a joint probability density $p(z) = \mathcal{N}(z; \hat{z}, \Sigma)$. In this framework, the dual control problem reduces to stochastic optimal control of the augmented system. In this notation, the optimal exploration-exploitation trade-off—relative to the probabilistic priors defined above—can be written compactly as optimal control of the augmented system with a new observation model $p(y_t \,|\, z_t) = \mathcal{N}(y_t; \tilde{C} z_t, W)$ using $\tilde{C} = [C, 0]$ and a cost analogous to Equation (7.5).

Even for the linear and deterministic case, including the augmented state z results in a nonlinear system because θ and x interact multiplicatively

$$z_{t+1} = \begin{pmatrix} x_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} A(\theta_t) & 0 \\ 0 & I \end{pmatrix} z_t + \begin{pmatrix} B(\theta_t) \\ 0 \end{pmatrix} u_t =: \tilde{f}(z_t, u_t). \tag{7.2}$$

The parameters θ are assumed to be deterministic, but not known to the controller. This uncertainty is captured by the distribution $p(\theta)$ representing the lack of knowledge.

At initialization, $t = 0$, the belief over states and parameters is assumed to be Gaussian

$$p\left(\begin{bmatrix} x_0 \\ \theta_0 \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} x_0 \\ \theta_0 \end{bmatrix}; \begin{bmatrix} \hat{x}_0 \\ \hat{\theta}_0 \end{bmatrix}, \begin{bmatrix} \Sigma_0^{xx} & \Sigma_0^{x\theta} \\ \Sigma_0^{\theta x} & \Sigma_0^{\theta\theta} \end{bmatrix}\right). \tag{7.3}$$

For simplicity, we also assume that the dynamics do not change over time: $p(\theta_{t+1} \mid \theta_t) = \delta(\theta_{t+1} - \theta_t)$. This could be relaxed to an autoregressive model $p(\theta_{t+1} \mid \theta_t) = \mathcal{N}(\theta_{t+1}; \Theta\theta_t, \Xi)$, which would give additive terms in the derivations below. Throughout, we assume a finite horizon with terminal time T and a quadratic cost function in states and control inputs

$$\mathcal{L}(\mathbf{x}, \mathbf{u}) = \sum_{t=0}^{T} (x_t - x_t^{\text{ref}})^\intercal Q_t (x_t - x_t^{\text{ref}}) + \sum_{t=0}^{T-1} u_t^\intercal R_t u_t, \tag{7.4}$$

where $\mathbf{x}^{\text{ref}} = [x_0^{\text{ref}}, \ldots, x_T^{\text{ref}}]$ is a target trajectory. Q_t and R_t define state and control cost, they can be time-varying. The goal, in line with the standard in both optimal control and reinforcement learning, is to find the control sequence $\mathbf{u}$ that, at each t, minimizes the *expected cost* to the horizon

$$J_t(\mathbf{u}_{t:T-1}, p(z_t)) = \mathbb{E}_{z_t}\left[(x_t - x_t^{\text{ref}})^\intercal Q_t (x_t - x_t^{\text{ref}}) + u_t^\intercal R_t u_t + J_{t+1}(\mathbf{u}_{t+1:T-1}, p(z_{t+1})) \mid p(z_t)\right], \tag{7.5}$$

where past measurements $\mathbf{y}_{1:t}$, controls $\mathbf{u}_{1:t-1}$ and prior information $p(z_0)$ are incorporated into the belief $p(z_t)$, relative to which the expectation is calculated. Effectively, $p(z_t)$ serves as a bounded rationality approximation to the true information state [76, §6.5]. Since the equation above is recursive, the final element of the cost is

[76] Kumar and Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*, 1986

$$J_T(p(z_T)) = \mathbb{E}_{z_T}\left[(x_T - x_T^{\text{ref}})^\intercal Q_T (x_T - x_T^{\text{ref}}) \mid p(z_T)\right]. \tag{7.6}$$

The optimal control sequence minimizing this cost will be denoted $\mathbf{u}^*$, with associated cost

$$J_t^*(p(z_t)) = \min_{u_t} \mathbb{E}_{z_t}\left[(x_t - x_t^{\text{ref}})^\intercal Q_t (x_t - x_t^{\text{ref}}) + u_t^\intercal R_t u_t + J_{t+1}^*(p(z_{t+1})) \mid p(z_t)\right]. \tag{7.7}$$

This recursive formulation, if written out, amounts to alternating minimization and expectation steps. As u_t influences x_{t+1} and y_{t+1}, it enters the latter expectation nonlinearly. Classic optimal control is the base case with known θ, where $\mathbf{u}^*$ can be found by dynamic programming [19, 20].

[19] Bellman, *Dynamic Programming*, 1957

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

Unfortunately, the dynamics for the augmented system are nonlinear, even if the original physical system is linear. This is because inference is always nonlinear and future states influence future parameter beliefs, and vice versa. A first problem, not unique to dual control, is thus that inference is not analytically tractable, even under the Gaussian assumptions above. [42, 4] The standard remedy is to use approximations, most popularly the linearization of the extended Kalman filter [122, §5.2]. This gives a sequence of approximate Gaussian likelihood terms. But even incorporating these Gaussian likelihood terms into future dynamics is still intractable because it involves expectations over rational polynomial functions, whose degree increases with the length of the prediction horizon. The following section provides an intuition for this complexity, but also the descriptive power of the augmented state space.

[42] Fel'dbaum, "Dual Control Theory I-IV," 1960–1961

[4] Aoki, *Optimization of Stochastic Systems*, 1967

[122] Särkkä, *Bayesian Filtering and Smoothing*, 2013

Remark 7.1
Several authors have previously pointed out another possible construction of an augmented state: incorporating not the actual value *of the parameters θ_t in the state, but the parameters $\hat{\theta}_t, \Sigma_t^{\theta\theta}$ of a Gaussian belief $p(\theta_t \mid \hat{\theta}_t, \Sigma_t^{\theta\theta}) = \mathcal{N}(\theta_t; \hat{\theta}_t, \Sigma_t^{\theta\theta})$ over them. [65, 56] The advantage of this is that, if the state x_t is observed without noise, these belief parameters follow stochastic differential equations—more precisely, $\Sigma_t^{\theta\theta}$ follows an ordinary (deterministic) differential equation, while $\hat{\theta}_t$ follows a stochastic differential equation—and it can then be attempted to solve the control problem for these differential equations more directly.*

[65] Kappen, "Optimal Control Theory and the Linear Bellman Equation," 2011

[56] Hennig, "Optimal Reinforcement Learning for Gaussian Systems," 2011

While it can be a numerical advantage, this formulation of the augmented state also has some drawbacks, which is why we have here decided not to adopt it: First, the simplicity of the directly formalizable SDE vanishes in the POMDP setting, i. e. if the state is observed with noise. If the state observations are corrupted, the exact belief state is not a Gaussian process, so that the parameters $\hat{\theta}_t$ and $\Sigma_t^{\theta\theta}$ have no natural meaning. Approximate methods can be used to retain a Gaussian belief (and we will do so below), but the dynamics of $\hat{\theta}_t, \Sigma_t^{\theta\theta}$ are then intertwined with the chosen approximation (i. e. changing the approximation changes their dynamics), which causes additional complication. More generally speaking, it is not entirely natural to give differing treatment to the state x_t and parameters θ_t: Both state and parameters should thus be treated within the same framework; this also allows extending the framework to the case where also the parameters do follow an SDE.

7.1.1 An Educational Problem

To provide intuition for the sheer complexity of optimal dual control, consider the perhaps simplest possible example: the linear, scalar system

$$x_{t+1} = ax_t + bu_t + \xi_t, \tag{7.8}$$

with target $x_t^{\text{ref}} = 0$ and noise-free observations ($W = 0$). If a and b are known, the optimal u_t for a one-step horizon is

$$u^*_{t,\text{oracle}} = -\frac{abx_t}{R + b^2}. \tag{7.9}$$

Let now parameter b be uncertain, with current belief $p(b) = \mathcal{N}(b; \hat{b}_t, \sigma_t^2)$ at time t. The naïve option of simply replacing the parameter with the current mean estimate is known as *certainty equivalence* (CE) control[2] in the dual control literature [16]. The resulting control law is

[2] Sometimes this is also called *enforced* certainty equivalence.

[16] Bar-Shalom and Tse, "Dual Effect, Certainty Equivalence, and Separation in Stochastic Sontrol," 1974

$$u^*_{t,\text{CE}} = -\frac{a\hat{b}_t x_t}{R + \hat{b}_t^2}. \tag{7.10}$$

CE control is used in many adaptive control settings in practice, but has substantial deficiencies: If the uncertainty is large, the mean is not a good estimate, and the CE controller might apply completely useless control signals. This often results in large overshoots at the beginning.

A more elaborate solution is to compute the expected cost $\mathbb{E}_b[x_{t+1}^2 + Ru_t^2 \mid \hat{b}_t, \sigma_t^2]$ and then optimize for u_t. This gives *optimal feedback* (OF) or "cautious" control [38][3]:

[38] Dreyfus, "Some Types of Optimal Control of Stochastic Systems," 1964

[3] Dreyfus used the term "open loop optimal feedback" for his approach, a term that is misleading to modern readers because it is in fact a closed-loop algorithm.

$$u^*_{t,\text{OF}} = -\frac{a\hat{b}_t x_t}{R + \sigma_t^2 + \hat{b}_t^2}. \tag{7.11}$$

This control law reduces control actions in cases of high parameter uncertainty. This mitigates the main drawback of the CE controller, but leads to another problem: Since the OF controller decreases control with rising uncertainty, it can entirely prevent learning. Consider the posterior on b after observing x_{t+1}, which is a closed-form Gaussian because u_t is chosen by the controller and has no uncertainty

$$p(b \mid \hat{b}_{t+1}, \sigma_{t+1}^2) = \mathcal{N}(b; \hat{b}_{t+1}, \sigma_{t+1}^2) = \mathcal{N}\left(b; \frac{\sigma_t^2 u_t(bu_t + \xi_t) + \hat{b}_t D}{u_t^2\sigma_t^2 + D}, \frac{\sigma_t^2 D}{u_t^2 \sigma_t^2 + D}\right) \tag{7.12}$$

(b shows up in the fully observed $x_{t+1} = ax_t + bu_t + \xi_t$). The dual effect here is that the updated σ_{t+1}^2 depends on u_t. For large values of σ_t^2, according to (7.11), $u^*_{t,\text{OF}} \to 0$, and thus the uncertainty does not change ($\sigma_{t+1}^2 \approx \sigma_t^2$). The system will never learn or act, even for large x_t. This is known as the "turn-off phenomenon" [14].

[14] Bar-Shalom, "Stochastic Dynamic Programming: Caution and Probing," 1981

However, the derivation for OF control above amounts to minimizing Equation (7.5) for the myopic controller, where the horizon is only a single step long ($T = 1$). Therefore, OF control is indeed optimal for this case. By the optimality principle [20, §1.3], this means that Equation (7.11) is the optimal solution for the last step of *every* controller. But since it does not show any form of exploration or "probing" [15], a myopic controller is not enough to show the dual properties.

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

[15] Bar-Shalom and Tse, "Caution, Probing, and the Value of Information in the Control of Uncertain Systems," 1976

In order to expose the dual features, the horizon has to be at least of length $T = 2$. Since the optimal controller follows Bellman's principle, the solution proceeds backwards. The solution for the second control action u_1 is identical to the solution of the myopic controller (7.11); but after applying the first control action u_0, the belief over the unknown parameter b is updated according to Equation (7.12), resulting in

$$u_1^* = -\left[R + \frac{\sigma_0^2 D}{u_0^2\sigma_0^2 + D} + \left(\frac{\sigma_0^2 u_0(bu_0 + \xi_0) + \hat{b}_0 D}{u_0^2\sigma_0^2 + D}\right)^2\right]^{-1}\left[a\frac{\sigma_0^2 u_0(bu_0 + \xi_0) + \hat{b}_0 D}{u_0^2\sigma_0^2 + D}x_1\right]. \tag{7.13}$$

Inserting into Equation (7.7) gives

$$\begin{aligned} J_0^*(x_0) &= \min_{u_0} \mathbb{E}_{x_0}\left[x_0^2 + Ru_0^2 + \min_{u_1} \mathbb{E}_{x_1}\left[x_1^2 + Ru_1^2 + \mathbb{E}_{x_2}[x_2^2]\right]\right] \\ &= \min_{u_0}\left[x_0^2 + Ru_0^2 + \mathbb{E}_{\xi_0,b}\left[x_1^2 + R(u_1^*)^2 + \mathbb{E}_{\xi_1,b}\left[(x_1 + bu_1^* + \xi_1)^2 \mid \hat{b}_1, \sigma_1^2\right] \mid \hat{b}_0, \sigma_0^2\right]\right]. \end{aligned} \tag{7.14}$$

Since u_1^* from Equation (7.13) is already a rational function of fourth order in b, and shows up quadratically in Equation (7.14), the relevant expectations can not be computed in closed form. [4, §III.3] For this simple case, it is possible to compute the optimal dual control by performing the expectation through sampling b, ξ_0, ξ_1 from the prior. Figure 47 shows such samples of $\mathcal{L}(u_0)$ (—; one sample highlighted —), and the empirical expectation $J(u_0)$ (- - -). Each sample is a rational function of even leading order. The average dual cost has its minima not at zero, but to either side of it, reflecting the optimal amount of exploration in this particular belief state.

[4] Aoki, *Optimization of Stochastic Systems*, 1967

While it is not out of the question that the Monte Carlo solution can remain feasible for larger horizons, we are not aware of successful solutions for continuous state spaces (however, see the paper by Poupart et al. [112] for a sampling solution to Bayesian reinforcement learning in discrete spaces, including notes on the considerable computational complexity of this approach). The next section describes a tractable *analytic* approximation that does not involve samples.

[112] Poupart et al., "An Analytic Solution to Discrete Bayesian Reinforcement Learning," 2006

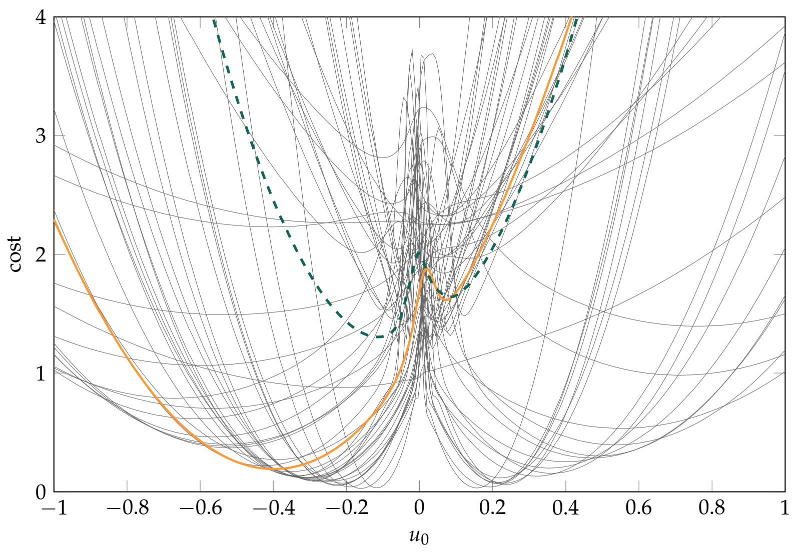


Figure 47: Computing the $T = 2$ dual cost for the simple system of Equation (7.8). Costs $\mathcal{L}(u_0)$ under optimal control on u_1 for sampled parameter b (——; one sample highlighted ——) and the expected dual cost $J(u_0)$ (- - -). The optimal u_0^* lies at the minimum of the dashed green line.

7.2 Approximate Dual Control for Linear Systems

In 1973, Tse, Bar-Shalom, and Meier proposed a method [142] and an algorithm [141] for approximate dual (AD) control, based on the series expansion of the cost-to-go. This is related to differential dynamic programming for the control of nonlinear dynamic systems [96]. It separates into three conceptual steps, where the first step represents the outer loop of the algorithm. Together they yield what, from a contemporary perspective, amounts to a structured Gaussian approximation to Bayesian RL:

[142] Tse, Bar-Shalom, and Meier, "Wide-sense Adaptive Dual Control for Nonlinear Stochastic Systems," 1973

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

[96] Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," 1966

① Perform a one-step prediction for an arbitrary control input u_t (as opposed to the analytically computed control inputs for later steps). Optimize u_t numerically by repeated computation of steps ② and ③ at varying u_t to minimize the approximate cost.

② Find an optimal trajectory for the deterministic part of the system under the mean model: the *nominal* trajectory under certainty equivalent control. For linear systems this is easy (see details below), for nonlinear ones it poses a nontrivial, but feasible nonlinear model predictive control problem [3, 34].

[3] Allgöwer et al., "Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview," 1999

[34] Diehl, Ferreau, and Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation," 2009

③ Around the nominal trajectory obtained in step ②, construct a local *quadratic expansion* that approximates the effects of future observations. Because the expansion is quadratic, an optimal control law relative to the deterministic system—the *perturbation control*—can be constructed by dynamic programming. Plugging this perturbation control into the residual dynamics of the approximate quadratic system gives an approximation for the cost-to-go. This step adds the cost of uncertainty to the deterministic control cost.

These three steps will be explained in detail in the subsequent sections. The interplay between the different parts of the algorithm is shown in Figure 48.

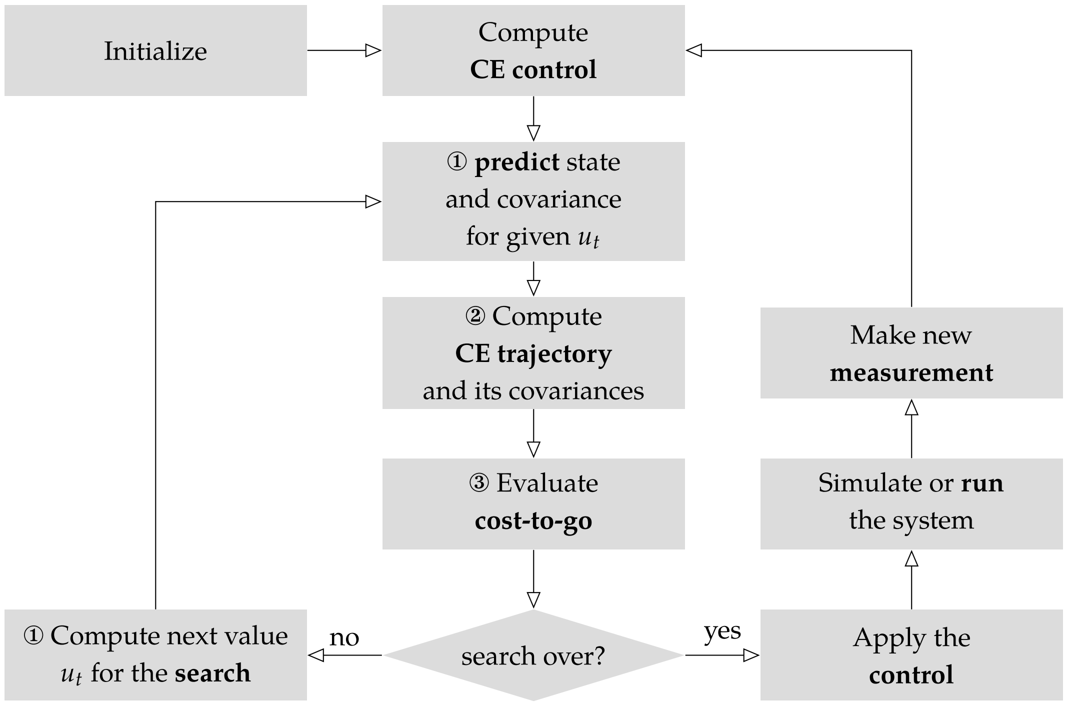

Initialize
Compute **CE control**
① **predict** state and covariance for given u_t
② Compute **CE trajectory** and its covariances
③ Evaluate **cost-to-go**
search over?
no
yes
① Compute next value u_t for the **search**
Apply the **control**
Simulate or **run** the system
Make new **measurement**

Figure 48: Flowchart of the approximate dual control algorithm to show the overall structure. Adapted from [141]. The left cycle is the outer loop of the algorithm, performing the nonlinear optimization.

The main purpose of this algorithm is to reduce the highly nonlinear optimization algorithm in multiple dimensions (control inputs over the horizon) to nonlinear optimization of only the first control input, by approximating the cost-to go, in an iterative fashion. While retaining the possibility to explore, this approach alleviates the curse of dimensionality. This procedure also circumvents the difficulties of the receding horizon approach in the dual control setting, as noted by Marafioti, Bitmead, and Hovd [94]: If the excitation is planned for future time steps, in closed-loop the excitation can be delayed at every time instance, leading to non-explorative behavior. Forcing the excitation to occur in the first step, this problem does not arise.

[94] Marafioti, Bitmead, and Hovd, "Persistently Exciting Model Predictive Control," 2014

7.2.1 The Nominal Reference Trajectory

The certainty equivalent model uses the assumption that the uncertain parameters θ coincide with their most likely value, the mean $\hat{\theta}$ of $p(\theta)$, and that the system propagates deterministically without noise. This means that the nominal parameters $\bar{\theta}$ are the current mean values $\hat{\theta}$, which decouples θ entirely from x in Equation (7.2), and the optimal control for the finite horizon problem can be computed by dynamic programming (DP) [20], yielding an optimal linear control law

[20] Bertsekas, *Dynamic Programming and Optimal Control*, 2005

$$\bar{u}_i^* = -\left(\bar{B}^\intercal \bar{V}_{i+1} \bar{B} + R_i\right)^{-1} \bar{B}^\intercal \left[\bar{V}_{i+1} \bar{A} \bar{x}_i + \bar{v}_{i+1}\right], \tag{7.15}$$

where we have momentarily simplified notation to $\bar{A} = A(\bar{\theta}_i), \bar{B} = B(\bar{\theta}_i)$, $\forall i$ because the $\bar{\theta}_i$ are constant. The $\bar{V}_i$ and $\bar{v}_i$ for $i = t+2, \dots, T$ are defined and computed recursively by the Riccati equation

$$\bar{V}_i = \bar{A}^\intercal \left(\bar{V}_{i+1} - \bar{V}_{i+1}\bar{B}\left(\bar{B}^\intercal \bar{V}_{i+1}\bar{B} + R_i\right)^{-1} \bar{B}^\intercal \bar{V}_{i+1}\right)\bar{A} + Q_i \qquad \bar{V}_T = Q_T \tag{7.16a}$$

$$\bar{v}_i = \bar{A}^\intercal \left(\bar{v}_{i+1} - \bar{V}_{i+1}\bar{B}\left(\bar{B}^\intercal \bar{V}_{i+1}\bar{B} + R_i\right)^{-1} \bar{B}^\intercal \bar{v}_{i+1}\right) - Q_i x_i^{\text{ref}} \qquad \bar{v}_T = -Q_T x_T^{\text{ref}}, \tag{7.16b}$$

where $\mathbf{x}^{\text{ref}}$ is the reference trajectory to be followed. This CE controller gives the *nominal trajectory* of inputs $\bar{\mathbf{u}}_{t+1:T-1}$ and states $\bar{\mathbf{x}}_{t+2:T}$, from the time after the first prediction until the end of the horizon. The true future trajectory is subject to stochasticity and uncertainty, but the deterministic nominal trajectory $\bar{\mathbf{x}}$, with its optimal control $\bar{\mathbf{u}}^*$ and associated nominal cost $\bar{J}_t^* = \mathcal{L}(\bar{\mathbf{x}}_{t:T}, \bar{\mathbf{u}}_{t:T-1}^*)$ provides a base, relative to which an approximation will be constructed.

7.2.2 Quadratic Expansion Around the Nominal Trajectory

The central idea of AD control is to project the nonlinear objective $J_t(\mathbf{u}_{t:T-1}, p(z_t))$ of Equation (7.5) onto a quadratic, by locally linearizing around the nominal trajectory $\bar{\mathbf{x}}$ and maintaining a joint Gaussian belief.

To do so, we introduce small perturbations around nominal cost, states, and control: $\Delta J_i = J_i - \bar{J}_i, \Delta z_i = z_i - \bar{z}_i$, and $\Delta u_i = u_i - \bar{u}_i$. These perturbations arise from both the stochasticity of the state and the parameter uncertainty. Note that a change in the state results in a change of the control signal because the optimal control signal in each step depends on the state. Even though the origin of the uncertainties is different (Δx arises from stochasticity and $\Delta\theta$ from the lack of knowledge), both can be modeled in a joint probability distribution.

Approximate Gaussian filtering ensures that beliefs over Δz remain Gaussian:

$$p(\Delta z_i) = \mathcal{N}\left[\begin{pmatrix}\Delta x_i \\ \Delta\theta_i\end{pmatrix}; \begin{pmatrix}\Delta\hat{x}_i \\ 0\end{pmatrix}, \begin{pmatrix}\Sigma_i^{xx} & \Sigma_i^{x\theta} \\ \Sigma_i^{\theta x} & \Sigma_i^{\theta\theta}\end{pmatrix}\right]. \tag{7.17}$$

Note that shifting the mean to the nominal trajectory does not change the uncertainty. Note further that the expected perturbation in the parameters is nil. This is because the parameters are assumed to be deterministic and are not affected by any state or input.

Calculating the Gaussian filtering updates is in principle not possible for future measurements, since it violates the causality principle [49, §1.4]. Nonetheless, it is possible to use the *expected* measurements

[49] Glad and Ljung, *Control Theory: Multivariable and Nonlinear Methods*, 2000

to simulate the effects of the future measurements on the uncertainty, since these effects are deterministic. This is sometimes referred to as *preposterior* analysis [114, §5A.3].

[114] Raiffa and Schlaifer, *Applied Statistical Decision Theory*, 1961

The cost is approximated, to second order around the nominal trajectory, by

$$J_t(\mathbf{u}_{t:T-1}, p(z_t)) = \bar{J}_t^* + \Delta J_t \approx \bar{J}_t^* + \Delta \tilde{J}_t, \tag{7.18}$$

where $\bar{J}_t^*$ is the optimal cost for the nominal system and $\Delta \tilde{J}_t$ is the approximate additional cost from the perturbation:

$$\Delta \tilde{J}_t := \mathbb{E}_{\mathbf{z}_{t:T}} \left[\sum_{i=t}^{T} \left\{ (\bar{x}_i - x_i^{\text{ref}})^\intercal Q_i \Delta x_i + \frac{1}{2} \Delta x_i^\intercal Q_i \Delta x_i \right\} + \sum_{i=t}^{T-1} \left\{ \bar{u}_i^\intercal R_i \Delta u_i + \frac{1}{2} \Delta u_i^\intercal R_i \Delta u_i \right\} \right]. \tag{7.19}$$

Although the uncertain parameters θ do not show up explicitly in the above equation, this step captures dual effects: The uncertainty of the trajectory $\Delta \mathbf{x}$ depends on θ via the dynamics. Higher uncertainty over θ at time $i-1$ causes higher predictive uncertainty over Δx_i (for each i), and thus increases the expectation of the quadratic term $\Delta x_i^\intercal Q_i \Delta x_i$. Control that decreases uncertainty in θ can lower this approximate cost, modeling the benefit of exploration. For the same reason, Equation (7.19) is in fact still not a quadratic function and has no closed form solution. To make it tractable, Tse and Bar-Shalom [141] make the ansatz that all terms in the expectation of Equation (7.19) can be written as $v_i + v_i^\intercal \Delta z_i + {}^1/{}_2 \Delta z_i^\intercal V_i \Delta z_i$. This amounts to applying dynamic programming on the perturbed system. Expectations over the cost under Gaussian beliefs on Δz can then be computed analytically. Because all $\Delta\theta$ have zero mean, linear terms in these quantities vanish in the expectation. This allows analytic minimization of the approximate optimal cost for each time step

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

$$\Delta \tilde{J}_i^*(p(z_i)) = \min_{\Delta u_i} \left\{ (x_i - x_i^{\text{ref}})^\intercal Q_i \Delta \hat{x}_{i|i} + \frac{1}{2} \Delta \hat{x}_{i|i}^\intercal Q_i \Delta \hat{x}_{i|i} + u_i^\intercal R_i \Delta u_i + \frac{1}{2} \Delta u_i^\intercal R_i \Delta u_i \right.$$
$$\left. + \frac{1}{2} \operatorname{tr} \left[Q_i \Sigma_{i|i}^{xx} \right] + \mathop{\mathbb{E}}_{\Delta z_{i+1}} \left[\Delta \tilde{J}_{i+1}^*(\mathbf{y}_{1:i+1}) \mid p(z_i) \right] \right\}, \tag{7.20}$$

which is feasible given an explicit description of the Gaussian filtering update. It is important to note that, assuming extended Kalman filtering, the update to the mean from *expected* future observations y_{i+1} is nil. This is because we expect to see measurements consistent with the current mean estimate. Nonetheless, the covariance changes depending on the control input u_i, which is the dual effect.

Following the dynamic programming equations for the perturbed problem (see Section 2.3.4), including the additional cost from uncertainty (see Section 2.3.5), the resulting cost amounts to [142]

[142] Tse, Bar-Shalom, and Meier, "Wide-sense Adaptive Dual Control for Nonlinear Stochastic Systems," 1973

$$\Delta \tilde{J}_t^*(p(z_t)) = \tilde{v}_{t+1} + \tilde{v}_{t+1}^\mathsf{T} \Delta \hat{z}_t + \frac{1}{2} \Delta \hat{z}_t^\mathsf{T} \tilde{V}_{t+1} \Delta \hat{z}_t \\ + \frac{1}{2} \operatorname{tr} \left\{ Q_T \Sigma_{T|T}^{xx} + \sum_{i=t}^{T-1} \left[Q_i \Sigma_{i|i}^{xx} + (\Sigma_{i+1|i} - \Sigma_{i+1|i+1}) \tilde{V}_{i+1} \right] \right\}. \quad (7.21)$$

Recalling that $\Delta\theta = 0$ and dropping the constant part, the dual cost can be approximated to be

$$J_t^d = \frac{1}{2} \operatorname{tr} \left\{ Q_T \Sigma_{T|T}^{xx} + \sum_{i=t}^{T-1} \left[Q_i \Sigma_{i|i}^{xx} + (\Sigma_{i+1|i} - \Sigma_{i+1|i+1}) \tilde{V}_{i+1} \right] \right\} \qquad (= \Delta \tilde{J}_t^* - \text{const}) \quad (7.22)$$

where the recursive equation

$$\tilde{V}_i = \tilde{A}_i^\mathsf{T} \left(\tilde{V}_{i+1} - \tilde{V}_{i+1} \tilde{B}_i \left(B_i^\mathsf{T} \tilde{V}_{i+1}^{xx} B_i + U_i \right)^{-1} \tilde{B}_i^\mathsf{T} \tilde{V}_{i+1} \right) \tilde{A}_i + \tilde{Q}_i \qquad \tilde{V}_T = \tilde{Q}_T \quad (7.23)$$

is defined for the augmented system (7.2), with $\tilde{A}_i = \frac{\partial}{\partial z} \tilde{f}|_{\bar{z}_i}$, $\tilde{B}_i = \frac{\partial}{\partial u} \tilde{f}|_{\bar{z}_i}$ and $\tilde{Q}_i = \operatorname{blkdiag}(Q_i, 0)$. The approximation to the overall cost is then $\bar{J}_t^* + J_t^d$, which is used as a cost function in the subsequent optimization procedure.

7.2.3 The Value of Information

The value of information refers to the fact that not all parameters of an uncertain model are equally important. If a certain parameter is important for the future control performance, it can be beneficial to identify this parameter and it might pay off to invest some energy in its identification. If a parameter does not have an important impact on the future cost, its identification can be neglected.

The second-order approximation defines a quadratic reference tracking problem based on the CE trajectory. The resulting cost-to-go contains the dual term (7.22), which adds a cost that results from the uncertainty. The term $Q_{i+1} \Sigma_{i+1}^{xx}$ represents the cost of the state uncertainty in future time steps. Since the source of this uncertainty is mostly control actions with uncertain outcome (such as an unknown gain), adding this term results in cautious behavior of the control system. The final term $\left[\Sigma_{i+1|i} - \Sigma_{i+1} \right] V_{i+1}$ is the most interesting part, as it represents an approximate measure for the value of information: It introduces a cost that weighs the covariance update $\left[\Sigma_{i+1|i} - \Sigma_{i+1} \right]$ by the value matrix V_{i+1}. This results in high cost for important parameters, indicated by large values in V_{i+1}, that are learned during the process, indicated by large values in the covariance update. If the parameters are either unimportant, precisely known, or can not be learned, this additional cost term vanishes. Thus, this term in the dual cost is an approximation to the value of information.

7.2.4 Approximate Dual Control by Optimizing the Next Input

The first step ① amounts to the outer loop of the overall algorithm. A gradient-free black-box optimization algorithm[4] is used to find the minimum of the overall cost function, including the dual cost. In every step, this algorithm proposes a control input u_t for which the cost is evaluated.

[4] We use Matlab's `fminsearch` in our implementation.

Depending on u_t, approximate filtering is carried out until the end of the horizon. The perturbation control is plugged into Equation (7.20) to give an analytic, recursive definition for $\tilde{V}_i$, and an approximation for the dual cost J_t^d, as a function of the current control input u_t.

Nonlinear optimization—through repetitions of steps ② and ③ for proposed locations u_t—then yields an approximation to the optimal dual control u_t^*. Conceptually the simplest part of the algorithm, this outer loop dominates computational cost because for every location u_t the whole machinery of ② and ③ has to be evaluated.

7.3 A Simplistic Experiment

The educational example from Section 7.1.1 can be used to show qualitative differences between cost functions for different approximation techniques, highlighting some of the dual control features. We compare the approximate dual controller introduced in Section 7.2 and two other controllers: The certainty equivalent (CE) controller and a controller minimizing the sum of CE cost and a Bayesian exploration bonus (EB) [149], which in this particular example amounts to

[149] Wittenmark, "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters," 1975

$$l_{\text{EB}} = \tau\sigma^2, \tag{7.24}$$

where τ is a scalar exploration weight and σ is the uncertainty of the parameter b. The additional cost term l_{EB} is evaluated for the predicted parameter covariance. This type of controller is sometimes also counted towards dual control, while being referred to as *explicit dual control*, where the dual features are obtained by a modified cost function [44].

[44] Filatov and Unbehauen, "Survey of Adaptive Dual Control Methods," 2000

For the noise-free linear system of Section 7.1.1, ($a = 1$ (known), $b = 2$, $p(b) = \mathcal{N}(b; 1, 10)$, $D = 10^{-1}$, $W = 0$, $Q = 1$, $R = 1$, $T = 2$), Figure 49 compares the cost functions of the different controllers and the sampling solution, which is close to the exact one, but only available for this very simple setup. All cost functions are shifted by an irrelevant constant. The CE cost is quadratic and indifferent about zero, i. e. the location of zero has no influence on the shape of this cost. The EB ($\tau = 0.1$) gives additional structure near zero that encourages learning. While qualitatively similar to the dual cost, its global minimum is almost at the same location as that of CE. The dual control approximates the sampling solution much closer.

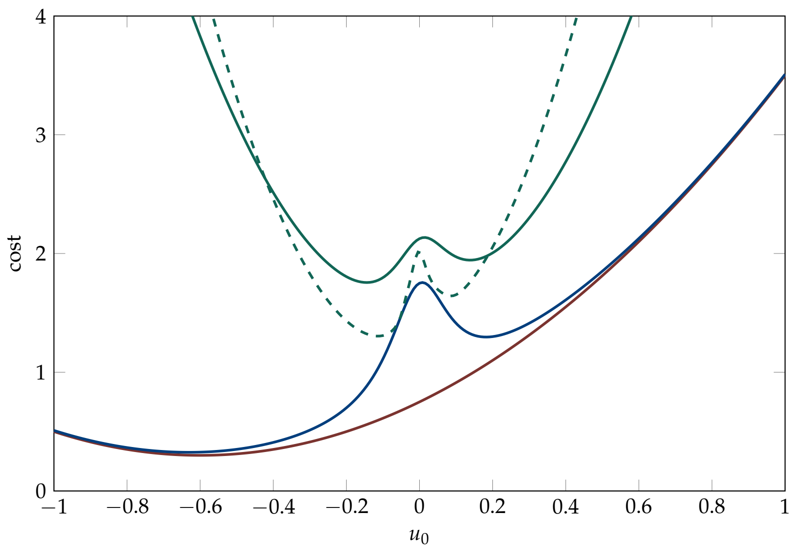


Figure 49: Comparison of sampling (- - -) to three approximations: CE (——), CE with Bayesian exploration bonus (——), and the approximate dual control constructed in Section 7.2 (——).

7.4 Conclusion

In this chapter, we discussed the basic idea of dual control and the fundamental problem of intractability arising from nested expectations and minimizations. We presented an existing dual control approximation that is based on series-expansion of the cost-to-go function, and we analyzed the resulting cost function to highlight the value of information. In a simple experiment, we showed the effect of the approximate dual control algorithm on the cost function and compared it to other approaches as well as an almost exact sampling solution.

The presented approximation to dual control is promising, but the original method only was applied to linear systems. This motivates the extension to nonlinear systems in the following chapter.

8 Nonlinear Dual Control

Optimal identification and control of uncertain dynamical systems can only be achieved approximately. The preceding chapter gave an introduction to the topic of dual control and the approximation originally used by Tse, Bar-Shalom, and Meier [142]. While this introductory work is relatively general, the explicit formulation [141] only applies to linear systems.

[142] Tse, Bar-Shalom, and Meier, "Wide-sense Adaptive Dual Control for Nonlinear Stochastic Systems," 1973

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

In this chapter, we extend the framework with ideas from contemporary machine learning. Specifically, we show the necessary changes to apply dual control to parametric linear regression in nonlinear feature spaces (Section 8.1.1), and how this idea can be carried further to work non-parametrically in a Gaussian process context (Section 8.1.2). We also give a simple, small-scale example for the use of this algorithm if the environment model is constructed with a feed-forward neural network rather than a Gaussian process (Section 8.1.3). Finally, we show experimental results from simulations to visualize the important features of the dual control framework (Section 8.2).

8.1 Extension to Nonlinear Models

Again, we consider a discrete-time, finite-horizon stochastic optimal control problem. In contrast to Section 7.1, we consider a nonlinear system of the form

$$x_{t+1} = f_t(x_t, u_t; \theta_t) + \xi_t \quad \text{(state dynamics)} \tag{8.1a}$$

$$y_t = Cx_t + \gamma_t \quad \text{(observation model),} \tag{8.1b}$$

with state $x_t \in \mathbb{R}^{n_x}$ and input $u_t \in \mathbb{R}$. The state is disturbed by zero-mean Gaussian noise ξ_t of covariance D and the measurement by γ_t of covariance W. The generative model is $p(x_{t+1} \mid x_t, u_t) = \mathcal{N}(x_{t+1}; f_t(x_t, u_t; \theta_t), D)$ and $p(y_t \mid x_t) = \mathcal{N}(y_t; Cx_t, W)$, with a linear map $C \in \mathbb{R}^{n_y \times n_x}$.

The extension to nonlinear models is guided by the desire to use a number of regression frameworks popular in machine learning, such as, parametric general least-squares regression, nonparametric Gaussian process regression, or feed-forward neural networks (including the base case of logistic regression).

8.1.1 Parametric Nonlinear Models

We begin with a generalized linear regression model of the form

$$x_{t+1} = A(\theta_t)\phi(x_t) + B(\theta_t)u_t + \xi_t, \tag{8.2}$$

where we use the same definition for the dynamics matrices $A(\theta_t)$ and $B(\theta_t)$ as for the linear system in Section 7.1. The difference is that the states x_t are now mapped into a nonlinear feature space. The nonlinear features $\phi(x)$ can in principle be any function (popular choices include sines and cosines, radial basis functions, sigmoids, polynomials and others), with the caveat that their structure crucially influences the properties of the model. From a modeling perspective, this approach is quite standard for machine learning. However, the dynamical learning setting requires an adaptation: To allow the modeling of higher-order dynamical systems, the original states must be included. This results in feature vectors of the form $\phi(x)^\intercal = [x^\intercal, \varphi(x)^\intercal]$, consisting of the linear representation, augmented by general features $\varphi(x)$. While we chose to model the input response linearly, it can of course also be included in the nonlinear feature space as $\phi(x, u)$, which complicates the equations below but otherwise will work as expected.

The main challenge to apply the approximate dual control scheme introduced in Section 7.2 is that the optimal control for nonlinear dynamical systems can not be optimized in closed form using dynamic programming, not even for the deterministic nominal system. Instead, we find the nominal reference trajectory using nonlinear model predictive control [3, 34]. This adds computational cost, and requires some care to achieve stable optimization performance for specific system setups.

[3] Allgöwer et al., "Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview," 1999

[34] Diehl, Ferreau, and Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation," 2009

State filtering from observations is also more involved in the case of nonlinear dynamics. In the experiments reported below, we stayed within the extended Kalman filtering framework [122, §5.2] to retain Gaussian beliefs over states and parameters. Other methods with this property will work similarly, this includes relatively standard options like unscented Kalman filtering [143], but also more recent developments in machine learning and probabilistic control, such as analytic moment propagation if the features φ are selected accordingly [33].

[122] Särkkä, *Bayesian Filtering and Smoothing*, 2013

[143] Uhlmann, "Dynamic Map Building and Localization: New Theoretical Foundations," 1995

[33] Deisenroth and Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," 2011

The final problem is the generalization of the dual cost formulation to the nonlinear dynamics. We take a relatively simplistic approach, which nevertheless turns out to work well. A linearization gives locally linear dynamics whose structure closely matches Equation (7.2):

$$\begin{aligned} z_{t+1} = \begin{pmatrix} \bar{x}_{t+1} + \Delta x_{t+1} \\ \bar{\theta}_{t+1} + \Delta\theta_{t+1} \end{pmatrix} &= \begin{pmatrix} A(\theta_t) & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \phi(\bar{x}_t + \Delta x_t) \\ \bar{\theta}_t + \Delta\theta_t \end{pmatrix} + \begin{pmatrix} B(\theta_t) \\ 0 \end{pmatrix} u_t + \begin{pmatrix} \xi_t \\ 0 \end{pmatrix} \\ &\approx \begin{pmatrix} A(\bar{\theta}_t) & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \phi(\bar{x}_t) \\ \bar{\theta}_t \end{pmatrix} + \begin{pmatrix} B(\bar{\theta}_t) \\ 0 \end{pmatrix} u_t + \begin{pmatrix} \xi_t \\ 0 \end{pmatrix} \\ &\quad + \begin{pmatrix} A(\bar{\theta}_t)\frac{\partial}{\partial x_t}\phi(\bar{x}_t) & \frac{\partial}{\partial\theta_t}\left(A(\bar{\theta}_t)\phi(\bar{x}_t) + B(\bar{\theta}_t)u_t\right) \\ 0 & I \end{pmatrix} \begin{pmatrix} \Delta x_t \\ \Delta\theta_t \end{pmatrix}. \end{aligned} \tag{8.3}$$

This essentially amounts to extended Kalman filtering on the augmented state. Using this linearization, the approximation described in Section 7.2 can be applied analogously, which results in an approximation for the dual cost.

8.1.2 Nonparametric Gaussian Process Models

The above treatment of parametric nonlinear models makes it comparably easy to extend the description from finitely many feature functions to an infinite-dimensional feature space defining a Gaussian process (GP) dynamics model:

$$x_{t+1} = f(x_t) + B(\theta_t)u_t + \xi_t \tag{8.4}$$

Assume that the true dynamics function f is a draw from a Gaussian process prior $p(f) = \mathcal{GP}(f; \bar{m}, \bar{k})$ with prior mean function $\bar{m} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, and prior covariance function (kernel) $\bar{k} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_x}$. This is using the widely used notion of "multi-output regression" [115, § 9.1], i. e. formulating the covariance as

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

$$\operatorname{cov}(f_i(x), f_j(x')) = \bar{k}_{ij}(x, x'). \tag{8.5}$$

To simplify the treatment, we assume that the covariance factorizes between inputs and outputs, i. e. $\bar{k}_{ij}(x, x') = U_{ij}k(x, x')$ with a univariate kernel $k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}$ and a positive semidefinite matrix $U \in \mathbb{R}^{n_x \times n_x}$ of output covariances.

Using Mercer's theorem [73, §3.a], we can decompose the kernel into a converging series over eigenfunctions $\phi_l(x)$, as

[73] König, *Eigenvalue Distribution of Compact Operators*, 1986

$$k(x, x') = \sum_{l=1}^{\infty} \lambda_l \phi_l(x)\phi_l^*(x') := \Phi\Lambda\Phi^\intercal, \tag{8.6}$$

where we have defined the infinite-dimensional inner product $\Phi\Lambda\Phi^\intercal$ for the feature vectors Φ and the infinite-dimensional diagonal matrix Λ with elements $\Lambda_{ll} = \lambda_l$.

Using this notation, we can use the suggestive notation $f_t(x_t) = L\Omega\Phi(x_t)$ for the generative model

$$f^i(x_t) = \sum_{j=1}^{n} L_{ij} \sum_{l=1}^{\infty} \Omega^{jl} \phi_l(x_t), \tag{8.7}$$

where L is a matrix satisfying $LL^\intercal = U$ (e. g., the Cholesky decomposition), and the elements of Ω are draws from the "white" Gaussian process $\Omega^{jl} \sim \mathcal{N}(0, \lambda_l)$. Because of Mercer's theorem above, Equation (8.7) exists in mean-square expectation, and is well-defined in this sense. [1, 115] This notation allows writing the current GP belief as a nonparametric prior with mean $\hat{\theta}_0$ and covariance $\Sigma_0^{\theta\theta} = U \otimes (\Phi\Lambda\Phi^\intercal)$.

[1] Adler, *The Geometry of Random Fields*, 1981

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

Using this notation, a tedious but straightforward linear algebra derivation (see Appendix A.3) shows that the posterior over $z^\intercal = [x^\intercal, \theta^\intercal]$ after a number t of EKF-linearized Gaussian observations is a tractable Gaussian process, for which the Gram matrix

$$\mathcal{G} = \mathcal{P} + \mathcal{D} + \mathcal{K} + \mathcal{A}^{-1}\mathcal{W}\mathcal{A}^{-\intercal} \tag{8.8}$$

consists of the parts

$$\mathcal{P} = \begin{bmatrix} A_0 \Sigma_0^{xx} A_0^\intercal & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathcal{D} = (D \otimes I) \qquad \mathcal{K} = k(\boldsymbol{y}_{0:t-1}, \boldsymbol{y}_{0:t-1}) \qquad \mathcal{W} = (W \otimes I) \tag{8.9}$$

of appropriate size, depending on the current time t. The multi-step state transition matrix

$$\mathcal{A} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ A_1 & I & 0 & \cdots & 0 \\ A_2 A_1 & A_2 & I & \cdots & 0 \\ \vdots & & & \ddots & \\ A_{t-1}\cdots A_1 & A_{t-1}\cdots A_2 & \cdots & & I \end{bmatrix} \tag{8.10}$$

is needed to account for the effect of the measurement noise W over time. The dynamics matrices A_t are the Jacobians $\nabla_x f(x)|_{x_t}$.

The posterior mean now evaluates to

$$\begin{bmatrix} \hat{x}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} \hat{x}_{t-1} \\ 0 \end{bmatrix} + \begin{bmatrix} \Phi(\hat{x}_{t-1})\Lambda\Phi(\boldsymbol{y}_{0:t-1})^\intercal \\ \Lambda\Phi(\boldsymbol{y}_{0:t-1})^\intercal \end{bmatrix} \mathcal{G}^{-1}(\boldsymbol{y}_{1:t} - \mathcal{C}\mathcal{A}_{:,0}A_0\hat{x}_0), \tag{8.11}$$

with $\mathcal{C} = (C \otimes I)$ of appropriate size.

The posterior covariance is comprised of

$$\bar{\Sigma}_t^{xx} = A_{t-1}\Sigma_{t-1}^{xx}A_{t-1}^\intercal + D + \Phi(\hat{x}_{t-1})\Sigma_{t-1}^{\theta x}A_{t-1}^\intercal + A_{t-1}\Sigma_{t-1}^{x\theta}\Phi(\hat{x}_{t-1})^\intercal + \Phi(\hat{x}_{t-1})\Sigma_{t-1}^{\theta\theta}\Phi(\hat{x}_{t-1})^\intercal \tag{8.12a}$$

$$\Sigma_t^{xx} = \bar{\Sigma}_t^{xx} - \bar{\Sigma}_t^{xx}\left[\bar{\Sigma}_t^{xx} + W\right]^{-1}\bar{\Sigma}_t^{xx} \tag{8.12b}$$

$$\Sigma_t^{x\theta} = \mathcal{A}_{t-1,:}\Phi(\boldsymbol{y}_{0:t-1})\Lambda - \mathcal{A}_{t-1,:}\left[\mathcal{P} + \mathcal{K} + \mathcal{D}\right]\mathcal{G}^{-1}\Phi(\boldsymbol{y}_{0:t-1})\Lambda \tag{8.12c}$$

$$\Sigma_t^{\theta x} = (\Sigma_t^{x\theta})^\intercal \tag{8.12d}$$

$$\Sigma_t^{\theta\theta} = \Lambda - \Lambda\Phi(\boldsymbol{y}_{0:t-1})^\intercal\mathcal{G}^{-1}\Phi(\boldsymbol{y}_{0:t-1})\Lambda. \tag{8.12e}$$

This formulation, together with the expositions in the preceding sections, defines a nonparametric dual control algorithm for Gaussian process priors. It is important to stress that this posterior is indeed "tractable" in so far as it depends only on a Gram matrix of size $n_x t \times n_x t$, and the posterior over any $f(x)$ can be computed in time $\mathcal{O}((n_x t)^3)$, despite the infinite-dimensional state space.

An Approximation of Constant Computational Cost

In practical control applications, continuously rising inference cost is rarely acceptable. It is thus necessary to project the GP belief onto a finite representation, replacing the infinite sum in Equation (8.6) with a finite one, to bound the computational cost of the matrix inversion in Equations (8.11), (8.12c) and (8.12e). We use a finite representation to approximate the kernel function, similar to the ones recently presented by Rahimi and Recht [113], and by Lázaro-Gredilla et al. [82]. The approximation is described in Section 1.6.2.

[113] Rahimi and Recht, "Random Features for Large-Scale Kernel Machines," 2008

[82] Lázaro-Gredilla et al., "Sparse Spectrum Gaussian Process Regression," 2010

8.1.3 Feedforward Neural Network Models

Another extension of the parametric linear models of Section 8.1.1 is to allow for a nonlinear parametrization of the dynamics function

$$f(x; \boldsymbol{\theta}) = \sum_i \theta_i^{\text{lin}}\phi_i(x; \theta_i^{\text{nonlin}}). \tag{8.13}$$

A particularly interesting example of this structure are multilayer perceptrons. Consider a two-layer network with logistic link function σ, defined for each state x^i

$$x_{t+1}^i = f^i(x_t) = \sum_l \theta_{il}^{\text{out}}\sigma\left(\sum_j \theta_{lj}^{\text{in}}x_t^j + \theta_l^{\text{bias}}\right), \tag{8.14}$$

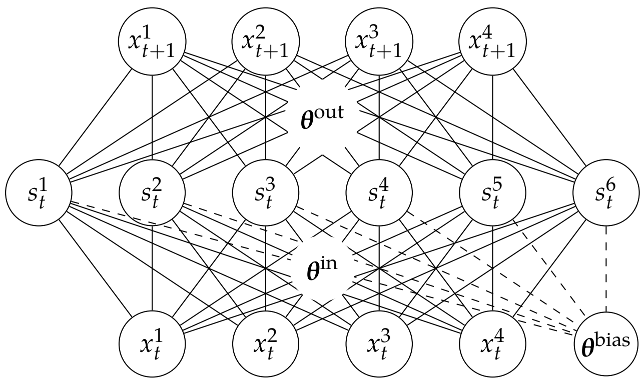
where $\boldsymbol{\theta}^{\text{out}}$ are the weights from the latent to the output layer, $\boldsymbol{\theta}^{\text{in}}$ are the weights from input to hidden units, and $\boldsymbol{\theta}^{\text{bias}}$ are the biases of the hidden units (see Figure 50). Superscripts denote vector elements.

Neural networks for control applications were proposed multiple times, see e. g., [102]. Instead of using backpropagation and stochastic gradient descent as in most applications of neural networks [121, 118],

[102] Nguyen and Widrow, "Neural Networks for Self-learning Control Systems," 1990

[121] Rumelhart, Hinton, and Williams, "Learning Representations by Back-propagating Errors," 1986

[118] Robbins and Monro, "A Stochastic Approximation Method," 1951



$$x^i_{t+1} = f^i(x_t) = \sum_j \theta^{\text{out}}_{ij} s^j_t$$

$$s^i_t = \sigma\left(\sum_j \theta^{\text{in}}_{ij} x^j_t + \theta^{\text{bias}}_i\right)$$

$$x^i_t$$

Figure 50: Two-layer feed-forward neural network. Sketch to illustrate the structure of Equation (8.14).

the EKF inference procedure can be used to train the weights as well [130]. This is possible because the EKF linearization can be applied for the nonlinear link function, e. g., the logistic function. Speaking in terms of feature functions, not only the weight of each feature but also the shape (steepness) can be inferred. A limiting factor for this inference is the number of data points: the more features and parameters are introduced, the more data points are necessary to learn.

[130] Singhal and Wu, "Training Multilayer Perceptrons with the Extended Kalman Algorithm," 1989

Using the state augmentation $z^\intercal = (x^\intercal\ \theta^{\text{out}\intercal}\ \theta^{\text{in}\intercal}\ \theta^{\text{bias}\intercal})$, and linearizing w. r. t. all parameters in each step, the EKF inference on the neural network parameters allows us not only to apply relatively cheap inference on them, but also to use the dual control framework to plan control signals, accounting for the effect of future observations and the subsequent change in the belief. This means the approximate dual controller described in Section 7.2 can identify those parts of the neural net that are relevant for applying optimal control to the problem at hand. In Section 8.2.2, we show an experiment with these properties.

8.2 Experiments

We show the features of dual control on a set of different simulated control problems. Relative to a lower bound (LB), which represents the minimal cost a fully informed system can obtain, we compare three different controllers: A simple certainty equivalent (CE) controller, a CE controller with exploration bonus (EB) and the approximate dual (AD) controller.

The exploration bonus [149] for multiple parameters is defined as

[149] Wittenmark, "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters," 1975

$$l_{\text{EB}} = \tau\,\mathrm{tr}\left[\Sigma^{\theta\theta}\right], \tag{8.15}$$

where τ is a scalar exploration weight and $\Sigma^{\theta\theta}$ is the uncertainty on the parameters. The exploration bonus is evaluated for the predicted parameter covariance where the prediction time is chosen according to the order of the system so that the effect of the current control signal shows up in the belief over the parameters. Every experiment was repeated 50 times with different random seeds, which were shared across controllers for better comparability.

All systems presented below are very simple setups. Their primary point is to show qualitative differences of the controllers' behaviors. The experiments were done with different approximations from the preceding section to show experimental feasibility for each of them.

The feature set used for a specific application is part of the prior assumptions for that application. Large uncertainty requires flexible models, which take longer to converge and require more exploration. Feature selection is important, but since it is independent of the dual control framework itself and a broad topic on its own, it is beyond the scope of this thesis. In the following experiments, different feature sets are used both as examples for the flexibility of the framework, but also to model different structural knowledge about the problems at hand.

8.2.1 Time-dependent Exploration

A cart on a rail is a simple example for a dynamical system. Combined with a nonlinearly varying slope, a simple nonlinear system can be constructed. The dynamics, prior beliefs, and true values for the parameters are chosen to be

$$x_{t+1} = \begin{bmatrix} 1 & 0.4 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 & 0 \\ \theta^1 & \theta^2 \end{bmatrix} \begin{bmatrix} \varphi^1(x_t^1) \\ \varphi^2(x_t^1) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t, \tag{8.16}$$

with

$$\theta \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \qquad \theta_{\text{true}} = \begin{bmatrix} 0.8 \\ 0.4 \end{bmatrix}, \tag{8.17}$$

where superscripts denote vector elements. The nonlinear functions φ are shifted logistic functions of the form

$$\varphi^1(x) = -\frac{1}{1+e^{(x+5)}} \qquad \varphi^2(x) = \frac{1}{1+e^{-(x-5)}}, \tag{8.18}$$

and disturbance/noise is chosen to be $D = W = 10^{-2} I$. We use this setup as a testbed for a time-structured exploration problem. The actual system and its dynamics are relatively irrelevant here, as we focus on a complication caused by the cost function: The reference to be tracked is

$$\mathbf{x}^{\text{ref}}_{0:11} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\text{ref}}_{12:14} = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\text{ref}}_{15} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\text{ref}}_{16:18} = \begin{bmatrix} -10 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\text{ref}}_{19:20} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{8.19}$$

which is also shown in each plot of Figure 51 (- - -). The state weighting is time-dependent with

$$\mathbf{Q}_{0:4} = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{Q}_{5:10} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{Q}_{11:20} = \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix}, \tag{8.20}$$

and control cost is relatively low: $R = 10^{-3}$. The task, thus, is to first keep the cart fixed at the origin, for the first 4 time steps. This is followed by a "loose" period between time steps 5 and 10. Then, the cart has to be moved to one side, back to the center, to the other side, and back again, all at high cost. A good exploration strategy in this setting is to act cautiously for the first 4 time steps, then aggressively explore in the "loose" phase, to finally be able to control the motion with high precision.

The inference model is a GP with approximated SE kernel, as described in Section 1.6.2. We use 30 alternating sine and cosine features that are distributed according to the power spectrum of the full SE kernel. Since the true nonlinearity of Equation (8.18) is not of this form, the approximation is out of model and the lower bound controller only represents a perfectly learned, but still not exact, model.

Figure 51 shows a density estimated from 50 state trajectories for the four different controllers. The lower bound controller (top) controls precisely at times of high cost, and does nothing for times with zero cost, controlling perfectly up to the measurement and state disturbances. The certainty equivalent controller (second from top) never explores actively, it only learns "accidentally" from observations arising during the run. Since the initial trajectory requires little action, it is left with a bad model when the reference starts to move at time step 12. The exploration bonus controller (second from bottom) continuously explores, because it has no way of knowing about the "loose" phase ahead. Of course, this strategy incurs a higher cost initially. The dual controller (bottom) effectively holds off exploration until it reaches the "loose" phase, where it explores aggressively.

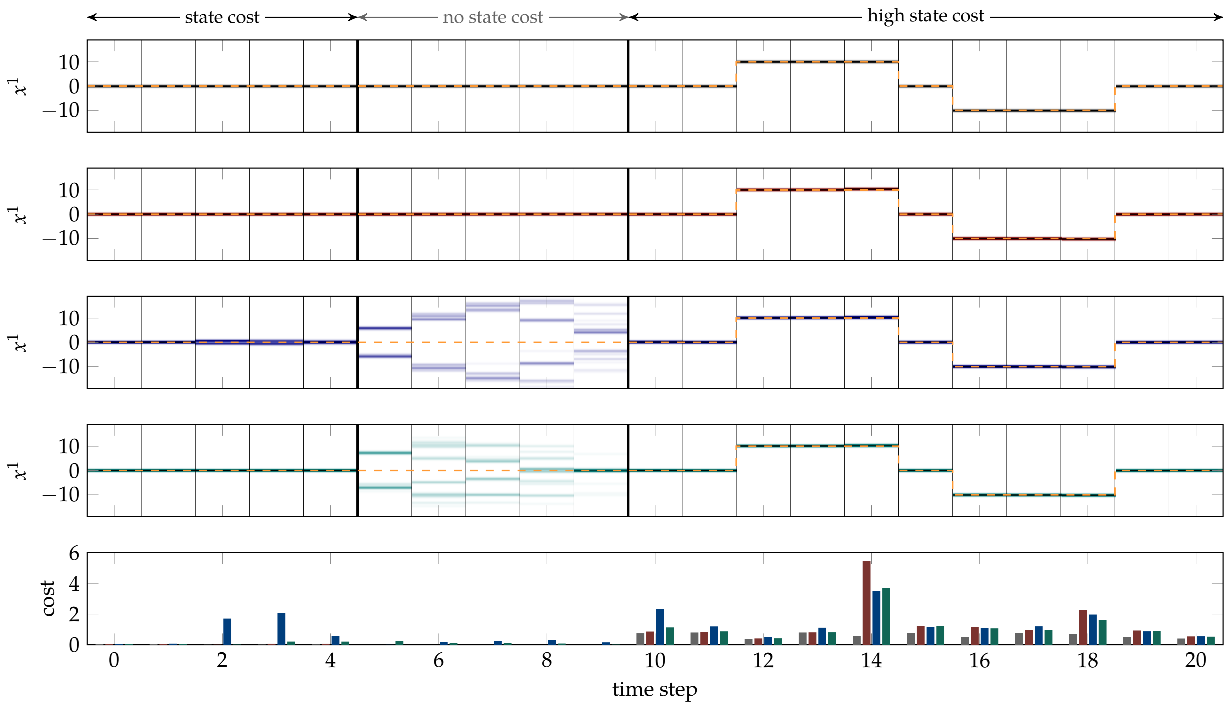


Figure 51: Controller comparison for time-dependent exploration. **Top four:** Density estimate for 50 trajectories (second state). From top to bottom: lower bound (▬), certainty equivalent control (▬), CE with Bayesian exploration bonus (▬), approximate dual control (▬). Reference trajectory (- - -). **Bottom:** The mean cost per time step is shown in the bottom plot, with colors matching the controllers noted above.

8.2.2 Relevance-dependent Exploration

The system, including nonlinearities, for this experiment is the same as before, although with noise parameters $D = W = 10^{-3}I$. The reference trajectory and state weighting are much simpler:

$$\mathbf{x}^{\text{ref}}_{0:11} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\text{ref}}_{12:18} = \begin{bmatrix} 10 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{\text{ref}}_{18:20} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{8.21}$$

with the time-dependent weighting

$$\mathbf{Q}_{0:10} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{Q}_{11:20} = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}, \tag{8.22}$$

allowing for identification in the beginning, while penalizing deviations of the first state in later time steps.

Important to note here is that the reference trajectory only passes areas of the state space where φ^1 is strong, and φ^2 is negligible. Good exploration thus will ignore the second parameter θ^2, but this can only be found through reasoning about future trajectories.

In this experiment, the learned model is of the neural network form described in Section 8.1.3. We use 4 logistic features (see Equation (8.14)) with two free parameters each (θ^{in} and θ^{out}) and equally spaced θ^{bias} between -5 and 5, the locations of the true nonlinear features. This means it is possible to learn the perfect model in this case.

Figure 52 shows a density estimated from 50 state trajectories for the four different controllers. Because of symmetry in the cost function and feature functions, EB (with $\tau = 1$) can not "decide" between the relevant θ^1 and the irrelevant θ^2, identifying both under high control cost. It thus reduces the uncertainty on θ^2, which does not help the subsequent control. The AD controller ignores θ^2 completely, and only identifies θ^1 in early phases, leading to good control performance.

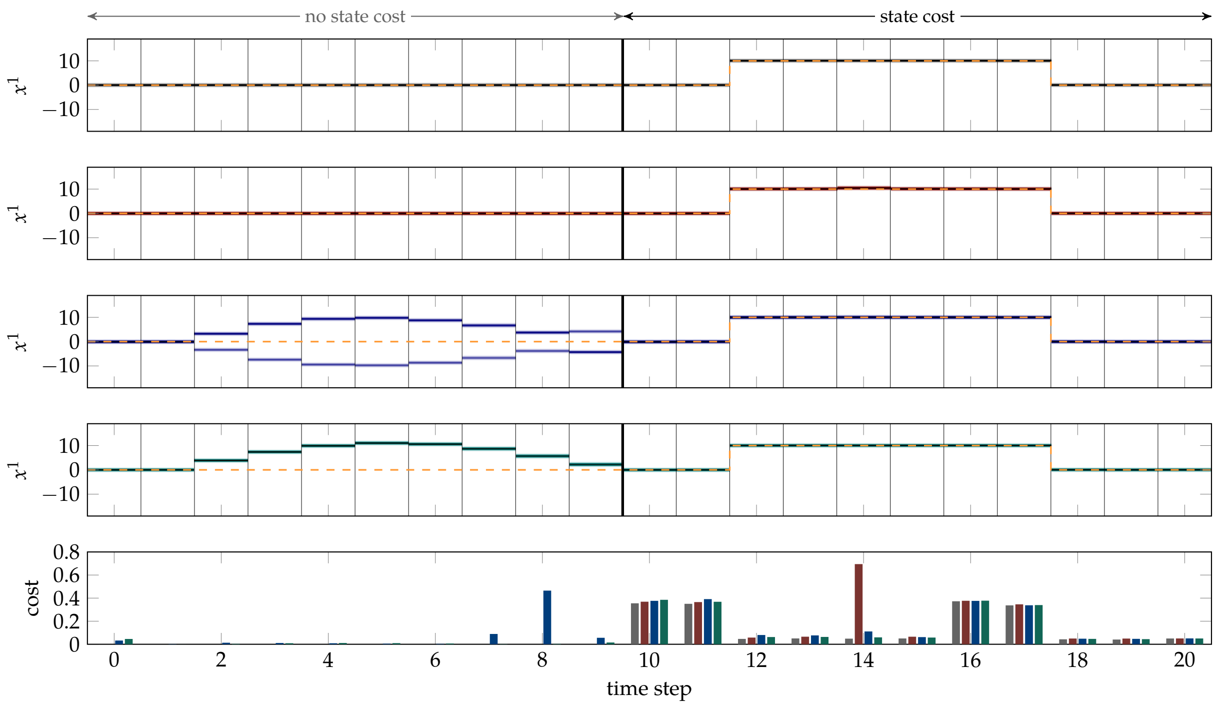


Figure 52: Controller comparison for relevance-dependent exploration. **Top four:** Density estimate for 50 trajectories (second state). From top to bottom: lower bound (⬛), certainty equivalent control (🟥), CE with Bayesian exploration bonus (🟦), approximate dual control (🟩). Reference trajectory (- - -). **Bottom:** The mean cost per time step is shown in the bottom plot, with colors matching the controllers noted above.

8.2.3 Information Maintenance

The last experiment is again similar to Section 8.2.1, but uses a different set of nonlinear functions: shifted, non-normalized Gaussian functions

$$\varphi^1(x) = e^{-\frac{(x-2)^2}{2}} \qquad \varphi^2(x) = e^{-\frac{(x+2)^2}{2}} \qquad \theta_{\text{true}} = \begin{bmatrix} 1.0 \\ 0.8 \end{bmatrix}. \tag{8.23}$$

For this experiment, the model is learned with parametric linear regression, according to Section 8.1.1. The fundamental difference to the other experimental setups is that the model now assumes *parameter drift*. This results in growing uncertainty for the parameters over time. The true parameters are kept constant for simplicity.

The reference to be tracked passes through both nonlinear features but then stays at one of them:

$$\mathbf{x}_{0:6}^{\text{ref}} = \begin{bmatrix} -5 \\ 0 \end{bmatrix} \qquad \mathbf{x}_{7}^{\text{ref}} = \begin{bmatrix} -4 \\ 0 \end{bmatrix} \qquad \mathbf{x}_{8}^{\text{ref}} = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \qquad \mathbf{x}_{9}^{\text{ref}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}_{10:20}^{\text{ref}} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}. \tag{8.24}$$

The cost structure is

$$\mathbf{Q}_{0:5} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{Q}_{6:20} = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}, \tag{8.25}$$

such that state deviations from the reference are penalized starting from time instant 6.

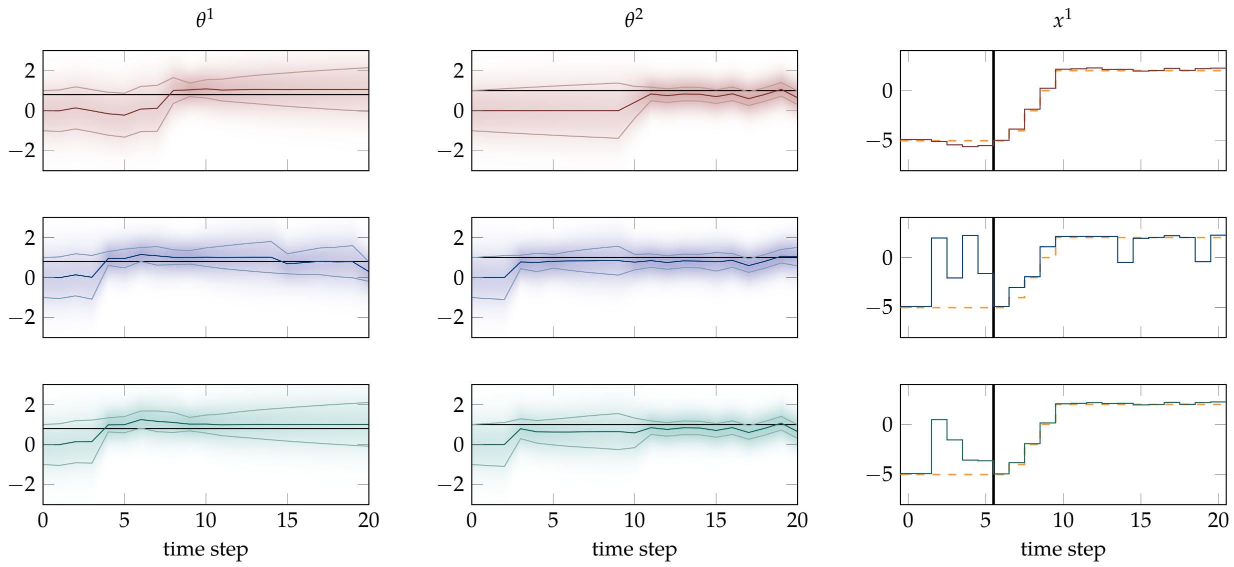


Figure 53: Controller comparison under fading beliefs. Parameter knowledge (left, middle) and state trajectory (right) for different controllers. From top to bottom: certainty equivalent control (——), CE with Bayesian exploration bonus (——), approximate dual control (——). The true parameters are the black lines.

Figure 53 shows the parameter belief and relevant state of a single run of this experiment over time. It shows that the in the beginning necessary parameter θ^1 is learned early by EB and the AD controller, while CE learns only "accidentally". The EB controller also learns the second parameter in the beginning, even though the knowledge will be lost over time. When the trajectory reaches the zone of the second parameter, the EB controller tries to lower the growing uncertainty over the first parameter θ^1 every now and then (visible by the change in state x^1), incurring high cost. AD control completely ignores the growing uncertainty on θ^1 after reaching the area of θ^2, thus preventing unnecessary exploration.

8.2.4 Quantitative Comparison

The above experiments aim at emphasizing qualitative strengths of AD control over simpler approximations. It is desirable for a controller to deal with flexible models of many parameters, many of which will invariably be superfluous for a given trajectory. For reference, Table 3 also shows quantitative results: Averages and standard deviations of the cost, from the 50 runs for each controller. The AD controller shows good performance overall; interestingly, it also has low variance. CE and EB were more prone to instabilities.

	Exp. 8.2.1		Exp. 8.2.2		Exp. 8.2.3	
	mean	std	mean	std	mean	std
LB	7.15	3.85	1.75	0.33	0.66	0.51
CE	15.72	5.20	2.49	0.74	1.76	0.90
EB	20.88	6.74	2.64	0.37	84.91	6.77
AD	14.33	5.40	1.96	0.34	1.62	0.56

Table 3: Quantitative comparison of different controllers. Average and standard deviation of costs in the experiments for 50 runs.

8.3 Conclusion

The dual control framework developed by Tse and Bar-Shalom [141] is a promising approach to the intractable dual control problem because the approximation to the dual cost retains the value of information. In this chapter, we showed how this method can be applied to contemporary nonlinear inference methods from machine learning, including approximate Gaussian process regression and multi-layer networks. The result is a tractable approximation that captures notions of structured exploration, like the value of waiting for future exploration opportunities, and distinguishing relevant from irrelevant model parameters.

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

On several simulated systems we showed the potential of this framework in the nonlinear setting. Retaining the value of information is crucial for applying dual control for rich model classes because simpler approaches—like exploration bonuses or constant excitation—can show limited performance in these cases.

9 Dual Control for Buildings

Building climate control is a problem that has raised significant attention in recent years, due to its potential impact on the worldwide energy consumption: A large amount of the globally consumed energy is used for buildings. [80] This has sparked recent interest in model predictive control (MPC) for buildings [54, 106, 87, 11], making use of predictions of the model as well as external error sources, such as weather conditions and occupancy. However, these techniques require a sufficiently accurate system model. As the parameters of the model generally vary with the building and potentially with time, parameter identification has to be performed individually for each building during operation, which can be expensive.

[80] Lausten, *Energy Efficiency Requirements In Building Codes, Energy Efficiency Policies For New Buildings*, 2008

[54] Gwerder and Tödtli, "Predictive Control for Integrated Room Automation," 2005

[106] Oldewurtel et al., "Energy Efficient Building Climate Control Using Stochastic Model Predictive Control and Weather Predictions," 2010

[87] Ma, Matusko, and Borrelli, "Stochastic Model Predictive Control for Building HVAC Systems: Complexity and Conservatism," 2015

[11] Aswani et al., "Reducing Transient and Steady State Electricity Consumption in HVAC Using Learning-Based Model-Predictive Control," 2012

Adaptive controllers offer the potential to obtain accurate models for a low energy footprint over the whole lifetime of the building. While passively learning adaptive control systems can only learn by evaluating past measurements, dual controllers, as presented in Chapters 7 and 8, can enhance the learning procedure by also reasoning about the effect of current actions on the future control performance. This way, dual control can make use of certain parts of the problem structure to identify the model more efficiently than purely passive control systems. For example, a dual controller can identify at times where the energy cost or demand is low (making use of real-time-pricing or day/night tariffs) to obtain more precise control at times of high control cost.

Dual control has regained attention over the past few years, not only in combination with MPC [27], but also in the use for building control [151]. However, most of these methods are relying on explicit dual control techniques [44], where the cost function is modified in a way to enforce exploration. In contrast, we favor an implicit dual control formulation [141], where the value of information emerges from the approximation to optimal dual control on the augmented system (see Chapter 7 for an overview of the overall method).

[27] Cheng, Haghighat, and Di Cairano, "Robust Dual Control MPC with Application to Soft-landing Control," 2015

[151] Zacekova et al., "Dual Control Approach for Zone Model Predictive Control," 2013

[44] Filatov and Unbehauen, "Survey of Adaptive Dual Control Methods," 2000

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

In its classic form, the framework by Tse and Bar-Shalom [141] is not able to address aspects central to many modern control problems: nonlinear dynamics, constraints, and non-quadratic cost functions. In Chapter 8, we extended the approximate dual control framework to nonlinear systems. In this chapter, we additionally address systems with constraints and non-quadratic cost functions. Dual control is particularly beneficial for systems with economic (linear) cost, since it can exploit time-varying cost structures to optimally identify the latent parameters of dynamical systems.

After introducing the problem setting (Section 9.1), we provide a procedure for using the method for economic cost and constrained systems using hierarchical tracking MPC and soft constraints (Section 9.2). We apply the proposed technique to a simple building control problem and analyze the performance with respect to passively learning methods as well as simplistic dual control (Section 9.3).

9.1 Problem Statement

We consider the continuous-time system

$$\dot{x}(t_k) = f(x(t_k), u(t_k), w(t_k)) + \xi(t_k), \tag{9.1}$$

with state $x \in \mathbb{R}^{n_x}$, input $u \in \mathbb{R}^{n_u}$, disturbance $w \in \mathbb{R}^{n_w}$ and white noise $\xi \in \mathbb{R}^{n_x}$. We assume that the dynamics f are not known, but can be described up to Gaussian uncertainty by a general linear model with linear and nonlinear features ϕ, and an unknown matrix M of appropriate size:

$$\dot{x}(t_k) = M\phi(x(t_k), u(t_k), w(t_k)) + \xi(t_k). \tag{9.2}$$

The linear part of the system can be discretized in numerous ways, but for maximal accuracy while retaining the possibility to directly calculate the Jacobian w. r. t. the parameters, we use element-wise zero-order-hold linearization: Each state dynamics is discretized as scalar differential equation, considering the other states as inputs. Using this method, we arrive at the discretized system

$$x_{t+1} = A_t x_t + B_t u_t + E_t w_t + M_t \phi^n(x_t, u_t, w_t) + \xi_t, \tag{9.3}$$

with time index t, matrices A_t, B_t, E_t of appropriate sizes and Gaussian disturbance $\xi_t \sim \mathcal{N}(0, D)$. The matrix M_t is the result of a first-order Euler forward exponential integrator [57] of the nonlinear features ϕ^n. For simplicity of notation, we subsume the non-zero elements of A_t, B_t, E_t and M_t into a parameter vector θ_t. The system is subject to possibly time-varying state and input constraints $x_t \in \mathbb{X}_t$ and $u_t \in \mathbb{U}_t$, where $\mathbb{X}_t \subset \mathbb{R}^{n_x}$ and $\mathbb{U}_t \subset \mathbb{R}^{n_u}$ are polytopes.

[57] Hochbruck and Ostermann, “Exponential Integrators,” 2010

9.2 Non-Quadratic Cost and Constraints

Classic approximate dual control algorithms were posed in the LQG setting, assuming linear dynamics, quadratic cost and Gaussian noise. In this setting, the optimal CE trajectory and the subsequent perturbation control can be obtained in closed form with dynamic programming because there is a recursive solution for the optimal controller at each time step.

However, many control problems where dual control may have an important impact involve economic costs. An example is the considered application to building control, where the cost is linear (energy prices) and the inputs and states are constrained (bounds on the temperature, heating/cooling limits). In this setting, dynamic programming is computationally expensive, since there is no simple recursive solution to obtain a second-order approximation to the cost.

In order to deal with more general cost structures and constraints, we therefore propose to use a common hierarchical tracking scheme: 1) An economic reference satisfying the constraints is computed using the CE system and standard MPC techniques; 2) the reference is tracked using an approximate dual controller, where state constraints are considered in the form of soft constraints. The details of this scheme are outlined in the following sections.

9.2.1 Economic Reference

The economic reference for the controller is generated by solving a discrete-time MPC problem for the CE system

$$(\mathbf{x}^{\mathrm{ref}},\mathbf{u}^{\mathrm{ref}}) := \arg\min_{\mathbf{x},\mathbf{u}} \quad l_N(x_N) + \sum_{i=0}^{N-1} l_i(x_i,u_i) \tag{9.4a}$$

$$\text{s.t.} \quad x_{i=0} = x_t \tag{9.4b}$$

$$x_{i+1} = A_i x_i + B_i u_i + E_i w_i + M_i \phi(x_i,u_i,w_i) \tag{9.4c}$$

$$x_i \in \mathbb{X}_i \tag{9.4d}$$

$$u_i \in \mathbb{U}_i, \tag{9.4e}$$

where l_i is the stage cost. This nonlinear MPC problem is solved with standard algorithms, depending on the cost structure, e. g., sequential linear programming[1] [17, §10.3].

[1] Originally termed *successive linear programming*, we use “sequential” for consistency.

[17] Bazaraa, Sherali, and Shetty, *Nonlinear Programming: Theory and Algorithms*, 2006

9.2.2 Soft Constraints and Uncertainty

Assuming Gaussian uncertainty, hard constraint satisfaction can not be guaranteed. In order to capture the state constraints when tracking the reference, we introduce soft constraints. For constraints of the form $\mathbb{X}_t := \{x_t \mid P_t x_t \le p_t\}$, these take the form

$$\varepsilon_t(x_t) = \max(P_t x_t - p_t, 0) \tag{9.5a}$$

$$l_t^{\mathrm{c}}(x_t) = \varepsilon_t(x_t)^{\mathsf{T}} Q_t \varepsilon_t(x_t). \tag{9.5b}$$

With the max-operator defined element-wise, ε_t captures the amount of constraint violation, while Q_t penalizes the constraint violation in an cost term that is added to the stage cost considered by the dual controller.

In order to apply the approximate dual control scheme (Section 7.2.4), it would be desirable to marginalize the Gaussian distributed state against the soft constraint penalty function. This calculation is of the form

$$\int_{-\infty}^{\infty} \varepsilon_t(x_t)^\intercal Q_t \varepsilon_t(x_t) \cdot \mathcal{N}(x_t; \hat{x}_t, \Sigma_t) dx_t, \tag{9.6}$$

which has generally no closed-form solution because of the max-operator in the definition of $\varepsilon_t(x_t)$. Only when the mean $\hat{x}_t$ coincides with the constraint boundary there is a closed-form solution, amounting to

$$\int_{\hat{x}_t}^{\infty} (P_t x_t - p_t)^\intercal Q_t (P_t x_t - p_t) \cdot \mathcal{N}(x_t; \hat{x}_t, \Sigma_t) dx_t = \frac{1}{2} \left[(P_t \hat{x}_t - p_t)^\intercal Q_t (P_t \hat{x}_t - p_t) + \operatorname{tr}\{Q_t \Sigma_t\} \right]. \tag{9.7}$$

For all states on the constraint boundary, this means that Gaussian marginalization of the soft constraints is equivalent to an additional quadratic tracking cost

$$\tilde{l}_t^{\mathrm{c}}(x_t) = (x_t - x_t^{\mathrm{ref}})^\intercal \tilde{Q}_t (x_t - x_t^{\mathrm{ref}}) \tag{9.8}$$

with $\tilde{Q}_t = \frac{1}{2} Q_t$. This can now be used to modify the second order approximation of the cost-to-go, which is based on the CE reference trajectory: For states lying on the constraint boundary, the cost term (9.8) is added. For states inside of the constraint boundaries, the additional cost can be reduced to zero, or to a small fraction of Q_t to keep the cost positive definite if no other state cost is applied. This procedure essentially amounts to building a quadratic approximation of the soft constraint function around the deterministic trajectory.

With this approximation, we can capture some of the nonlinear effects of the state constraints and add them to the stage cost of the dual tracking controller

$$l_t^{\mathrm{DC}} = (x_t - x_t^{\mathrm{ref}})^\intercal Q_t^{\mathrm{d}} (x_t - x_t^{\mathrm{ref}}) + (u_t - u_t^{\mathrm{ref}})^\intercal R_t^{\mathrm{d}} (u_t - u_t^{\mathrm{ref}}) + \tilde{l}_t^{\mathrm{c}}(x_t), \tag{9.9}$$

where Q_t^{d} and R_t^{d} are the cost matrices for states and inputs.

9.2.3 Maintaining the Value of Information

With the aforementioned modifications, the series-expansion based approximate dual control scheme presented in Chapters 7 and 8 can be applied to constrained linear programming problems. The basic idea is to obtain an approximation of the cost-to-go by tracking a CE trajectory satisfying constraints and minimizing an economic cost with a stochastic optimal controller. The rest of the procedure remains the same. The additional cost induced by parameter uncertainty and the value of information can thereby be maintained for more general cost functions and polytopic state constraints.

9.3 Experiments

The automatic control of building temperature is a promising application for adaptive control systems. There is high potential for energy savings, hence efficient use of control inputs, such as heating and cooling, is desirable. However, in order to make use of building controllers, a good model is required. With the changes introduced in Section 9.2, we can apply approximate dual control to the building climate control problem, which allows for identifying relevant parts of the model during closed-loop control.

9.3.1 The Building Model

We consider the simplified case of temperature control for a building equipped with a heat pump, a setup motivated by the increasing use of heat pumps in buildings. In this case, electrical energy is the energy source for both heating and cooling. The simplified building model is shown in Figure 54. The model is adapted from [54], [51], [88], and models the temperature in a single room inside a larger building.

[54] Gwerder and Tödtli, “Predictive Control for Integrated Room Automation,” 2005

[51] Gondhalekar, Oldewurtel, and Jones, “Least-restrictive Robust MPC of Periodic Affine Systems With Application to Building Climate Control,” 2010

[88] Maasoumy et al., “Handling Model Uncertainty in Model Predictive Control for Energy Efficient Buildings,” 2014

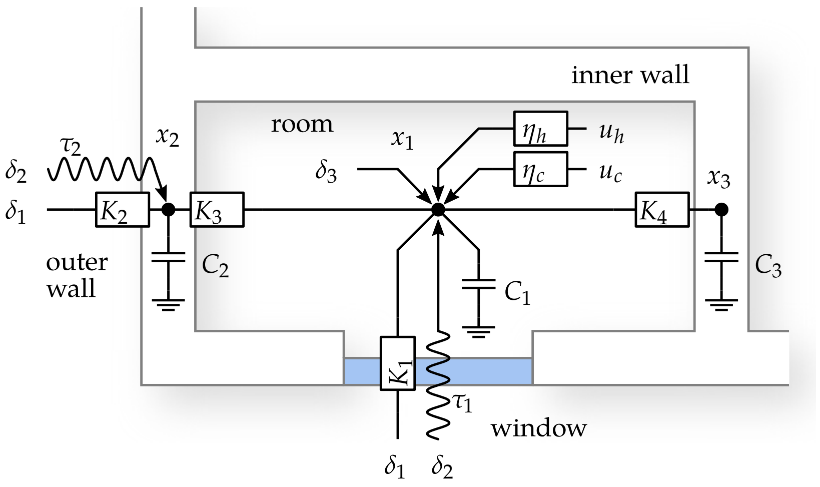


Figure 54: Schematic overview of the building model. The room of which the temperature is to be controlled exchanges heat with the outside air through the window and the outer wall, and with the rest of the building through the inner wall. All symbols are explained in Table 4.

Table 4: Overview of the model states, disturbances and parameters.

symbol	meaning	unit
x_1	room air temperature	[°C]
x_2	exterior wall temperature	[°C]
x_3	interior wall temperature	[°C]
δ_1	outside air temperature	[°C]
δ_2	solar radiation	[kW]
δ_3	internal heat sources	[kW]
u	electrical input power	[kW]
u_h	electrical heating power ($u > 0$)	[kW]
u_c	electrical cooling power ($u < 0$)	[kW]
η_h	heating efficiency	-
η_c	cooling efficiency	-
τ_1	window radiation coefficient	-
τ_2	outer wall radiation coefficient	-
K_{1-4}	heat conductivities	[kW/°C]
C_{1-3}	heat capacities	[kJ/°C]

Table 5: Numerical values of the model parameters.

$C_1 = 0.256 \cdot 10^5$ kJ/°C
$C_2 = 0.970 \cdot 10^6$ kJ/°C
$C_3 = 1.695 \cdot 10^5$ kJ/°C
$\eta_h = 4 \pm 2$
$\eta_c = 2 \pm 2$
$\tau_1 = 0.25$
$\tau_2 = 0.75$
$K_1 = 5$ kW/°C
$K_2 = 23.04$ kW/°C
$K_3 = 30.5$ kW/°C
$K_4 = 122.5$ kW/°C

The linear part of the system is usually relatively easy to identify and is therefore assumed to be known. The input efficiencies η_h and η_c, in contrast, are generally not known, but highly important and only identifiable while the respective inputs are active. For the simulation we consider 50 different buildings with parameters drawn from Gaussian distributions, $\eta_h \sim \mathcal{N}(4, 2)$ and $\eta_c \sim \mathcal{N}(2, 2)$, where we use rejection sampling to limit the range to $\eta_h \in [1, 10]$ and $\eta_c \in [0.35, 5]$.

The continuous-time dynamics of this building model are

$$\dot{x}_1 = \frac{1}{C_1} \left[K_3(x_2 - x_1) + K_1(\delta_1 - x_1) + K_4(x_3 - x_1) \ + \tau_1\delta_2 + \eta_h u_h + \eta_c u_c + \delta_3\right] \tag{9.10a}$$

$$\dot{x}_2 = \frac{1}{C_2} \left[K_2(\delta_1 - x_2) + K_3(x_1 - x_2) + \tau_2\delta_2\right] \tag{9.10b}$$

$$\dot{x}_3 = \frac{1}{C_3} \left[K_4(x_1 - x_3)\right], \tag{9.10c}$$

where all variables and parameters are explained in Table 4, with their numerical value in Table 5. The model is simulated continuously, but state and control cost are defined for the discretized system. The model is simulated for a whole day with a discretization interval of $\Delta t = 600\,\text{s}$.

The input constraints

$$-1000 \leq u_t \leq 1000 \tag{9.11}$$

are imposed at all times, representing the power limitations of the heat pump system. These constraints are chosen to retain feasibility also in the case of poor efficiency (low η_h and/or η_c). The input constraints are enforced through the MPC for generating the nominal trajectory. If the approximate dual controller violates the input constraints, the constraints are enforced by saturation. The state constraints are time-dependent to account for different temperature demands during and outside working hours

$$\mathbb{X}_t = \begin{cases} 21 \leq x_1 \leq 26 & \text{from 08:00 to 18:00} \\ 19 \leq x_1 \leq 30 & \text{otherwise.} \end{cases} \tag{9.12a}$$

The cost on constraint violation is also defined to be time-varying to account for the reduced importance of constraint satisfaction during the night

$$Q_t = \begin{cases} 10^3 & \text{from 08:00 to 18:00} \\ 10^{-1} & \text{otherwise.} \end{cases} \tag{9.13}$$

The energy cost is linear

$$l_t^u(u_t) = r_t^\intercal |u_t|, \tag{9.14}$$

where the prices r_t are based on a day/night pricing, which is a common scheme for electricity used for heating

$$r_t = \begin{cases} 0.025 & \text{from 06:00 to 22:00} \\ 0.010 & \text{otherwise.} \end{cases} \tag{9.15}$$

The overall cost is the sum of energy and constraint cost

$$l_t(x_t, u_t) = l_t^u(u_t) + l_t^c(x_t). \tag{9.16}$$

For the purpose of comparing the different controllers, we assume that accurate predictions of outside air temperature, solar radiation and internal heat gains are known. The disturbance trajectories are shown in Figure 55. Nonetheless, not all simulated days are identical: The mean temperature is drawn from a Gaussian distribution $\delta_0 \sim \mathcal{N}(20, 5)$ for each of the 50 building scenarios to provide a comparison of the controller types at days with different weather conditions.

9.3.2 Controller Types

In order to analyze the performance of the approximate dual controller, we compare it to four other controllers. First of all, an optimal

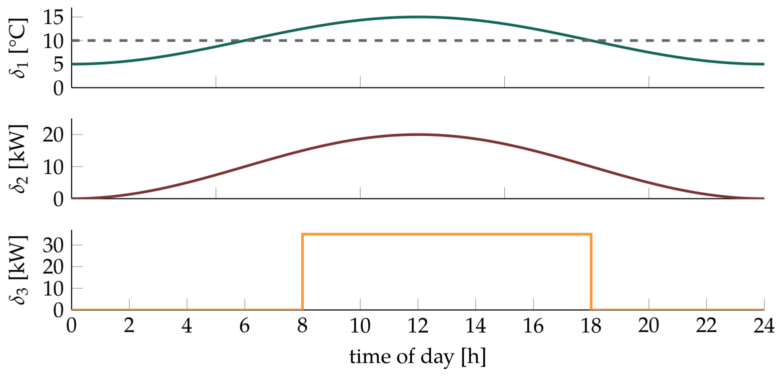


Figure 55: Disturbance trajectories over 24 hours. **Top:** The outside air temperature (——), around the mean temperature $\delta_0 = 10$°C (- - -). **Middle:** The solar radiation. **Bottom:** The internal heat gains.

controller having access to the true parameter values is employed to serve as a lower bound (LB) to the cost for a specific instance of the problem.

The second approach is the CE controller [16], simply using the expectation of the uncertain parameters.

[16] Bar-Shalom and Tse, "Dual Effect, Certainty Equivalence, and Separation in Stochastic Sontrol," 1974

One of the more elaborate options when dealing with parameter uncertainties in MPC is the scenario approach (SA) [24]: Instead of relying on the mean value only, samples from the parameter distribution are used for marginalization. We use a simplified version of the scenario approach, where the MPC is solved for all sampled dynamics individually, averaging the optimal control afterwards. In order to obtain fast and reliable sampling, we use the Latin hypercube sampling technique [99] for this process.

[24] Calafiore and Campi, "The Scenario Approach to Robust Control Design," 2006

[99] McKay, Beckman, and Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," 1979

Since dual control is about the benefits of exploration, we also compare to a controller with modified cost function that favors exploration, also known as exploration bonus (EB) [149, 31, 93, 12]. This approach is often referred to as dual control, but it lacks the selective identification feature. Exploration bonus based controllers can not automatically decide which features of the dynamics are important. As a result, they aim at identifying as much as possible, defined by the trade-off between the arbitrary uncertainty cost and the actual cost. We use an exploration bonus with an additional cost term of the form

[149] Wittenmark, "An Active Suboptimal Dual Controller for Systems with Stochastic Parameters," 1975

[31] Dayan and Sejnowski, "Exploration Bonuses and Dual Control," 1996

[93] Macready and Wolpert, "Bandit Problems and the Exploration/Exploitation Tradeoff," 1998

[12] Audibert, Munos, and Szepesvári, "Exploration-exploitation Tradeoff Using Variance Estimates in Multi-armed Bandits," 2009

$$\operatorname{diag}(\Sigma^{\theta\theta})^\intercal Q^{\text{EB}} \operatorname{diag}(\Sigma^{\theta\theta}). \tag{9.17}$$

The matrix Q^{EB} has to be chosen to encourage exploration, but also not to dominate the certainty equivalent cost structure entirely. In the experiments, we reached this behavior by selecting

$$Q^{\text{EB}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{9.18}$$

The last controller in the comparison is the approximate dual controller (AD) as presented in this chapter.

All controllers, except for the LB, use the element-wise zero-order hold discretization as described in Section 9.1 and all use a horizon length of one day ($T = 144$).

9.3.3 Experimental Results

In order to provide a fair comparison of the different controllers under uncertainty, we sampled 50 different buildings with 50 different weather conditions, as described in Section 9.3.1. For each of these setups, the performance of all five controllers was evaluated. Since the optimal performance even under full knowledge varies tremendously based on the temperature and the heat pump efficiencies, the performances of the tested controllers are also evaluated relative to the lower bound performance. The aggregated results are shown in Table 6. Since the variability due to the different scenarios is high, it is difficult to draw strong general conclusions. Nonetheless it is noticeable that the approximate dual controller shows the best average performance. Relative to the lower bound, the AD shows more than 50% improvement compared to the standard CE approach and about 28% compared to EB.

Figure 56 shows the performance of the different controller types as color-coded entries of the result matrix, visualizing the performance differences. In most cases AD outperforms EB, but in some cases it is the other way round. This is due to the fact that, based on the weather, for certain days only the heating is necessary, for certain days only the cooling, and for some days both.

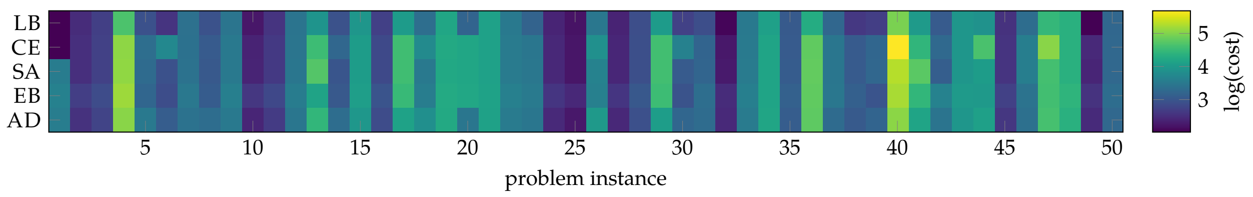


Figure 56: Visual overview of the control performance for 50 different problem instances. The overall cost after one day is color-coded on a log scale. From top to bottom: Lower bound (LB), certainty equivalent (CE), scenario approach (SA), exploration bonus (EB), approximate dual (AD).

	absolute			relative to LB		
	mean	std	SEM	mean	std	SEM
LB	34.45	27.24	3.85	0.00	0.00	0.00
CE	49.44	50.24	7.11	14.99	26.65	3.77
SA	45.15	40.76	5.76	10.70	18.09	2.56
EB	44.60	37.70	5.33	10.15	14.27	2.02
AD	41.75	33.45	4.73	7.30	10.61	1.50

Table 6: Overall performance comparison. Aggregated costs over 50 different problem instances. Controllers as in Figure 56. Provided are the sample mean, sample standard deviation and the standard error of the mean (SEM).

Note that for days where both cooling and heating are used, the EB and AD controllers perform almost equally well, since both input parameters have to be identified. Remaining differences are due to the used approximation and tuning. Figure 57 illustrates such a case (problem instance 23), where the AD has no benefit over the EB. Figure 58, on the other hand, shows a day where only heating, but no cooling, is needed (problem instance 20). This is an example of a situation where it is profitable to use AD instead of EB. Any controller with exploration bonus tries to identify *all* uncertain parameters, whereas the approximate dual controller only identifies the parameters that are important, or *valuable*, in this scenario.

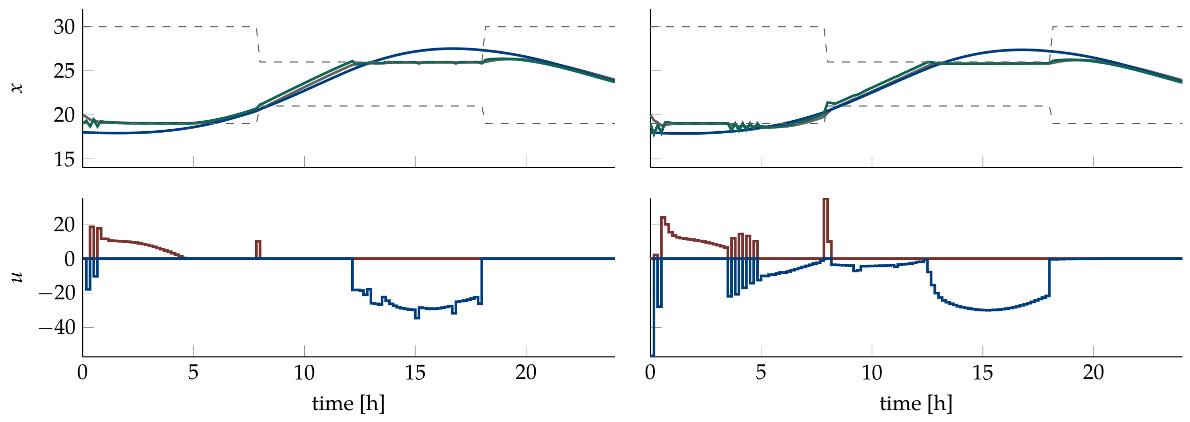


Figure 57: Problem instance where both heating and cooling are used (problem instance 23). **Top:** Room temperature (——), outer wall temperature (——) and inner wall temperature (——). The constraints are imposed on the room temperature (– – –). **Bottom:** Control inputs for heating (——) and cooling (——). **Left:** Exploration bonus controller. **Right:** Dual controller.

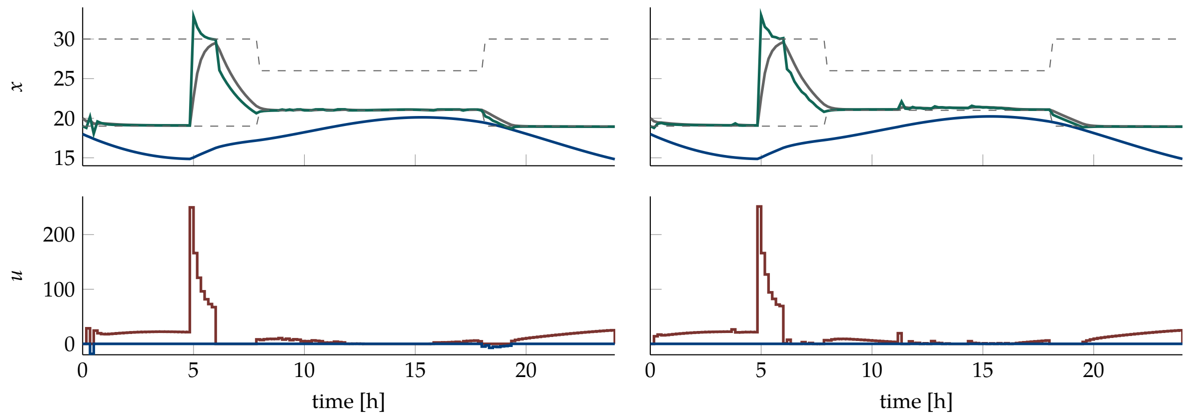


Figure 58: Problem instance where only heating is necessary (problem instance 20). Colors and controllers as in Figure 57. The pre-heating around 5 am is due to the lower energy price at this time.

9.4 Conclusion

The value of information is a feature of dual control often neglected. Using an approximation to the optimal dual control formulation in terms of a series expansion of the cost function, we constructed a controller that maintains an approximation of the value of information in systems with linear cost structure. This controller favors the identification of relevant features and ignores features that are not necessary for future control.

We developed a method based on the construction of a tracking reference found by solving the optimal control problem for the current mean estimate of the parameters. This reference is subsequently tracked by a quadratic low-level dual controller based on dynamic programming.

Since constraint satisfaction can not be guaranteed by the low-level controller under Gaussian assumptions, we used soft constraints with high cost to penalize constraint violation. Further, we proposed a formulation that allows for marginalization of the augmented cost in closed form.

The proposed method combining reference tracking and soft constraint marginalization allows for the approximate evaluation of the value of information. This can be used to increase the average control performance under high initial parameter uncertainty.

In simulation experiments with a simple building model, we illustrate that this method improves performance over simpler alternative approximations to dual control that are based on changes of the cost function.

Epilogue

10 Conclusions and Outlook

The goal of this thesis was to combine regression models from machine learning with discrete-time optimal control methods. More specifically, the work focused on two areas: Using quasiperiodic Gaussian process models for disturbance forecasting and correction, and extending the approximate dual control framework to nonlinear regression methods.

Nonparametric Disturbance Correction

Unlike many other control methods, model-based optimal control enables us to easily use predictions of the environment alongside predictions of the system dynamics to enhance control performance. As accurate models are often difficult to obtain, it is useful to train regression models on the disturbance and use their predictions as disturbance forecast in this setting. Especially promising in this regard are periodic disturbances because the knowledge about their periodicity makes it possible to predict them for a sufficiently long time period into the future.

We developed a framework for the use of quasiperiodic Gaussian processes in model predictive control (Chapter 4). The imperfection of the real world makes it necessary to relax the assumption of a strictly periodic model to a quasiperiodic one. Making use of results from reference tracking model predictive control, the predictions of the periodic disturbance can be incorporated into the controller. This turns extrapolation power into control performance.

We applied the quasiperiodic reference tracking controller to the telescope guiding problem for astronomical imaging (Chapter 5). Periodic prediction models are promising for obtaining high pointing precision in telescopes, since many of them suffer from periodic errors due to revolving gears. We showed on different experiments, both in simulation and on hardware, how the telescope system benefits from the use of a quasiperiodic Gaussian process model.

Building on the promising results from the telescope experiments, we implemented a software solution for the periodic error correction as part of an existing telescope guiding software (Chapter 6). As a robust implementation, capable of working on different telescope setups, has different requirements than a laboratory setup on a single telescope, the algorithm was substantially improved to cope with these requirements.

Nonlinear Dual Control

Dual control can be seen as the ultimate goal of reinforcement learning, since it solves the famous exploration-exploration trade-off optimally—in theory. While many approximate solutions were developed over time, only a few maintain all important aspects of the original idea. One promising example is the approximate dual control approach by quadratic expansion of the cost function, considering uncertainty [141]. However, this method so far was only used for linear systems.

[141] Tse and Bar-Shalom, "An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach," 1973

After giving a modern review of the original method (Chapter 7), we expanded the framework to nonlinear regression frameworks that are frequently used in modern machine learning (Chapter 8). With the help of nonlinear model predictive control and iterative linearization, we developed a nonlinear version of the approximate dual control algorithm that is now capable of acting near-optimally in nonlinear dynamical systems. All features of dual control, especially the value of information, are preserved. This becomes more and more important the larger the model gets. On simulated experiments, we showed that the described method works in different regression settings: parametric nonlinear regression, Gaussian process regression and neural network regression.

When facing practical problems, the quadratic cost structure most dual control approaches are built on is often limiting. Therefore, we developed a method to apply dual control to systems of economic cost structure (Chapter 9). Since the original dual control approximation works for quadratic systems only, we introduced a quadratic reference tracking scheme on top of a linear-cost reference. With this extension, we were able to apply dual control to a classic building control problem, showing a significant improvement in average performance in a simulated, uncertain environment.

Future Directions of Nonlinear Dual Control

With the growing complexity of modern regression models, it is important to be selective about which parts of the model to identify. At a certain level of complexity, it is not sufficient to enforce identification by penalizing uncertainty in the cost function because this approach is likely to invest significant control effort in the identification of irrelevant features of the system. Instead, dual control can guide exploration to those parts of the model that are important.

However, with the existing methods this is only possible to a limited degree. The high computational demand approximate dual control methods are facing poses a challenge—which is prohibitive in many settings. It is desirable to develop methods that can be run at faster time-scales than existing ones, enabling dual control for systems with fast dynamics and challenging sampling times.

While parallelization and increasing computing power will likely alleviate this challenge, they will not necessarily solve it. Therefore, it is important to think of ways to make approximate dual control applicable without sacrificing the important features. One potential way to go is time-scale separation: Long horizons are important for preemptive learning, but they also increase the computational cost. Therefore, the different aspects of dual control could be treated separately, according to their time-scales: While acting cautiously under high uncertainty is important for the low-level execution, the explorative features are not. At the same time, in order to retain the value of information, the exploration needs to be planned ahead much further. This could be accomplished by using larger time steps for exploration planning than for low-level control, so that planning ahead remains computationally feasible.

Another interesting direction is the investigation of alternative approximations derived from different principles than dynamic programming. In the current formulation, the approximate dual control method for nonlinear systems consists of a combination of model predictive control to find a nominal trajectory and dynamic programming to evaluate the cost. This complicates the algorithm and makes it computationally expensive. An improved method based on model predictive control only could potentially simplify and improve the presented approach.

Appendix

A Additional Material

A.1 Reference Tracking Dynamic Programming

We start with the standard discrete-time system

$$x_{t+1} = Ax_t + Bu_t \tag{A.1}$$

with system matrices A and B of appropriate size, and the quadratic reference tracking cost function

$$J(x_0, \mathbf{u}, \mathbf{x}^{\text{ref}}) = \frac{1}{2}(x_T - x_T^{\text{ref}})^\intercal Q_T (x_T - x_T^{\text{ref}}) + \frac{1}{2}\sum_{i=0}^{T-1}(x_i - r_i)^\intercal Q_i (x_i - r_i) + u_i^\intercal R_i u_i, \tag{A.2}$$

where $\mathbf{x}^{\text{ref}} := [x_0^{\text{ref}}, \dots, x_T^{\text{ref}}]$ is the state reference trajectory. Trajectories in input can be added in a similar way, but are not relevant for this thesis.

For the Riccati recursion, the quadratic ansatz has to include the linear terms as well:

$$J_i(x_i) = \frac{1}{2}x_i^\intercal V_i x_i + v_i^\intercal x_i + \text{const}, \tag{A.3}$$

where the remaining parts (depending neither on x_i nor on u_i) are dropped, since they are not relevant for the optimization.

The recursion starts at the last time step T:

$$J_T(x_T) = J_T^*(x_T) = \frac{1}{2}x_T^\intercal Q_T x_T - x_T^{\text{ref}\intercal} Q_T x_T + \text{const} = \frac{1}{2}x_T^\intercal V_T x_T + v_T^\intercal x_T + \text{const}\,. \tag{A.4}$$

In each subsequent recursion step, the stage cost is evaluated and the quadratic optimal cost-to-go is expanded according to the state-transition function $x_{i+1} = Ax_i + Bu_i$. This amounts to

$$\begin{aligned} J_i(x_i, u_i) = \frac{1}{2}x_i^\intercal Q_i x_i - x_i^{\text{ref}\intercal} Q_i x_i + \frac{1}{2}u_i^\intercal R_i u_i + \frac{1}{2}x_i^\intercal A^\intercal V_{i+1} A x_i \\ + \frac{1}{2}u_i^\intercal B^\intercal V_{i+1} B u_i + x_i^\intercal A^\intercal V_{i+1} B u_i + v_{i+1}^\intercal A x_i + v_{i+1}^\intercal B u_i + \text{const}, \end{aligned} \tag{A.5}$$

where we take the derivative to find the optimal input u_i^*:

$$\frac{\partial}{\partial u_i} J_i(x_i, u_i) = u_i^\intercal R_i + u_i^\intercal B^\intercal V_{i+1} B + x_i^\intercal A^\intercal V_{i+1} B + v_{i+1}^\intercal B. \tag{A.6}$$

Equating to zero and solving for u_i results in the optimal solution

$$u_i^*(x_i) = -\left(R_i + B^\intercal V_{i+1} B\right)^{-1}\left(B^\intercal V_{i+1} A x_i + B^\intercal v_{i+1}\right), \tag{A.7}$$

which is the optimal control policy. Inserting (A.7) back into the cost (A.5) to obtain the optimal cost results in

$$J_i^*(x_i) = \frac{1}{2} x_i^\intercal (A^\intercal V_{i+1} A - A^\intercal V_{i+1} B \, (B^\intercal V_{i+1} B + R_i)^{-1} \, B^\intercal V_{i+1} A + Q_i) x_i \\ + \left(v_{i+1}^\intercal A - x_i^{\text{ref}\intercal} Q_i - v_{i+1}^\intercal B \, (B^\intercal V_{i+1} B + R_i)^{-1} \, B^\intercal V_{i+1} A \right) x_i + \text{const}, \quad \text{(A.8)}$$

where we can already read off the recursion for V_i and v_i:

$$V_i = A^\intercal \left(V_{i+1} - V_{i+1} B \, (B^\intercal V_{i+1} B + R_i)^{-1} \, B^\intercal V_{i+1} \right) A + Q_i \qquad V_T = Q_T \qquad \text{(A.9a)}$$

$$v_i = A^\intercal \left(v_{i+1} - V_{i+1} B \, (B^\intercal V_{i+1} B + R_i)^{-1} \, B^\intercal v_{i+1} \right) - Q_i x_i^{\text{ref}} \qquad v_T = -Q_T x_T^{\text{ref}}, \qquad \text{(A.9b)}$$

A.2 Stochastic Dynamic Programming

In order to assess the overall value of a given state under uncertain dynamics, it is important to keep the terms that are usually dropped in the DP calculations. We start with the discrete stochastic system

$$x_{t+1} = A x_t + B u_t + \xi_t \qquad \xi_t \sim \mathcal{N}(0, D) \qquad \text{(A.10a)}$$

$$y_t = C x_t + \gamma_t \qquad \gamma_t \sim \mathcal{N}(0, W), \qquad \text{(A.10b)}$$

with system matrices A, B, and C of appropriate size, and the quadratic cost function

$$J(p(x_0), \mathbf{u}, \mathbf{x}^{\text{ref}}) = \mathbb{E}_{\mathbf{x}} \left[\frac{1}{2} x_T^\intercal Q_T x_T + \frac{1}{2} \sum_{i=0}^{T-1} x_i^\intercal Q_i x_i + u_i^\intercal R_i u_i \right], \quad \text{(A.11)}$$

with state cost matrices Q_i and control cost matrices R_i.

For the Riccati recursion we use the standard quadratic ansatz

$$J_i(p(x_i)) = J_i^*(p(x_i)) = \mathbb{E}_{x_i} \left[\frac{1}{2} x_i^\intercal V_i x_i \right] + \nu_i, \qquad \text{(A.12)}$$

where we choose collect the constant terms outside of the expectation.

The recursion starts at the last time step T:

$$J_T^*(p(x_T)) = \mathbb{E}_{x_T} \left[\frac{1}{2} x_T^\intercal Q_T x_T \right] = \mathbb{E}_{x_T} \left[\frac{1}{2} x_T^\intercal V_T x_T \right] \qquad \text{(A.13)}$$

In each recursion step, the stage cost is evaluated and the quadratic optimal cost-to-go is expanded according to the state-transition function $x_{i+1} = A x_i + B u_i + \xi_i$. This amounts to

$$J_i(p(x_i), u_i) = \mathbb{E}_{x_i, \xi_i} \left[\frac{1}{2} x_i^\intercal Q_i x_i + \frac{1}{2} u_i^\intercal R_i u_i + \frac{1}{2} x_i^\intercal A^\intercal V_{i+1} A x_i + \frac{1}{2} u_i^\intercal B^\intercal V_{i+1} B u_i + \frac{1}{2} \xi_i^\intercal V_{i+1} \xi_i \right. \\ \left. + x_i^\intercal A^\intercal V_{i+1} B u_i + x_i^\intercal A^\intercal V_{i+1} \xi_i + u_i^\intercal B^\intercal V_{i+1} \xi_i \right], \quad \text{(A.14)}$$

which we can simplify by using $\xi_i \sim \mathcal{N}(0, D)$ and $p(x_i) = \mathcal{N}(x_i; \hat{x}_i, \Sigma_{i|i})$, where the belief over x_i is generated by a Kalman filter, to

$$J_i(p(x_i), u_i) = \frac{1}{2}\hat{x}_i^\intercal Q_i \hat{x}_i + \frac{1}{2}u_i^\intercal R_i u_i + \frac{1}{2}\hat{x}_i^\intercal A^\intercal V_{i+1} A \hat{x}_i + \frac{1}{2}u_i^\intercal B^\intercal V_{i+1} B u_i + \hat{x}_i^\intercal A^\intercal V_{i+1} B u_i \\ + \frac{1}{2}\operatorname{tr}\left[Q_i \Sigma_{i|i}\right] + \frac{1}{2}\operatorname{tr}\left[A^\intercal V_{i+1} A \Sigma_{i|i}\right] + \frac{1}{2}\operatorname{tr}\left[V_{i+1} D\right], \quad \text{(A.15)}$$

where the last two summands can be reformulated to $\operatorname{tr}[(A\Sigma_{i|i}A^\intercal + D)V_{i+1}] = \operatorname{tr}[\Sigma_{i+1|i}V_{i+1}]$ by noting that cyclic permutations are allowed inside of the trace.

Taking the derivative to find the optimal input u_i^* results in the same calculations as for the non-stochastic case because the cost of uncertainty does not depend on the input u_i:

$$\frac{\partial}{\partial u_i} J_i(p(x_i), u_i) = u_i^\intercal R_i + u_i^\intercal B^\intercal V_{i+1} B + \hat{x}_i^\intercal A^\intercal V_{i+1} B \quad \text{(A.16)}$$

Equating to zero and solving for u_i results in the optimal solution

$$u_i^*(p(x_i)) = -\left(R_i + B^\intercal V_{i+1} B\right)^{-1} B^\intercal V_{i+1} A \hat{x}_i, \quad \text{(A.17)}$$

which is the optimal control policy. Inserting the optimal policy back into the cost gives

$$J_i^*(p(x_i)) = \frac{1}{2}\hat{x}_i^\intercal \left(Q_i + A^\intercal V_{i+1} A - A^\intercal V_{i+1} B \left(R_i + B^\intercal V_{i+1} B\right)^{-1} B^\intercal V_{i+1} A\right)\hat{x}_i \\ + \frac{1}{2}\operatorname{tr}\left[Q_i \Sigma_{i|i}\right] + \frac{1}{2}\operatorname{tr}\left[\Sigma_{i+1|i} V_{i+1}\right], \quad \text{(A.18)}$$

which we can simplify to

$$J_i^*(p(x_i)) = \frac{1}{2}\hat{x}_i^\intercal V_i \hat{x}_i + \frac{1}{2}\operatorname{tr}\left[Q_i \Sigma_{i|i}\right] + \frac{1}{2}\operatorname{tr}\left[\Sigma_{i+1|i} V_{i+1}\right] \quad \text{(A.19)}$$

$$= \mathbb{E}_{x_i}\left[\frac{1}{2}x_i^\intercal V_i x_i\right] - \frac{1}{2}\operatorname{tr}\left[\Sigma_{i|i} V_i\right] + \frac{1}{2}\operatorname{tr}\left[Q_i \Sigma_{i|i}\right] + \frac{1}{2}\operatorname{tr}\left[\Sigma_{i+1|i} V_{i+1}\right], \quad \text{(A.20)}$$

using the usual Riccati recursion

$$V_i = Q_i + A^\intercal V_{i+1} A - A^\intercal V_{i+1} B \left(R_i + B^\intercal V_{i+1} B\right)^{-1} B^\intercal V_{i+1} A. \quad \text{(A.21)}$$

If we initialize the Riccati recursion with a modified, but equivalent, cost for the last time step

$$J_T^*(p(x_T)) = \mathbb{E}_{x_T}\left[\frac{1}{2}x_T^\intercal V_T x_T\right] + \frac{1}{2}\operatorname{tr}\left[\Sigma_{T|T} V_T\right] - \frac{1}{2}\operatorname{tr}\left[\Sigma_{T|T} V_T\right], \quad \text{(A.22)}$$

we obtain the value function as

$$J_0^*(p(x_0)) = \mathbb{E}_{x_0}\left[\frac{1}{2}x_0^\intercal V_0 x_0\right] + \frac{1}{2}\operatorname{tr}\left\{\sum_{j=0}^{T-1}\left[\Sigma_{j+1|j} - \Sigma_{j+1|j+1}\right] V_{j+1}\right\} + \frac{1}{2}\operatorname{tr}\left\{\sum_{j=0}^{T} Q_j \Sigma_{j|j}\right\}. \quad \text{(A.23)}$$

A.3 Derivation of the Nonparametric EKF

The standard Kalman filter (KF) [64] can be found in many textbooks, e. g., [122, §4.3]. We consider a linear autonomous system

[64] Kálmán, "A New Approach to Linear Filtering and Prediction Problems," 1960

[122] Särkkä, *Bayesian Filtering and Smoothing*, 2013

$$x_{t+1} = A_t x_t + \xi_t \tag{A.24}$$
$$y_t = C x_t + \gamma_t \tag{A.25}$$

where we assume time-varying dynamics A_t, time-invariant measurement C, Gaussian disturbance $\xi_t \sim \mathcal{N}(0, D)$ and noise $\gamma_t \sim \mathcal{N}(0, W)$. In the standard formulation, the Kalman filter maintains a belief over the state $p(x_t) = \mathcal{N}(x_t; m_t, P_t)$ by prediction and update steps as follows:

$$\bar{m}_{t+1} = A_t m_t \tag{A.26}$$
$$\bar{P}_{t+1} = A_t P_t A_t^\mathsf{T} + D \tag{A.27}$$
$$m_{t+1} = \bar{m}_{t+1} + \bar{P}_{t+1} C^\mathsf{T} \left(C\bar{P}_{t+1}C^\mathsf{T} + R\right)^{-1} \left(y_{t+1} - C\bar{m}_{t+1}\right) \tag{A.28}$$
$$P_{t+1} = \bar{P}_{t+1} - \bar{P}_{t+1} C^\mathsf{T} \left(C\bar{P}_{t+1}C^\mathsf{T} + R\right)^{-1} C\bar{P}_{t+1}, \tag{A.29}$$

where $\bar{m}_{t+1}$, $\bar{P}_{t+1}$ denote the predicted belief, and m_{t+1}, P_{t+1} denote the updated belief after measuring y_{t+1}.

Starting from the standard equations, we derive a general multi-step formulation of the classic KF. From there, state augmentation with an infinite-dimensional weight vector gives the expected result.

A.3.1 Derivation of the Multi-Step KF Formulation

Assuming that the result of the KF and the Gaussian process framework should be identical under certain circumstances, we wish to transform the KF to a formulation with full Gram matrix. Therefore, the prediction and update steps have to be combined to

$$P_1 = \left(A_0 P_0 A_0^\mathsf{T} + D\right) - \left(A_0 P_0 A_0^\mathsf{T} + D\right) C^\mathsf{T} S_1^{-1} C \left(A_0 P_0 A_0^\mathsf{T} + D\right) \tag{A.30a}$$
$$S_1 = C\left(A_0 P_0 A_0^\mathsf{T} + D\right) C^\mathsf{T} + W. \tag{A.30b}$$

The same can be done for the second time step, but it is beneficial to introduce a compact notation for the predictive covariance first:

$$\begin{aligned}
&\left(A_1 P_1 A_1^\mathsf{T} + D\right) \\
&= \left(A_1 \left[\left(A_0 P_0 A_0^\mathsf{T} + D\right) - \left(A_0 P_0 A_0^\mathsf{T} + D\right) C^\mathsf{T} S_1^{-1} C \left(A_0 P_0 A_0^\mathsf{T} + D\right)^\mathsf{T}\right] A_1^\mathsf{T} + D\right) \\
&= \underbrace{A_1 \left(A_0 P_0 A_0^\mathsf{T} + D\right) A_1^\mathsf{T} + D}_{:=g_{11}} - \underbrace{A_1 \left(A_0 P_0 A_0^\mathsf{T} + D\right)}_{:=g_{10}} C^\mathsf{T} (C \underbrace{\left(A_0 P_0 A_0^\mathsf{T} + D\right)}_{:=g_{00}} C^\mathsf{T} + W)^{-1} C \underbrace{\left(A_0 P_0 A_0^\mathsf{T} + D\right) A_1^\mathsf{T}}_{:=g_{01}} \\
&= g_{11} - g_{10} C^\mathsf{T} \left(C g_{00} C^\mathsf{T} + W\right)^{-1} C g_{01}.
\end{aligned} \tag{A.31}$$

Using the compact notation and defining S_2 analogously to S_1, we can write the two-step update as

$$P_2 = \left(g_{11} - g_{10}C^\mathsf{T}S_1^{-1}Cg_{01}\right) - \left(g_{11} - g_{10}C^\mathsf{T}S_1^{-1}Cg_{01}\right)C^\mathsf{T}S_2^{-1}C\left(g_{11} - g_{10}C^\mathsf{T}S_1^{-1}Cg_{01}\right) \tag{A.32}$$

$$\begin{aligned}&= g_{11} - g_{10}C^\mathsf{T}S_1^{-1}Cg_{01} - g_{11}C^\mathsf{T}S_2^{-1}Cg_{11} - g_{10}C^\mathsf{T}S_1^{-1}Cg_{01}C^\mathsf{T}S_2^{-1}Cg_{10}C^\mathsf{T}S_1^{-1}Cg_{01}\\ &\qquad + g_{11}C^\mathsf{T}S_2^{-1}Cg_{10}C^\mathsf{T}S_1^{-1}Cg_{01} + g_{10}C^\mathsf{T}S_1^{-1}Cg_{01}C^\mathsf{T}S_2^{-1}Cg_{11}\end{aligned} \tag{A.33}$$

$$= g_{11} - \begin{bmatrix} g_{10}C^\mathsf{T} & g_{11}C^\mathsf{T}\end{bmatrix} \underbrace{\begin{bmatrix} S_1^{-1} + S_1^{-1}Cg_{01}C^\mathsf{T}S_2^{-1}Cg_{10}C^\mathsf{T}S_1^{-1} & -S_1^{-1}Cg_{01}C^\mathsf{T}S_2^{-1} \\ -S_2^{-1}Cg_{10}C^\mathsf{T}S_1^{-1} & S_2^{-1}\end{bmatrix}}_{:=G^{-1}} \begin{bmatrix} Cg_{01} \\ Cg_{11}\end{bmatrix}. \tag{A.34}$$

Application of Schur's lemma gives

$$G = \begin{bmatrix} Cg_{00}C^\mathsf{T} + W & Cg_{01}C^\mathsf{T} \\ Cg_{10}C^\mathsf{T} & Cg_{11}C^\mathsf{T} + W\end{bmatrix}. \tag{A.35}$$

Assuming full state measurement ($C = I$) for compactness of notation, the two-step update is

$$\begin{aligned}P_2 &= g_{11} - \begin{bmatrix} g_{10}^\mathsf{T} & g_{11}^\mathsf{T}\end{bmatrix} \begin{bmatrix} g_{00} + W & g_{01} \\ g_{10} & g_{11} + W\end{bmatrix}^{-1} \begin{bmatrix} g_{01} \\ g_{11}\end{bmatrix}\\ &= A_1\left(A_0P_0A_0^\mathsf{T} + D\right)A_1^\mathsf{T} + D - \begin{bmatrix} A_1\left(A_0P_0A_0^\mathsf{T} + D\right) & \left(A_1\left(A_0P_0A_0^\mathsf{T} + D\right)A_1^\mathsf{T} + D\right)\end{bmatrix}\\ &\cdot \begin{bmatrix} \left(A_0P_0A_0^\mathsf{T} + D\right) + W & \left(A_0P_0A_0^\mathsf{T} + D\right)A_1^\mathsf{T} \\ A_1\left(A_0P_0A_0^\mathsf{T} + D\right) & \left(A_1\left(A_0P_0A_0^\mathsf{T} + D\right)A_1^\mathsf{T} + D\right) + W\end{bmatrix}^{-1} \begin{bmatrix} \left(A_0P_0A_0^\mathsf{T} + D\right)A_1^\mathsf{T} \\ \left(A_1\left(A_0P_0A_0^\mathsf{T} + D\right)A_1^\mathsf{T} + D\right)\end{bmatrix},\end{aligned} \tag{A.36}$$

which already looks similar to GP inference. We can now generalize this two-step result to the general form by building the Gram matrix according to

$$G = \mathcal{A}\mathcal{P}\mathcal{A}^\mathsf{T} + \mathcal{A}\mathcal{D}\mathcal{A}^\mathsf{T} + \mathcal{W}, \tag{A.37}$$

where the individual parts are

$$\mathcal{P} = \begin{bmatrix} A_0P_0A_0^\mathsf{T} & 0 \\ 0 & 0\end{bmatrix} \qquad \mathcal{D} = (D \otimes I) \qquad \mathcal{W} = (W \otimes I) \tag{A.38}$$

of appropriate size, depending on the current time t. The multi-step state transition matrix

$$\mathcal{A} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ A_1 & I & 0 & \cdots & 0 \\ A_2A_1 & A_2 & I & \cdots & 0 \\ \vdots & & & \ddots & \\ A_{t-1}\cdots A_1 & A_{t-1}\cdots A_2 & \cdots & & I \end{bmatrix} \tag{A.39}$$

is needed to shift the initial covariance and drift covariances through time. Put together, this results in

$$P_t = \mathcal{A}_{t-1,:}(\mathcal{P}+\mathcal{D})\mathcal{A}_{t-1,:}^{\intercal} - \mathcal{A}_{t-1,:}(\mathcal{P}+\mathcal{D})\mathcal{A}(\mathcal{A}\mathcal{P}\mathcal{A}^{\intercal} + \mathcal{A}\mathcal{D}\mathcal{A}^{\intercal} + \mathcal{W})^{-1}\mathcal{A}^{\intercal}(\mathcal{P}+\mathcal{D})\mathcal{A}_{t-1,:}^{\intercal}. \tag{A.40}$$

A more compact notation can be achieved by using $\mathcal{A}^{-1}$ to obtain

$$P_t = \mathcal{A}_{t-1,:}(\mathcal{P}+\mathcal{D})\mathcal{A}_{t-1,:}^{\intercal} - \mathcal{A}_{t-1,:}(\mathcal{P}+\mathcal{D})(\mathcal{P}+\mathcal{D}+\mathcal{A}^{-1}\mathcal{W}\mathcal{A}^{-\intercal})^{-1}(\mathcal{P}+\mathcal{D})\mathcal{A}_{t-1,:}^{\intercal}. \tag{A.41}$$

Calculating the mean prediction is done analogously:

$$m_t = \mathcal{A}_{t-1,0}A_0m_0 + \mathcal{A}_{t-1,:}(\mathcal{P}+\mathcal{D})(\mathcal{P}+\mathcal{D}+\mathcal{A}^{-1}\mathcal{W}\mathcal{A}^{-\intercal})^{-1}(\mathbf{y}_{1:t} - \mathcal{C}\mathcal{A}_{:,0}A_0m_0), \tag{A.42}$$

with $\mathcal{C} = (C \otimes I)$ of appropriate size.

A.3.2 Augmenting the State

Instead of tracking only the state covariance, in the GP setting also the dynamics function has to be inferred. The system equations of the nonlinear system are now

$$x_t = f(x_{t-1}) + \xi_{t-1} \qquad \xi_{t-1} \sim \mathcal{N}(0, D) \tag{A.43a}$$
$$y_t = Cx_t + \gamma_t \qquad \gamma_t \sim \mathcal{N}(0, W), \tag{A.43b}$$

where $f \sim \mathcal{GP}(0, k)$. The inference in this model can be done through the EKF with augmented state. We adopt the weight-space view with $f(x) = \Phi(x)\theta$ [115, §2.1] to augment the state with the infinite-dimensional weight vector θ:

[115] Rasmussen and Williams, *Gaussian Processes for Machine Learning*, 2006

$$z = \begin{pmatrix} x \\ \theta \end{pmatrix} \qquad \Sigma = \begin{pmatrix} P & \Sigma^{x\theta} \\ \Sigma^{\theta x} & \Sigma^{\theta\theta} \end{pmatrix} \qquad J_A = \begin{pmatrix} \frac{\partial \bar{f}}{\partial x} & \frac{\partial \bar{f}}{\partial \theta} \\ 0 & I \end{pmatrix} = \begin{pmatrix} A & \Phi \\ 0 & I \end{pmatrix}, \tag{A.44}$$

where, in (A.36), the original x is replaced by the augmented z, P by Σ and A by J_A.

Choosing $C = [I, 0]$, so that C recovers the original states from the augmented state vector, we obtain, after calculations similar to those above, a Gram matrix with additional terms including feature functions and the prior on them:

$$G^\star = G + \begin{pmatrix} \Phi_0 \Sigma_0^{\theta\theta} \Phi_0^\intercal & \Phi_0 \Sigma_0^{\theta\theta} \left(\Phi_0^\intercal A_1^\intercal + \Phi_1^\intercal \right) \\ \left(A_1 \Phi_0 + \Phi_1 \right) \Sigma_0^{\theta\theta} \Phi_0^\intercal & \left(A_1 \Phi_0 + \Phi_1 \right) \Sigma_0^{\theta\theta} \left(\Phi_0^\intercal A_1^\intercal + \Phi_1^\intercal \right) \end{pmatrix}, \tag{A.45}$$

where $\Phi_0 = \Phi(y_0)$, etc. At this point it is important to note that the infinite inner product $\Phi_t \Sigma_0^{\theta\theta} \Phi_{t'}^\intercal$ corresponds to an evaluation of the kernel

$$\Phi(y_t) \Sigma_0^{\theta\theta} \Phi(y_{t'})^\intercal = \sum_i^\infty \phi_i(y_t) \Sigma_{0,ii}^{\theta\theta} \phi_i(y_{t'}) = k(y_t, y_{t'}). \tag{A.46}$$

This means we can write the Gram matrix as

$$G^\star = G + \begin{pmatrix} k_{00} & k_{00} A_1^\intercal + k_{01} \\ A_1 k_{00} + k_{10} & A_1 k_{00} A_1^\intercal + k_{10} A_1^\intercal + A_1 k_{01} + k_{11} \end{pmatrix}, \tag{A.47}$$

where we have written $k_{\cdot\cdot}$ for $k(y_\cdot, y_\cdot)$ to save space. In total, the Gram matrix is then

$$G^\star = \mathcal{A}\mathcal{P}\mathcal{A}^\intercal + \mathcal{A}\mathcal{D}\mathcal{A}^\intercal + \mathcal{A}\mathcal{K}\mathcal{A}^\intercal + \mathcal{W}, \tag{A.48}$$

with $\mathcal{K} = k(\mathbf{y}_{0:t-1}, \mathbf{y}_{0:t-1})$. Since inference is more compact and numerically stable if we absorb $\mathcal{A}$ into the Gram matrix as in Equation (A.41), we define

$$\mathcal{G} = \mathcal{P} + \mathcal{D} + \mathcal{K} + \mathcal{A}^{-1} \mathcal{W} \mathcal{A}^{-\intercal}. \tag{A.49}$$

Inference is done according to

$$P_t = \mathcal{A}_{t-1,:}(\mathcal{P} + \mathcal{D} + \mathcal{K})\mathcal{A}_{t-1,:}^\intercal - \mathcal{A}_{t,:}(\mathcal{P} + \mathcal{D} + \mathcal{K})\mathcal{G}^{-1}(\mathcal{P} + \mathcal{D} + \mathcal{K})\mathcal{A}_{t,:}^\intercal \tag{A.50a}$$

$$\Sigma_t^{x\theta} = \left(\Sigma_t^{\theta x} \right)^\intercal = \mathcal{A}_{t-1,:}\Phi(\mathbf{y}_{0:t-1})\Sigma_0^{\theta\theta} - \mathcal{A}_{t-1,:} \left(\mathcal{P} + \mathcal{K} + \mathcal{D} \right) \mathcal{G}^{-1} \Phi(\mathbf{y}_{0:t-1}) \Sigma_0^{\theta\theta} \tag{A.50b}$$

$$\Sigma_t^{\theta\theta} = \Sigma_0^{\theta\theta} - \Sigma_0^{\theta\theta} \Phi(\mathbf{y}_{0:t-1})^\intercal \mathcal{G}^{-1} \Phi(\mathbf{y}_{0:t-1}) \Sigma_0^{\theta\theta} \tag{A.50c}$$

for the covariance and

$$m_t = \mathcal{A}_{t-1,0} m_0 + \mathcal{A}_{t-1,:}(\mathcal{P} + \mathcal{D} + \mathcal{K})\mathcal{G}^{-1}(\mathbf{y}_{1:t} - \mathcal{C}\mathcal{A}_{:,0} A_0 m_0) \tag{A.51a}$$

$$\hat{\theta}_t = \Sigma_0^{\theta\theta} \Phi(\mathbf{y}_{0:t-1})^\intercal \mathcal{G}^{-1} (\mathbf{y}_{1:t} - \mathcal{C}\mathcal{A}_{:,0} A_0 m_0) \tag{A.51b}$$

for the mean.

A.4 Gradients and Hessians of Dynamics Functions

A.4.1 Neural Network Basis Functions

The neural network dynamics function is

$$f(x) = \sum_{i=1}^{F} v_i \sigma(w_i(x - b_i)), \qquad \sigma(a) = \frac{1}{1 + e^{-a}}, \tag{A.52}$$

with the well-known derivatives of the logistic

$$\frac{\partial}{\partial a}\sigma(a) = \sigma(a)(1 - \sigma(a)), \qquad \frac{\partial^2}{\partial a^2}\sigma(a) = \sigma(a)\left(1 - \sigma(a)\left(3 - 2\sigma(a)\right)\right). \tag{A.53}$$

The gradient of $f(x)$ is

$$\nabla f(x) = \begin{bmatrix} \sum_{i=1}^{F} v_i w_i \sigma(w_i(x - b_i))(1 - \sigma(w_i(x - b_i))) \\ \sigma(w_1(x - b_1)) \\ \vdots \\ \sigma(w_F(x - b_F)) \\ v_1(x - b_1)\sigma(w_1(x - b_1))(1 - \sigma(w_1(x - b_1))) \\ \vdots \\ v_F(x - b_F)\sigma(w_F(x - b_F))(1 - \sigma(w_F(x - b_F))) \end{bmatrix}. \tag{A.54}$$

The Hessian, written in parts, using $a_i = w_i x + b_i$, is:

$$\nabla_x^2 f(x) = \sum_{i=1}^{F} v_i w_i^2 \sigma(a_i)(1 - \sigma(a_i)(3 - 2\sigma(a_i))) \tag{A.55a}$$

$$\nabla_x \nabla_{v_i} f(x) = w_i \sigma(a_i)(1 - \sigma(a_i)) \tag{A.55b}$$

$$\nabla_x \nabla_{w_i} f(x) = v_i \sigma(a_i)(1 - \sigma(a_i)) + (x - b_i) w_i v_i \sigma(a_i)(1 - \sigma(a_i))((1 - 2\sigma(a_i)) \tag{A.55c}$$

$$\nabla_{v_i} \nabla_{v_i} f(x) = 0 \tag{A.55d}$$

$$\nabla_{v_i} \nabla_{w_i} f(x) = (x - b_i)\sigma(a_i)(1 - \sigma(a_i)) \tag{A.55e}$$

$$\nabla_{w_i} \nabla_{w_i} f(x) = v_i (x - b_i)^2 \sigma(a_i)(1 - \sigma(a_i)(3 - 2\sigma(a_i))). \tag{A.55f}$$

A.4.2 Fourier Basis Functions

The Fourier approximation to the dynamics function has the form

$$f(x) = \sqrt{\frac{2}{F}} \sum_{i=1}^{F/2} v_{2i-1} \sin(\omega_{2i-1} x) + v_{2i} \cos(\omega_{2i} x). \tag{A.56}$$

The gradient of $f(x)$ is

$$\nabla f(x) = \begin{bmatrix} \sqrt{\frac{2}{F}} \sum_{i=1}^{F/2} v_{2i-1}\omega_{2i-1}\cos(\omega_{2i-1}x) - v_{2i}\omega_{2i}\sin(\omega_{2i}x) \\ \sqrt{\frac{2}{F}}\sin(\omega_1 x) \\ \sqrt{\frac{2}{F}}\cos(\omega_2 x) \\ \vdots \\ \sqrt{\frac{2}{F}}\sin(\omega_{F-1} x) \\ \sqrt{\frac{2}{F}}\cos(\omega_F x) \end{bmatrix}. \tag{A.57}$$

The Hessian, written in parts, using $c = \sqrt{2/F}$ for normalization, is:

$$\nabla_x^2 f(x) = c\sum_{i=1}^{F/2} -v_{2i-1}\omega_{2i-1}^2\sin(\omega_{2i-1}x) - v_{2i}\omega_{2i}^2\cos(\omega_{2i}x) \tag{A.58a}$$

$$\nabla_x\nabla_{v_i} f(x) = \begin{cases} c\omega_i\cos(\omega_i x) & i \text{ odd} \\ -c\omega_i\sin(\omega_i x) & i \text{ even} \end{cases} \tag{A.58b}$$

$$\nabla_{v_i}\nabla_{v_i} f(x) = 0. \tag{A.58c}$$

A.4.3 Radial Basis Functions

With radial basis function features, the dynamics function is

$$f(x) = \sum_{i=1}^{F} v_i \exp\left(-\frac{(x-c_i)^2}{2\lambda^2}\right). \tag{A.59}$$

The gradient of $f(x)$ is

$$\nabla f(x) = \begin{bmatrix} \sum_{i=1}^{F} v_i \exp\left(-\frac{(x-c_i)^2}{2\lambda^2}\right)\frac{(c_i - x)}{2\lambda^2} \\ \exp\left(-\frac{(x-c_1)^2}{2\lambda^2}\right) \\ \vdots \\ \exp\left(-\frac{(x-c_F)^2}{2\lambda^2}\right) \end{bmatrix}. \tag{A.60}$$

The Hessian, written in parts, is:

$$\nabla_x^2 f(x) = \sum_{i=1}^{F} v_i \exp\left(-\frac{(x-c_i)^2}{2\lambda^2}\right)\left[\left(\frac{c_i - x}{2\lambda^2}\right)^2 - \frac{1}{\lambda^2}\right] \tag{A.61a}$$

$$\nabla_x\nabla_{v_i} f(x) = \exp\left(-\frac{(x-c_i)^2}{2\lambda^2}\right)\frac{c_i - x}{2\lambda^2} \tag{A.61b}$$

$$\nabla_{v_i}\nabla_{v_i} f(x) = 0 \tag{A.61c}$$

B
Bibliography

[1] R. J. Adler. *The Geometry of Random Fields*. Wiley, 1981 (cit. on p. 100).

[2] C. W. Allen and A. N. Cox. *Allen's Astrophysical Quantities*. Springer, 2000 (cit. on p. 61).

[3] F. Allgöwer, T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright. "Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview." In: *Advances in Control*. Springer, 1999, pp. 391–449 (cit. on pp. 37, 48, 90, 98).

[4] M. Aoki. *Optimization of Stochastic Systems*. Academic Press, 1967 (cit. on pp. 84, 87, 89).

[5] K. B. Ariyur and M. Krstić. *Real-time Optimization by Extremum-Seeking Control*. Wiley, 2003 (cit. on p. 39).

[6] K. J. Åström. *Introduction to Stochastic Control Theory*. Academic Press, 1970 (cit. on p. 83).

[7] K. J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, 1994 (cit. on pp. 39, 41).

[8] K. J. Åström and T. Hägglund. *PID Controllers: Theory, Design and Tuning*. Instrumentation, Systems and Automatic Society, 1995 (cit. on p. 78).

[9] K. J. Åström and B. Wittenmark. "On Self-tuning Regulators." In: *Automatica* 9.2 (1973), pp. 185–199 (cit. on p. 40).

[10] A. Aswani, H. González, S. S. Sastry, and C. Tomlin. "Provably Safe and Robust Learning-Based Model Predictive Control." In: *Automatica* 49.5 (2013), pp. 1216–1226 (cit. on pp. 47, 48).

[11] A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin. "Reducing Transient and Steady State Electricity Consumption in HVAC Using Learning-Based Model-Predictive Control." In: *Proceedings of the IEEE* 100.1 (2012), pp. 240–253 (cit. on p. 109).

[12] J. Y. Audibert, R. Munos, and C. Szepesvári. "Exploration-exploitation Tradeoff Using Variance Estimates in Multi-armed Bandits." In: *Theoretical Computer Science* 410.19 (2009), pp. 1876–1902 (cit. on p. 116).

[13] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2011 (cit. on p. 19).

[14] Y. Bar-Shalom. "Stochastic Dynamic Programming: Caution and Probing." In: *IEEE Transactions on Automatic Control* 26.5 (1981), pp. 1184–1195 (cit. on p. 89).

[15] Y. Bar-Shalom and E. Tse. "Caution, Probing, and the Value of Information in the Control of Uncertain Systems." In: *Annals of Economic and Social Measurement* 5.3 (1976), pp. 323–337 (cit. on pp. 84, 89).

[16] Y. Bar-Shalom and E. Tse. "Dual Effect, Certainty Equivalence, and Separation in Stochastic Sontrol." In: *IEEE Transactions on Automatic Control* 19.5 (1974), pp. 494–500 (cit. on pp. 88, 116).

[17] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. 3rd. Wiley, 2006 (cit. on p. 111).

[18] J. Beish. *Design a German Equatorial Mount for the Planetary Telescope*. Tech. rep. Defense Technical Information Center, 2001 (cit. on p. 61).

[19] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957 (cit. on pp. 27, 87).

[20] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd. Athena Scientific, 2005 (cit. on pp. 27–29, 32, 36, 87, 89, 91).

[21] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. on p. 11).

[22] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004 (cit. on p. 34).

[23] C. G. Broyden. "A New Double-rank Minimization Algorithm." In: *Notices of the American Mathematical Society* 16 (1969), p. 670 (cit. on p. 67).

[24] G. C. Calafiore and M. C. Campi. "The Scenario Approach to Robust Control Design." In: *IEEE Transactions on Automatic Control* 51.5 (2006), pp. 742–753 (cit. on pp. 42, 116).

[25] M. C. Campi, S. Garatti, and M. Prandini. "The Scenario Approach for Systems and Control Design." In: *Annual Reviews in Control* 33.2 (2009), pp. 149–157 (cit. on p. 57).

[26] O. Chapelle and L. Li. "An Empirical Evaluation of Thompson Sampling." In: *Advances in Neural Information Processing Systems (NIPS)*. 2011, pp. 2249–2257 (cit. on p. 83).

[27] Y. Cheng, S. Haghighat, and S. Di Cairano. "Robust Dual Control MPC with Application to Soft-landing Control." In: *American Control Conference (ACC)*. 2015, pp. 3862–3867 (cit. on p. 109).

[28] J. W. Cooley and J. W. Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series." In: *Mathematics of Computation* 19 (1965), pp. 297–301 (cit. on p. 76).

[29] J. L. Crassidis and F. L. Markley. "Predictive Filtering for Nonlinear Systems." In: *Journal of Guidance, Control, and Dynamics* 20.3 (1997), pp. 566–572 (cit. on p. 47).

[30] G. B. Dantzig. "Linear Programming." In: *Operations Research* 50.1 (2002), pp. 42–47 (cit. on p. 27).

[31] P. Dayan and T. J. Sejnowski. "Exploration Bonuses and Dual Control." In: *Machine Learning* 25.1 (1996), pp. 5–22 (cit. on p. 116).

[32] R. Dearden, N. Friedman, and D. Andre. "Model Based Bayesian Exploration." In: *Uncertainty in Artificial Intelligence (UAI)*. 1999, pp. 150–159 (cit. on p. 83).

[33] M. P. Deisenroth and C. E. Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search." In: *International Conference on Machine Learning (ICML)*. 2011, pp. 465–472 (cit. on p. 98).

[34] M. Diehl, H. J. Ferreau, and N. Haverbeke. "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation." In: *Nonlinear Model Predictive Control*. Springer, 2009, pp. 391–417 (cit. on pp. 90, 98).

[35] M. Diehl, H. G. Bock, and J. P. Schlöder. "A Real-time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control." In: *SIAM Journal on control and optimization* 43.5 (2005), pp. 1714–1736 (cit. on p. 37).

[36] A. Domahidi. *FORCES: Fast Optimization for Real-time Control on Embedded Systems*. `http://forces.ethz.ch`. 2012. (Visited on 2016-09-28) (cit. on p. 57).

[37] J. R. Dormand and P. J. Prince. "A Family of Embedded Runge-Kutta Formulae." In: *Journal of Computational and Applied Mathematics* 6.1 (1980), pp. 19–26 (cit. on p. 58).

[38] S. E. Dreyfus. "Some Types of Optimal Control of Stochastic Systems." In: *Journal of the Society for Industrial & Applied Mathematics, Series A: Control* 2.1 (1964), pp. 120–134 (cit. on p. 88).

[39] 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 (ICML)*. 2013, pp. 1166–1174 (cit. on pp. 51, 74).

[40] D. Duvenaud. "Automatic Model Construction with Gaussian Processes." PhD thesis. University of Cambridge, 2014 (cit. on p. 16).

[41] T. Erm and S. Sandrock. "Adaptive Periodic Error Correction for the VLT." In: *Large Ground-based Telescopes*. 2003, pp. 900–909 (cit. on p. 47).

[42] A. A. Fel'dbaum. "Dual Control Theory I-IV." In: *Avtomatika i Telemekhanika* 21.9, 21.11, 22.1, 22.2 (1960–1961), pp. 1240–1249, 1453–1464, 3–16, 129–142 (cit. on pp. 83–85, 87).

[43] N. M. Filatov and H. Unbehauen. *Adaptive Dual Control*. Springer, 2004 (cit. on pp. 39, 84).

[44] N. M. Filatov and H. Unbehauen. "Survey of Adaptive Dual Control Methods." In: *IEEE Proceedings on Control Theory and Applications* 147.1 (2000), pp. 118–128 (cit. on pp. 95, 109).

[45] R. Fletcher. "A New Approach to Variable Metric Algorithms." In: *The Computer Journal* 13.3 (1970), pp. 317–322 (cit. on p. 67).

[46] B. Friedland. *Control System Design: An Introduction to State-space Methods*. Dover, 2005 (cit. on p. 25).

[47] C. F. Gauss. "Nachlass: Theoria Interpolationis Methodo Nova Tractata." In: *Werke*. Vol. 3. Königliche Gesellschaft der Wissenschaften, Göttingen, 1866, pp. 265–330 (cit. on p. 76).

[48] H. Genceli and M. Nikolaou. "New Approach to Constrained Predictive Control with Simultaneous Model Identification." In: *AIChE Journal* 42.10 (1996), pp. 2857–2868 (cit. on p. 84).

[49] T. Glad and L. Ljung. *Control Theory: Multivariable and Nonlinear Methods*. Taylor & Francis, 2000 (cit. on p. 92).

[50] D. Goldfarb. "A Family of Variable Metric Updates Derived by Variational Means." In: *Mathematics of Computation* 24.109 (1970), pp. 23–26 (cit. on p. 67).

[51] R. Gondhalekar, F. Oldewurtel, and C. N. Jones. "Least-restrictive Robust MPC of Periodic Affine Systems With Application to Building Climate Control." In: *IEEE Conference on Decision and Control (CDC)*. 2010, pp. 5257–5263 (cit. on pp. 49, 113).

[52] M. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming, Version 2.1*. `http://cvxr.com/cvx`. 2014. (Visited on 2016-09-28) (cit. on pp. 33, 57).

[53] G. Guennebaud, B. Jacob, et al. *Eigen v3*. `http://eigen.tuxfamily.org`. 2010. (Visited on 2016-09-28) (cit. on p. 72).

[54] M. Gwerder and J. Tödtli. "Predictive Control for Integrated Room Automation." In: *REHVA World Congress for Building Technologies*. 2005 (cit. on pp. 109, 113).

[55] J. Hall, C. E. Rasmussen, and J. M. Maciejowski. "Modelling and Control of Nonlinear Systems Using Gaussian Processes with Partial Model Information." In: *IEEE Conference on Decision and Control (CDC)*. 2012, pp. 5266–5271 (cit. on p. 48).

[56] P. Hennig. "Optimal Reinforcement Learning for Gaussian Systems." In: *Advances in Neural Information Processing Systems (NIPS)*. 2011, pp. 325–333 (cit. on pp. 47, 84, 87).

[57] M. Hochbruck and A. Ostermann. "Exponential Integrators." In: *Acta Numerica* 19 (2010), pp. 209–286 (cit. on p. 110).

[58] D. W. Hogg, M. Blanton, D. Lang, K. Mierle, and S. Roweis. "Automated Astrometry." In: *Astronomical Data Analysis Software and Systems*. 2008, pp. 27–34 (cit. on p. 66).

[59] P. J. Huber. "Robust Estimation of a Location Parameter." In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101 (cit. on p. 78).

[60] D. J. Hughes and O. L. R. Jacobs. "Turn-off, Escape and Probing in Nonlinear Stochastic Control." In: *IFAC Symposium on Adaptive Control*. 1974 (cit. on p. 83).

[61] IBM. *ILOG CPLEX Optimizer*. `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`. 2016. (Visited on 2016-09-28) (cit. on p. 33).

[62] P. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice Hall, 1996 (cit. on p. 43).

[63] O. L. R. Jacobs and J. W. Patchell. "Caution and Probing in Stochastic Control." In: *International Journal of Control* 16.1 (1972), pp. 189–199 (cit. on p. 84).

[64] R. E. Kálmán. "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45 (cit. on pp. 31, 47, 58, 132).

[65] H. J. Kappen. "Optimal Control Theory and the Linear Bellman Equation." In: *Bayesian Time Series Models*. Ed. by D. Barber, A. T. Cemgil, and S. Chiappa. Cambridge University Press, 2011, pp. 363–387 (cit. on p. 87).

[66] E. D. Klenske and P. Hennig. "Dual Control for Approximate Bayesian Reinforcement Learning." In: *Journal of Machine Learning Research* 17.127 (2016). Ed. by M. Opper, pp. 1–30. URL: `http://jmlr.org/papers/v17/15-162.html` (cit. on pp. 6, 149).

[67] E. D. Klenske, P. Hennig, B. Schölkopf, and M. N. Zeilinger. "Approximate Dual Control Maintaining the Value of Information with an Application to Building Control." In: *European Control Conference (ECC)*. 2016, pp. 800–806. DOI: `10.1109/ECC.2016.7810387` (cit. on pp. 6, 149).

[68] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. "Gaussian Process-Based Predictive Control for Periodic Error Correction." In: *IEEE Transactions on Control Systems Technology* 24.1 (2016), pp. 110–121. DOI: `10.1109/TCST.2015.2420629` (cit. on pp. 5, 149).

[69] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. "Nonparametric Dynamics Estimation for Time Periodic Systems." In: *Annual Allerton Conference on Communication, Control, and Computing*. 2013, pp. 486–493. DOI: `10.1109/Allerton.2013.6736564` (cit. on pp. 5, 149).

[70] J. Ko and D. Fox. "GP-BayesFilters: Bayesian Filtering Using Gaussian Process Prediction and Observation Models." In: *Autonomous Robots* 27.1 (2009), pp. 75–90 (cit. on p. 47).

[71] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. "Gaussian Process Model Based Predictive Control." In: *American Control Conference (ACC)*. 2004, pp. 2214–2219 (cit. on pp. 47, 48).

[72] J. Z. Kolter and A. Y. Ng. "Near-Bayesian Exploration in Polynomial Time." In: *International Conference on Machine Learning (ICML)*. 2009, pp. 513–520 (cit. on p. 83).

[73] H. König. *Eigenvalue Distribution of Compact Operators*. Birkhäuser, 1986 (cit. on pp. 20, 99).

[74] B. Kouvaritakis and M. Cannon. "Stochastic Model Predictive Control." In: *Encyclopedia of Systems and Control*. Springer, 2014, pp. 1–9 (cit. on p. 55).

[75] B. Kouvaritakis and M. Cannon. *Non-linear Predictive Control: Theory and Practice*. Institution of Engineering and Technology, 2001 (cit. on p. 37).

[76] P. R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification and Adaptive Control*. Prentice Hall, 1986 (cit. on pp. 3, 83, 86).

[77] I. D. Landau. “A Survey of Model Reference Adaptive Techniques–Theory and Applications.” In: *Automatica* 10.4 (1974), pp. 353–379 (cit. on p. 40).

[78] C. A. Larsson. “Application-oriented Experiment Design for Industrial Model Predictive Control.” PhD thesis. KTH Royal Institute of Technology, 2014 (cit. on p. 84).

[79] A. Laub. “A Schur Method for Solving Algebraic Riccati Equations.” In: *IEEE Transactions on Automatic Control* 24.6 (1979), pp. 913–921 (cit. on p. 30).

[80] J. Lausten. *Energy Efficiency Requirements In Building Codes, Energy Efficiency Policies For New Buildings*. Tech. rep. International Energy Agency, 2008 (cit. on p. 109).

[81] N. D. Lawrence, M. Seeger, and R. Herbrich. “Fast Sparse Gaussian Process Methods: The Informative Vector Machine.” In: *Advances in Neural Information Processing Systems (NIPS)*. 2003, pp. 609–616 (cit. on p. 20).

[82] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. “Sparse Spectrum Gaussian Process Regression.” In: *Journal of Machine Learning Research* 11 (2010), pp. 1865–1881 (cit. on pp. 21, 101).

[83] M. Leblanc. “Sur l’électrification des chemins de fer au moyen de courants alternatifs de fréquence élevée.” In: *Revue générale de l’électricité* 12 (1922), pp. 275–277 (cit. on p. 39).

[84] D. J. Leith and W. E. Leithead. “Survey of Gain-scheduling Analysis and Design.” In: *International journal of control* 73.11 (2000), pp. 1001–1025 (cit. on p. 40).

[85] D. Limon, T. Alamo, D. Muñoz de la Peña, M. N. Zeilinger, C. N. Jones, and M. Pereira. “MPC for Tracking Periodic Reference Signals.” In: *IFAC Nonlinear Model Predictive Control Conference*. 2012, pp. 490–495 (cit. on p. 57).

[86] L. Ljung. *System Identification: Theory for the User*. 2nd. Prentice Hall, 1999 (cit. on pp. 3, 42).

[87] Y. Ma, J. Matusko, and F. Borrelli. “Stochastic Model Predictive Control for Building HVAC Systems: Complexity and Conservatism.” In: *IEEE Transactions on Control Systems Technology* 23.1 (2015), pp. 101–116 (cit. on p. 109).

[88] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. S. Vincentelli. “Handling Model Uncertainty in Model Predictive Control for Energy Efficient Buildings.” In: *Energy and Buildings* 77 (2014), pp. 377–392 (cit. on p. 113).

[89] J. M. Maciejowski and X. Yang. “Fault Tolerant Control Using Gaussian Processes and Model Predictive Control.” In: *Conference on Control and Fault-Tolerant Systems*. 2013, pp. 1–12 (cit. on p. 48).

[90] J. M. Maciejowski. *Predictive Control with Constraints*. Pearson, 2002 (cit. on pp. 32, 35, 36).

[91] D. J. C. MacKay. “Comparison of Approximate Methods for Handling Hyperparameters.” In: *Neural Computation* 11.5 (1999), pp. 1035–1068 (cit. on p. 19).

[92] D. J. C. MacKay. “Introduction to Gaussian Processes.” In: *NATO ASI Series F Computer and Systems Sciences* 168 (1998), pp. 133–166 (cit. on pp. 14, 50).

[93] W. G. Macready and D. H. Wolpert. “Bandit Problems and the Exploration/Exploitation Tradeoff.” In: *IEEE Transactions on Evolutionary Computation* 2.1 (1998), pp. 2–22 (cit. on p. 116).

[94] G. Marafioti, R. R. Bitmead, and M. Hovd. “Persistently Exciting Model Predictive Control.” In: *International Journal of Adaptive Control and Signal Processing* 28.6 (2014), pp. 536–552 (cit. on pp. 84, 91).

[95] J. C. Maxwell. “On Governors.” In: *Proceedings of the Royal Society of London* 16 (1867), pp. 270–283 (cit. on p. 3).

[96] D. Q. Mayne. “A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems.” In: *International Journal of Control* 3.1 (1966), pp. 85–95 (cit. on pp. 85, 90).

[97] D. Q. Mayne. “Model Predictive Control: Recent Developments and Future Promises.” In: *Automatica* 50.12 (2014), pp. 2967–2986 (cit. on p. 55).

[98] O. Mayr. *The Origins of Feedback Control*. MIT Press, 1970 (cit. on p. 3).

[99] M. D. McKay, R. J. Beckman, and W. J. Conover. “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code.” In: *Technometrics* 21.2 (1979), pp. 239–245 (cit. on pp. 21, 116).

[100] L. Meier. *Combined Optimal Control and Estimation Theory*. Tech. rep. NASA Ames Research Center, 1966 (cit. on p. 84).

[101] C. A. Micchelli, Y. Xu, and H. Zhang. “Universal Kernels.” In: *Journal of Machine Learning Research* 7 (2006), pp. 2651–2667 (cit. on pp. 16, 48).

[102] D. H. Nguyen and B. Widrow. “Neural Networks for Self-learning Control Systems.” In: *IEEE Control Systems Magazine* 10.3 (1990), pp. 18–23 (cit. on p. 101).

[103] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. Springer, 2006 (cit. on pp. 18, 34).

[104] K. Ogata. *Discrete-time Control Systems*. Prentice Hall, 1995 (cit. on pp. 26, 55).

[105] K. Ogata. *Modern Control Engineering*. Prentice Hall, 2010 (cit. on p. 25).

[106] F. Oldewurtel, A. Parisio, C. N. Jones, M. Morari, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and K. Wirth. “Energy Efficient Building Climate Control Using Stochastic Model Predictive Control and Weather Predictions.” In: *American Control Conference (ACC)*. 2010, pp. 5100–5105 (cit. on p. 109).

[107] F. Oldewurtel, A. Ulbig, A. Parisio, G. Andersson, and M. Morari. “Reducing Peak Electricity Demand in Building Climate Control Using Real-time Pricing and Model Predictive Control.” In: *IEEE Conference on Decision and Control (CDC)*. 2010, pp. 1927–1932 (cit. on p. 37).

[108] M. Ono and B. C. Williams. “Iterative Risk Allocation: A New Approach to Robust Model Predictive Control with a Joint Chance Constraint.” In: *IEEE Conference on Decision and Control (CDC)*. 2008, pp. 3427–3432 (cit. on p. 57).

[109] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time Signal Processing*. Prentice Hall, 1999 (cit. on p. 76).

[110] G. Parker. *Making Beautiful Deep-sky Images: Astrophotography with Affordable Equipment and Software*. Springer, 2007 (cit. on pp. 61, 62).

[111] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung. “Kernel Methods in System Identification, Machine Learning and Function Estimation: A Survey.” In: *Automatica* 50.3 (2014), pp. 657–682 (cit. on p. 48).

[112] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. "An Analytic Solution to Discrete Bayesian Reinforcement Learning." In: *International Conference on Machine Learning (ICML)*. 2006, pp. 679–704 (cit. on pp. 83, 84, 89).

[113] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines." In: *Advances in Neural Information Processing Systems (NIPS)*. 2008, pp. 1177–1184 (cit. on pp. 21, 101).

[114] H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. Harvard University, 1961 (cit. on p. 93).

[115] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006 (cit. on pp. 11, 12, 15, 18, 19, 23, 51, 53, 74, 99, 100, 134).

[116] J. Rathouský and V. Havlena. "MPC-Based Approximation of Dual Control by Information Maximization." In: *International Conference on Process Control*. 2011, pp. 247–252 (cit. on p. 84).

[117] J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009 (cit. on pp. 32, 55).

[118] H. Robbins and S. Monro. "A Stochastic Approximation Method." In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407 (cit. on p. 101).

[119] J. A. Rossiter. *Model-based Predictive Control: A Practical Approach*. CRC Press, 2003 (cit. on p. 32).

[120] R. Routledge. *Discoveries and Inventions of the Nineteenth Century*. 13th. G. Routledge and Sons, 1900 (cit. on p. 3).

[121] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Representations by Back-propagating Errors." In: *Nature* 323 (1986), pp. 533–536 (cit. on p. 101).

[122] S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013 (cit. on pp. 31, 47, 87, 98, 132).

[123] S. Sastry and M. Bodson. *Adaptive Control: Stability, Convergence and Robustness*. Courier Corporation, 2011 (cit. on p. 39).

[124] G. Schildbach, L. Fagiano, C. Frei, and M. Morari. "The Scenario Approach for Stochastic Model Predictive Control with Bounds on Closed-Loop Constraint Violations." In: *Automatica* 50.12 (2014), pp. 3009–3018 (cit. on p. 57).

[125] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002 (cit. on pp. 13, 20).

[126] J. Schur. "Bemerkungen zur Theorie der Beschränkten Bilinearformen mit Unendlich Vielen Veränderlichen." In: *Journal für die Reine und Angewandte Mathematik* 140 (1911), pp. 1–28 (cit. on p. 15).

[127] A. T. Schwarm and M. Nikolaou. "Chance-constrained Model Predictive Control." In: *AIChE Journal* 45.8 (1999), pp. 1743–1752 (cit. on p. 57).

[128] D. F. Shanno. "Conditioning of Quasi-Newton Methods for Function Minimization." In: *Mathematics of Computation* 24.111 (1970), pp. 647–656 (cit. on p. 67).

[129] J. H. Simonetti. *Measuring the Signal to Noise Ratio of the CCD Image of a Star or Nebula*. Tech. rep. 2004 (cit. on p. 73).

[130] S. Singhal and L. Wu. "Training Multilayer Perceptrons with the Extended Kalman Algorithm." In: *Advances in Neural Information Processing Systems (NIPS)*. 1989, pp. 133–140 (cit. on p. 102).

[131] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design.” In: *International Conference on Machine Learning (ICML)*. 2010, pp. 1015–1022 (cit. on p. 83).

[132] C. Stark, B. McKee, A. Galasso, et al. *PHD2 Guiding*. `http://openphdguiding.org`. 2016. (Visited on 2016-09-28) (cit. on p. 71).

[133] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, 1999 (cit. on p. 21).

[134] I. Steinwart. “On the Influence of the Kernel on the Consistency of Support Vector Machines.” In: *Journal of Machine Learning Research* 2 (2002), pp. 67–93 (cit. on p. 48).

[135] J. Sternby. “A Simple Dual Control Problem with an Analytical Solution.” In: *IEEE Transactions on Automatic Control* 21.6 (1976), pp. 840–844 (cit. on p. 84).

[136] V. Strassen. “Gaussian Elimination is Not Optimal.” In: *Numerische Mathematik* 13.4 (1969), pp. 354–356 (cit. on p. 19).

[137] M. Tanaskovic, L. Fagiano, R. Smith, P. J. Goulart, and M. Morari. “Adaptive Model Predictive Control for Constrained Linear Systems.” In: *European Control Conference (ECC)*. 2013, pp. 382–387 (cit. on p. 47).

[138] J. W. Taylor and P. E. McSharry. “Short-term Load Forecasting Methods: An Evaluation Based on European Data.” In: *IEEE Transactions on Power Systems* 22.4 (2007), pp. 2213–2219 (cit. on p. 49).

[139] M. B. Thompson and R. M. Neal. “Slice Sampling with Adaptive Multivariate Steps: The Shrinking-Rank Method.” In: *preprint at arXiv:1011.4722* (2010) (cit. on p. 53).

[140] E. Todorov and W. Li. “A Generalized Iterative LQG Method for Locally-optimal Feedback Control of Constrained Nonlinear Stochastic Systems.” In: *American Control Conference (ACC)*. 2005, pp. 300–306 (cit. on p. 85).

[141] E. Tse and Y. Bar-Shalom. “An Actively Adaptive Control for Linear Systems with Random Parameters via the Dual Control Approach.” In: *IEEE Transactions on Automatic Control* 18.2 (1973), pp. 109–117 (cit. on pp. 84, 90, 91, 93, 97, 108, 109, 124).

[142] E. Tse, Y. Bar-Shalom, and L. Meier. “Wide-sense Adaptive Dual Control for Nonlinear Stochastic Systems.” In: *IEEE Transactions on Automatic Control* 18.2 (1973), pp. 98–108 (cit. on pp. 84, 90, 94, 97).

[143] J. Uhlmann. “Dynamic Map Building and Localization: New Theoretical Foundations.” PhD thesis. University of Oxford, 1995 (cit. on p. 98).

[144] A. W. van der Vaart and J. H. van Zanten. “Information Rates of Nonparametric Gaussian Process Methods.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2095–2119 (cit. on pp. 16, 48).

[145] A. W. van der Vaart and J. H. van Zanten. “Rates of Contraction of Posterior Distributions Based on Gaussian Process Priors.” In: *The Annals of Statistics* 36.3 (2008), pp. 1435–1463 (cit. on p. 48).

[146] R. J. Vanderbei et al. *Roque de los Muchachos Observatory*. `https://www.princeton.edu/~rvdb/images/other/LaPalma.html`. 2012. (Visited on 2016-09-28) (cit. on p. 61).

[147] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2010 (cit. on p. 18).

[148] B. Wittenmark. “Adaptive Dual Control Methods: An Overview.” In: *IFAC Symposium on Adaptive Systems in Control and Signal Processing*. 1995, pp. 67–72 (cit. on pp. 4, 40, 84).

[149] B. Wittenmark. “An Active Suboptimal Dual Controller for Systems with Stochastic Parameters.” In: *Automatic Control Theory and Application* 3 (1975), pp. 13–19 (cit. on pp. 83, 84, 95, 102, 116).

[150] S. G. Yuen, P. M. Novotny, and R. D. Howe. “Quasiperiodic Predictive Filtering for Robot-assisted Beating Heart Surgery.” In: *IEEE International Conference on Robotics and Automation*. 2008, pp. 3875–3880 (cit. on pp. 47, 49).

[151] E. Zacekova, S. Privara, Z. Vana, J. Cigler, and L. Ferkl. “Dual Control Approach for Zone Model Predictive Control.” In: *European Control Conference (ECC)*. 2013, pp. 1398–1403 (cit. on p. 109).

[152] K. Zhou and J. C. Doyle. *Essentials of Robust Control*. Prentice Hall, 1998 (cit. on pp. 42, 83).

P Publications

E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. "Gaussian Process-Based Predictive Control for Periodic Error Correction." In: *IEEE Transactions on Control Systems Technology* 24.1 (2016), pp. 110–121. DOI: 10.1109/TCST.2015.2420629

E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. "Nonparametric Dynamics Estimation for Time Periodic Systems." In: *Annual Allerton Conference on Communication, Control, and Computing*. 2013, pp. 486–493. DOI: 10.1109/Allerton.2013.6736564

E. D. Klenske and P. Hennig. "Dual Control for Approximate Bayesian Reinforcement Learning." In: *Journal of Machine Learning Research* 17.127 (2016). Ed. by M. Opper, pp. 1–30. URL: http://jmlr.org/papers/v17/15-162.html

E. D. Klenske, P. Hennig, B. Schölkopf, and M. N. Zeilinger. "Approximate Dual Control Maintaining the Value of Information with an Application to Building Control." In: *European Control Conference (ECC)*. 2016, pp. 800–806. DOI: 10.1109/ECC.2016.7810387

Colophon

This document was typeset with LaTeX, using a blend of `classicthesis` developed by André Miede and `tufte-latex`, which is based on Edward Tufte's *Beautiful Evidence*. The design was also inspired by the PhD theses of Aaron Turon and Christian A. Larsson. Most of the graphics in this thesis were generated using `pgfplots` and `pgf/tikz`.