Event-triggered Learning

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M.Sc. Friedrich Solowjow
aus Stepnogorsk, Kasachstan

Tübingen
2022

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	30.08.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Sebastian Trimpe
2. Berichterstatter:	Prof. Dr. Martin Butz
3. Berichterstatter:	Prof. Dr. Thomas Schön

Abstract

Machine learning has seen many recent breakthroughs. Inspired by these, learning-control systems emerged. In essence, the goal is to learn models and control policies for dynamical systems. Dealing with learning-control systems is hard and there are several key challenges that differ from classical machine learning tasks. Conceptually, excitation and exploration play a major role in learning-control systems. On the one hand, we usually aim for controllers that stabilize a system with the goal of avoiding deviations from a setpoint or reference. However, we also need informative data for learning, which is often not the case when controllers work well. Therefore, there is a problem due to the opposing objectives of many control theoretical tasks and the requirements for successful learning outcomes.

Additionally, change of dynamics or other conditions is often encountered for control systems in practice. For example, new tasks, changing load conditions, or different external conditions have a substantial influence on the underlying distribution. Learning can provide the flexibility to adapt the behavior of learning-control systems to these events.

Since learning has to be applied with sufficient excitation there are many practical situations that hinge on the following problem:

When to trigger learning updates in learning-control systems?

This is the core question of this thesis and despite its relevance, there is no general method that provides an answer. We propose and develop a new paradigm for principled decision making on when to learn, which we call event-triggered learning (ETL).

The first triggers that we discuss are designed for networked control systems. All agents use model-based predictions to anticipate the other agents' behavior which makes communication only necessary when the predictions deviate too much. Essentially, an accurate model can save communication, while a poor model leads to poor predictions and thus frequent updates. The learning triggers are based on the inter-communication times (the time between two communication instances). They are independent and identically distributed random variables, which directly leads to sound guarantees. The framework is validated in experiments and leads to 70% communication savings for wireless sensor networks that monitor human walking.

In the second part, we consider optimal control algorithms and start with linear quadratic regulators. A perfect model yields the best possible controller, while poor models result in poor controllers. Thus, by analyzing the control performance, we can infer the model's accuracy. From a technical point of view, we have to deal with correlated data and work with more sophisticated tools to provide the desired theoretical guarantees. While we obtain a powerful test that is tightly tailored to the problem at hand, it does not generalize to different control architectures. Therefore, we also consider a more general point of view, where we recast the learning of linear systems as a filtering problem. We leverage Kalman filter-based techniques to derive a sound test and utilize the point estimate of the parameters for targeted learning experiments. The algorithm is independent of the underlying control architecture, but demonstrated for model predictive control.

Most of the results in the first two parts critically depend on linearity assumptions in the dynamics and further problem-specific properties. In the third part, we take a step back and ask the fundamental question of how to compare (nonlinear) dynamical systems directly from state data. We propose a kernel two-sample test that compares stationary distributions of dynamical systems. Additionally, we introduce a new type of mixing that can directly be estimated from data to deal with the autocorrelations.

In summary, this thesis introduces a new paradigm for deciding when to trigger updates in learning-control systems. Additionally, we develop three instantiations of this paradigm for different learning-control problems. Further, we present applications of the algorithms that yield substantial communication savings, effective controller updates, and the detection of anomalies in human walking data.

Zusammenfassung

Das maschinelle Lernen hat in letzter Zeit viele Durchbrüche erlebt. Inspiriert von diesen sind lernende Steuerungssysteme entstanden. Das Ziel besteht im Wesentlichen darin, Modelle und Steuerungsstrategien für dynamische Systeme zu erlernen. Der Umgang mit lernenden Steuerungssystemen ist schwierig, und es gibt mehrere zentrale Herausforderungen, die sich von klassischen maschinellen Lernaufgaben unterscheiden. Aus konzeptioneller Sicht spielen Anregung und Erkundung bei lernenden Regelsystemen eine große Rolle. Einerseits streben wir in der Regel Regler an, die ein System stabilisieren, mit dem Ziel, Abweichungen von einem Sollwert oder einer Referenz zu vermeiden. Zum Lernen benötigen wir aber auch informative Daten, was bei gut funktionierenden Reglern oft nicht der Fall ist. Ein Problem ergibt sich also aus den gegensätzlichen Zielen vieler steuerungstheoretischer Aufgaben und den Anforderungen an erfolgreiche Lernergebnisse.

Darüber hinaus gibt es auch eine natürliche Möglichkeit für Veränderungen. So haben beispielsweise neue Aufgaben, veränderte Belastungszustände oder andere äußere Einflüsse einen erheblichen Einfluss auf die zugrunde liegende Verteilung. Lernen kann die Flexibilität bieten, das Verhalten von lernenden Steuerungssystemen an diese Ereignisse anzupassen.

Da das Lernen mit ausreichender Anregung erfolgen muss, gibt es viele praktische Situationen, die von dem folgenden Problem abhängen:

> *Wann sollen Lernaktualisierungen in lernenden Kontrollsystemen ausgelöst werden?*

Dies ist die Kernfrage dieser Arbeit und trotz ihrer Relevanz gibt es keine allgemeine Methode, die eine Antwort liefert. Wir schlagen ein neues Paradigma für eine prinzipiengeleitete Entscheidungsfindung über den Zeitpunkt des Lernens vor, das wir ereignisgesteuertes Lernen (ETL) nennen.

Die ersten Auslöser, die wir diskutieren, sind für vernetzte Kontrollsysteme konzipiert. Alle Agenten verwenden modellbasierte Vorhersagen, um das Verhalten der anderen Agenten zu antizipieren, was eine Kommunikation nur dann erforderlich macht, wenn die Vorhersagen zu stark abweichen. Im Wesentlichen kann ein genaues Modell Kommunikation einsparen, während ein schlechtes Modell zu schlechten Vorhersagen und damit zu häufigen Aktualisierungen führt. Die Lernauslöser basieren auf den Inter-Kommunikationszeiten (die Zeit zwischen zwei Kommunikationsinstanzen). Es handelt sich um unabhängige und identisch verteilte Zufallsvariablen, was direkt zu soliden Garantien führt. Der Ansatz wurde in Experimenten validiert und führt zu 70 Prozent Kommunikationseinsparungen für drahtlose Sensornetzwerke, die das Gehen von Menschen überwachen.

Im zweiten Teil betrachten wir optimale Steuerungsalgorithmen und beginnen mit linearen quadratischen Reglern. Ein perfektes Modell ergibt den bestmöglichen Regler, während schlechte Modelle zu schlechten Reglern führen. Durch die Analyse der Regelungsleistung können wir also auf die Genauigkeit des Modells schließen. Aus theoretischer Sicht müssen wir uns mit korrelierten Daten befassen und mit anspruchsvolleren Methoden arbeiten, um die gewünschten Garantien zu erhalten. Wir erhalten zwar einen leistungsfähigen Test, der genau auf das vorliegende Problem zugeschnitten ist, aber er lässt sich nicht auf verschiedene Steuerungsarchitekturen verallgemeinern. Daher betrachten wir auch einen allgemeineren Ansatz, bei dem wir das Lernen von linearen Systemen als ein Filterproblem behandeln. Wir nutzen auf Kalman-Filtern basierende Techniken zur Herleitung eines fundierten Tests und setzen die Punktschätzung der Parameter für gezielte Lernexperimente ein. Der Algorithmus ist unabhängig von der zugrundeliegenden Steuerungsarchitektur, wird aber für die modellprädiktive Steuerung demonstriert.

Die meisten der Ergebnisse in den ersten beiden Teilen hängen entscheidend von Linearitätsannahmen in der Dynamik und weiteren problemspezifischen Eigenschaften ab. Im dritten Teil gehen wir einen Schritt zurück und stellen die grundlegende Frage, wie man (nichtlineare) dynamische Systeme direkt anhand von Zustandsdaten vergleichen kann. Wir schlagen einen Kernel-Zwei-Stichproben-Test vor, der stationäre Verteilungen dynamischer Systeme vergleicht. Außerdem führen wir eine neue Art des Mischens ein, die direkt aus den Daten geschätzt werden kann, um mit den Autokorrelationen umzugehen.

Zusammenfassend wird in dieser Arbeit ein neues Paradigma für die Entscheidung, wann Aktualisierungen in Lernkontrollsystemen ausgelöst werden sollen, vorgestellt. Darüber hinaus entwickeln wir drei Instanziierungen dieses Paradigmas für verschiedene Lernsteuerungsprobleme. Darüber hinaus stellen wir Anwendungen der Algorithmen vor, die erhebliche Kommunikationseinsparungen, effektive Steuerungsaktualisierungen und die Erkennung von Anomalien in menschlichen Gehdaten ermöglichen.

Acknowledgements

First of all, I want to thank my advisor Prof. Dr. Sebastian Trimpe. I am grateful for everything that I have learned from him, all the trust that he has put in me, and for the freedom to pursue my own ideas.

I would also like to thank Prof. Dr. Martin Butz and Prof. Dr. Thomas Schön for examining my thesis, Prof. Dr. Philipp Hennig and Prof. Dr. Hendrik Lensch for being part of my examination committee, and I would like to thank my IMPRS-IS thesis advisory committee members Prof. Dr. Bernhard Schölkopf and Prof. Dr. Ingo Steinwart for valuable feedback and insightful discussions.

My personal thanks goes to: Dr. Steve Heim, for constantly reminding me what science should be about and for being a good friend; Dr. Dominik Baumann, for sharing his dedication for research and hard work with me; Alexander von Rohr, for never saying no to coffee and all the support throughout the years; Dr. Alonso Marco-Valle, for making me feel welcome in Tübingen since my very first day; Dr. Michal Rolínek, for reminding me of the beauty in math and all the exciting games of chess; Christian Fiedler, for his rigor and great ideas; and A. René Geist, for being the spark of life in the office that made my PhD so much more enjoyable.

I would like to thank all the people I had the pleasure to collaborate and work with. In particular, I wan to to say thank you to Prof. Dr. Thomas Seel, Jonas Beuchert, Katharina Ensinger, Dr. Ravi Haksar, Behnam Khojasteh, and Dr. Katherine J. Kuchenbecker. Further, I had the privilege to supervise and co-supervise so many talented students, for which I am grateful. Thank you Henning Schlüter, Mona Buisson-Fenet, Sebastian Schlor, Pierre-François Massiani, Robin Kupper, and Claas Thesing. It was a pleasure to work with every one of you!

Last but not least, my family. My uncle Alexander, for teaching me that hard work pays off eventually. My parents, Nelli and Alexander, for all the support and always believing in me. And my brother Eugen, for encouraging me to follow my dreams, nurturing my passion for science, and for being an important role model in my life.

This work was supported by the Max Planck Society, the International Max Planck Research School for Intelligent Systems (IMPRS-IS), the Cyber Valley initiative, and the RWTH Aachen University.

Contents

«The brick walls are there for a reason. The brick walls are not there to keep us out. The brick walls are there to give us a chance to show how badly we want something. Because the brick walls are there to stop the people who don't want it badly enough. They're there to stop the other people.»

The Last Lecture

Randy Pausch

Chapter 1

Introduction

Learning-based algorithms have outperformed human experts in many environments. For example, in sophisticated games such as GO, chess, and StarCraft II. Further, they dominate most fields that involve computer vision and pattern recognition. Essentially, whenever it is possible to train complex learning algorithms in an offline fashion with nearly unlimited data, learning has proven to be successful. However, for many applications, this is not feasible.

Despite these recent success stories of learning algorithms, the substantial breakthroughs in applications that involve dynamical systems and optimal control tasks are less common. Clearly, the problems can be very difficult when complicated dynamics meet sophisticated control objectives. However, the issues are also rooted deeper and touch upon the very nature of dynamical systems. In contrast to games in virtual environments, learning-control problems are usually expected to extend to the real world, which is subject to constant change. Additionally, data is heavily correlated and thus, not independent. Generating informative data can be tough since it involves experiments on the system, which is costly, time consuming, and might damage the system. As a result, these properties make the problem fundamentally different from classical machine learning tasks. In the following, we will refer to the considered problem class as learning-control systems. Essentially, these are dynamical systems with a potential control input that are amenable to learning algorithms.

Learning-control systems are already challenging in an offline learning setting with a static controlled environment. However, most of these systems are supposed to work in the real world, which introduces a whole new dimension of problems. In particular, real-world environments are changing and the system needs to adapt its behavior to new and potentially unseen environments. However, there is an inherent cost to learning that may materialize itself in terms of energy, time, data, or on an abstract level – deviating from the control objective.

To this end, we propose a new paradigm for applying learning techniques to learning-control systems, which we call event-triggered learning (ETL). In a nutshell, learning updates are only triggered under the condition that there is a significant deviation between expected and actual behavior. Further, we actively influence the data properties by exciting the system to ensure well-behaved learning outcomes.

This thesis (including the following introduction sections), is partly based on multiple publications during the author's time as a PhD student. Details on the contributions and corresponding parts in this thesis will be given in Sec. 1.5 and throughout the thesis.

1.1 Motivation

"Never change a running system," or in a similar spirit "if it ain't broke, don't fix it," are popular heuristics when dealing with engineering systems. In practice, any type of update involves a risk of failure, i.e., the system starting to malfunction or being damaged. Therefore, a smoothly operating system is preferably only modified when necessary. When applying learning algorithms to learning-control systems, we propose to follow the same intuition. Only trigger the learning of new models and controllers, when the previous ones are insufficient.

The above arguments also extend to a deeper theoretical level and there are profound reasons to only learn when actually necessary. One key issue when working with learning-control systems are the opposing objectives of common control tasks and the requirements for successful learning outcomes. Regulating a system around a desired configuration with as little deviation or motion as possible or trajectory tracking are prominent examples. However, if a system is well regulated and barely moving, data is not informative and mainly dominated by noise. Hence, it is problematic to use it for learning and the outcomes can be unpredictable. Therefore, for successful learning, we need to excite the system, i.e., introduce some movement, which counteracts the control objective.

Besides not learning anything new due to the excitation problem, there are more reasons to only learn when necessary when dealing with learning-control systems. In particular, continuous learning can lead to a catastrophic forgetting of previously learned control behaviors. Hence, after some time it might fail to deal with disturbances. In the adaptive control community, this phenomenon is often referred to as bursting (Anderson, 2005). Theoretically, we could ensure informative data by exciting the system. However, the price of doing so is not negligible. Moving the system creates a physical cost and requires energy. Further, heavy computations can be required to update the underlying models.

Therefore, we should only learn if necessary. For this, we need to detect when there are changes in the environment that make learning necessary. For many machine learning tasks, it is possible to detect and learn changes in the distribution of test data, e.g., through sliding windows or related approaches.

Learning-control systems are substantially different from classical machine learning problems, which makes the above intuitive heuristics particularly relevant. Even in the most benign setting, we need to deal with correlated data, which is a challenge for most learning algorithms. Linear systems are among the most popular and tractable system classes. Even though it is possible to derive closed-form solutions to many relevant tasks, such as the prediction of future states, the design of quadratic regulators, or the filtering problem, most algorithms rely on precise knowledge of the system dynamics. While we can learn those dynamics, deriving statistical properties of estimators and learning outcomes is, due to the correlated data, far from trivial and subject of extensive research, e.g., (Simchowitz et al., 2018). The nonlinear problem is even worse and most problems become intractable.

We want to detect when learning is necessary, ensure informative data for a limited only through targeted learning experiments, and then stop the learning to exploit the improved models or controllers. Hence, the core question of ETL—when to trigger learning updates in learning-control systems.

1.2 Event-triggered Learning

Next, we introduce the main ideas behind ETL on an abstract level and develop the core arguments that lead to algorithms that can be applied to networked control systems and optimal control problems. In Fig. 1.1, we illustrate in a block diagram the interactions between the different components. Through the learning trigger, we only learn when necessary and can then excite the system. Correlated data is still an issue. We discuss how to derive statistical guarantees for various instances of learning triggers throughout the thesis. Learning is triggered when something unexpected happens, which is made tractable through the following technical key aspects.

Event-triggered Learning. *We define an ETL algorithm through the following steps:*

1) ***test signal:*** *the first step is defining a test signal or test object, Ψ, which we compare to a model of how the signal or object should behave, which we call $\hat{\Psi}$. It is important that the signal represents relevant system properties. One natural option are the system parameters themselves $\Psi = \Theta$ and $\hat{\Psi} = \hat{\Theta}$, which*

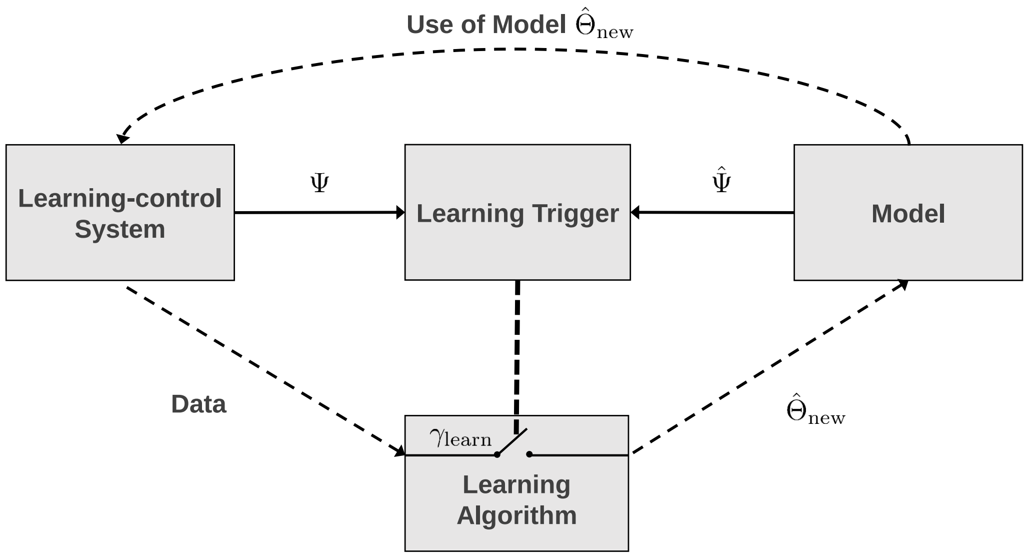


Figure 1.1: Proposed abstract event-triggered learning architecture. In the top left, we see learning-control system that generates the test signal Ψ. On the right, there is the model. The model-based quantity $\hat{\Psi}$ is compared by the learning trigger against the measured Ψ. Only when there is a significant deviation between the two objects, the learning trigger γ_{learn} gets activated. Then, the dashed lines become active and the learning algorithm receives data from the systems and outputs an improved model $\hat{\Theta}_{\text{new}}$. For example, the model can be used to design the learning-control system (as is the case in LQR) or to make predictions relevant for the control (as is the case for the NCS).

we consider in Chap. 5. Other choices are inter-communication times τ or the stationary distribution $\mathbb{P}$ of a dynamical systems.

2) ***learning trigger:*** *the learning trigger is essentially, a statistical test $\psi \in \mathbb{R}$ that acts on the test signal. We need critical thresholds κ^+ and κ^- that reject the null-hypothesis $H_0 : \Psi = \hat{\Psi}$ when $\psi \notin [\kappa^-, \kappa^+]$. In particular, we want to control the Type I error by bounding the false positives by some probability α. Whenever the test detects a deviation, we set the binary learning trigger γ_{learn} to $\gamma_{\text{learn}} = 1$. Controlling the Type II error is in most of the problems we are considering intractable. Thus, we might not detect small differences between two similar systems.*

3) ***learning algorithm:*** *implements a learning algorithm to update the model or controller and preferably leverage the statistical information from 2). Learning is only activated if the trigger from 2) detects significant deviations, i.e., $\psi \notin [\kappa^-, \kappa^+]$. In this thesis, we are relearning new models $\hat{\Theta}$, however, it is also possible to directly relearn control policies or communication strategies. By learning a new model, we can ensure that $\hat{\Theta}$ and Θ coincide again. After we have reached*

sufficient accuracy, we stop learning. We set the learning trigger $\gamma_{\text{learn}} = 0$ and continue to monitor the test statistic to detect further demand for learning. Through event-triggered learning, we can directly integrate adaptive behavior into many powerful algorithms as we will demonstrate in the next chapters.

The interaction between the learning trigger and the learning algorithm is what makes ETL special. Learning updates are only activated when a significant difference between expected behavior and observed data is detected. Thus, learning can be performed in a controlled environment. Afterward, the system stops learning and returns to its designated task. Nonetheless, the trigger keeps monitoring the behavior, and there are no more updates until the trigger again detects the change and new learning experiments are triggered. Due to the ETL architecture, we can always ensure a well-defined learning environment and avoid corrupted data that may lead to arbitrary outcomes otherwise.

1.3 Preliminaries

Linear time-invariant (LTI) systems are the backbone of classical control and systems theory. Due to their linear structure, they yield analytically tractable closed-form solutions to various relevant problems. Further, they can provide valid local approximations for nonlinear systems, which makes them applicable to many real-world problems. Our primary object of interest is the state $x_k \in \mathrm{R}^{d_x}$, which is usually related to physical quantities such as positions and velocities. In the following, we introduce preliminaries for LTI systems, which will be the focus of Part I and Part II. In Part III, we will extend those definitions to a broader class of nonlinear systems

Definition 1 (LTI System)**.** *Let $A \in \mathbb{R}^{d_x \times d_x}$, $B \in \mathbb{R}^{d_u \times d_x}$, and $\Sigma_x \in \mathbb{R}^{d_x \times d_x}$. Further, let Σ be positive definite. We call a system of the type*

$$x_{k+1} = Ax_k + Bu_k + \epsilon_k \tag{1.1}$$

an LTI system. The control input $u_k \in \mathbb{R}^{d_u}$ can be chosen and we assume the process noise $\epsilon_k \sim \mathcal{N}(0, \Sigma_x)$ is independent for all k.

To ensure well-defined control properties, we further assume that the pair (A, B) is controllable. Essentially, this means that we can move the system to arbitrary states by applying the corresponding control inputs.

We also consider the case where not all state variables are fully measured and we have observations $y_k \in \mathrm{R}^{d_y}$. This yields the additional measurement equation:

$$y_{k+1} = Cx_k + Du_k + \nu_k, \tag{1.2}$$

where the matrices C and D are of appropriate dimensions, the pair (A, C) is observable and $\nu_k \sim \mathcal{N}(0, \Sigma_y)$ independent measurement noise. The observability assumptions ensure well-defined estimates, for example, through Kalman filtering. Throughout this thesis, we assume everywhere that $D \equiv 0$.

Since we are considering linear systems, we can parameterize the system dynamics as $\Theta = (A, B, C, D, \Sigma_x, \Sigma_y)$. The true parameters Θ are usually unknown and therefore, we will focus on learning a model $\hat{\Theta}$ from data.

We also consider the continuous-time problem that is usually more involved and requires different types of mathematical tools. In particular, the process noise ϵ_k is mathematically a problematic object in continuous time and leads to the domain of stochastic calculus. Therefore, we need the following additional assumptions to ensure a well-defined problem setting. Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in\mathbb{R}^+}, P)$ be a filtered probability space, $S \subset \mathbb{R}^{d_x}$ a compact set, which is the state space of the dynamical system, and $\mathcal{B}$ the corresponding Borel σ-algebra.

In Chapter 2, the continuous-time framework is helpful to characterize inter-communication times as stopping times of certain stochastic processes, which yields well-behaved random variables for the learning trigger. All learning triggers are also derived for the practically more relevant discrete-time setting.

Assume $X(t)$ is a solution to the following stochastic differential equation (SDE)

$$\mathrm{d}X(t) = \mathcal{A}X(t)\mathrm{d}t + \Sigma_{\mathrm{X}}\,\mathrm{d}W(t), \quad X(0) = x_0. \tag{1.3}$$

Solutions to the SDE (1.3) are well investigated and known as Ornstein-Uhlenbeck (OU) processes (Øksendal, 2003). Further, let $\mathcal{A} \in \mathbb{R}^{n\times n}$ be a matrix with negative eigenvalues, which may in practice be obtained by applying local feedback control and considering the stable closed-loop dynamics. Assume $\Sigma_{\mathrm{X}} \in \mathbb{R}^{n\times n}$ is a positive definite matrix, $W(t) \in \mathbb{R}^n$ a standard Wiener process that models process noise, and the initial point $x_0 \in \mathbb{R}^n$ is known. Since the Wiener process is almost everywhere non-differentiable, the SDE is to be considered in an integral sense and the notation $\mathrm{d}W(t)$ indicates an Itô integral.

Mathematically, we are dealing with (stochastic) differential equations and their discretized equivalents. Thus, the states are clearly not independent, which is a challenge for learning algorithms and statistical testing. Independent data is usually a core requirement to ensure that some underlying probability distribution is sufficiently covered given a certain amount of data. For learning-control systems, we need additional assumptions to obtain statistical tests with theoretical guarantees. Instead of state data, we can also consider different properties or features that depend on the system dynamics and are better suited for statistical tests. For example, inter-communication times τ as in Part I, which characterize important system properties.

1.4 Main Contributions

Next, we present three instantiations of ETL algorithms that are core elements of this thesis. In Part I, we make the choices for ETL precise for networked control systems and in Part II for two different optimal control algorithms. In Part III, we take a step back and consider the fundamental question of comparing dynamical systems directly from states and designing statistical tests for certain classes of nonlinear dynamical systems.

1.4.1 Networked Control Systems

Before going into the details of how ETL looks like in this context, we first provide a brief background on networked control systems (NCSs). These are multiple systems connected to each other over a usually wireless communication network (Hespanha et al., 2007). A core aspect of networked control systems with many agents is to save communication in order to avoid network overload. As we will detail below, popular estimation and control methods use sporadic (or event-triggered) communication, where data is exchanged only when necessary (in some sense). Specifically, we consider a case where systems use model-based predictions and only need to communicate if those predictions deviate from the ground truth. Then, how much data is exchanged depends on the quality of the model used for control and estimation. We thus develop ETL for NCS where we trigger model learning based on communication properties.

We extend the paradigm of event triggering to model learning, which leads to event-triggered learning and is built on top of a typical event-triggered state estimation architecture (see Fig. 2.1 on Page 24). The new architecture can cope with changing dynamics and yields further communication savings in an NCS.

This instantiation of ETL is based on the time intervals between communication events, which we call inter-communication times τ. These inter-communication times are independent and identically distributed random variables that are fully parameterized through the system dynamics. Thus, by comparing expected communication (based on the model) and actual communication behavior (that depends on the ground truth dynamics), we can identify inaccurate models and trigger learning experiments to improve the models.

Next, we discuss the specific choices for the ETL algorithm that is designed around the communication behavior in networked control systems. The proposed architecture was introduced in in Solowjow and Trimpe (2020) and Solowjow et al. (2018).

Event-triggered Learning 1 (for Networked Control Systems)**.** *We obtain an ETL algorithm for saving communication in networked control systems through the following choices in the abstract formulation:*

1) ***test signal:*** *we consider inter-communication times τ since this is the key property we ultimately care about for networked control systems. Further, they are fully parameterized by the system dynamics and contain sufficient information about system properties;*

2) ***learning trigger:*** *we design statistical tests around the inter-communication times. Due to their advantageous statistical properties, i.e., independent and identically distributed, we can directly leverage classical tests that compare the expected value and the cumulative distribution function;*

3) ***learning algorithm:*** *we apply least squares estimators that identify the linear system dynamics every time from scratch.*

Here, we avoid many issues that are associated with learning-control systems. In particular, inter-communication times are suitable features that represent the quantity we ultimately care about and simultaneously, they are independent, which is from a statistical point of view very advantageous.

Based on a human walking experiment, we show in Beuchert et al. (2020a,b) actual communications savings for a practically relevant problem. Our method improves classical event-triggered state estimation algorithms that rely on static models. As long as the specific environment is not changing, the static models work well. However, once there is change, e.g., the subject changes the speed of walking or a slope is introduced, then the models become inaccurate and are not suitable anymore to predict future states. Detecting the change allows for the selective learning of improved models. Without change, the ETL algorithm essentially coincides with the classical framework.

1.4.2 Optimal Control

Further, we provide ETL algorithms for model-based optimal control methods. In particular, we consider two classical and popular control architectures, the linear quadratic regulator (LQR) and model-predictive control (MPC), which we both augment with ETL.

Linear Quadratic Regulators (LQR)

We assume system dynamics of the type (1.1) and the optimal control problem with the quadratic cost function

$$J = \lim_{N \to \infty} \frac{1}{N} \mathbb{E} \left[\sum_{j=0}^{N-1} x_j^\top Q_{\mathrm{LQR}} x_j + u_j^\top R_{\mathrm{LQR}} u_j \right], \tag{1.4}$$

where Q_{LQR} and R_{LQR} are symmetric and positive definite matrices with compatible dimensions. The optimal solution is a static feedback

$$u_k = -F x_k, \tag{1.5}$$

where $F \in \mathbb{R}^{q \times n}$ is the feedback gain. We can use Riccati equations to find analytical solutions for F.

These Riccati equations depend on the system dynamics. In practice, the exact parameters are usually unknown and instead, models are used. Since poor models usually lead to poor controllers, we want to detect these discrepancies. A consequence of poor models is a difference between the expected cost and the actual realization of the signal when the controller is applied to the system.

Since at the end of the day, we often only care about the control performance and not the model accuracy, we propose to design an ETL algorithm around the cost signal.

Event-triggered Learning 2 (for Linear Quadratic Regulators)**.** *We obtain the ETL algorithm that was proposed in Schluter et al. (2020) through the following design choices:*

1) ***test signal:*** *we consider the control cost J as a random variable.*
2) ***learning trigger:*** *the cost is heavily correlated since it is a quadratic transformation of consecutive states. By computing the full distribution of the cost signal in form of the moment generating function, we can apply sophisticated Chernoff bounds that yield the desired concentration results.*
3) ***learning algorithm:*** *learning is again realized with least-squares estimators that identify the system from scratch.*

Due to the strong assumptions on the system, i.e., linear dynamics, quadratic cost, and Gaussian noise, we can compute the full distribution of the cost and thus, explicitly deal with the correlations in the states. Next, we generalize the approach by directly considering the model quality.

Model Predictive Control (MPC)

The following ETL algorithm is independent of the control architecture and applicable to any linear learning-control system. However, we demonstrate it for MPC since it is popular and yields an efficient possibility to generate informative data through dedicated learning experiments.

The MPC setting is very similar to the above LQR problem. However, we consider a finite time horizon in (1.4) and optimal control inputs are computed in a receding horizon fashion by unrolling the model instead of a static feedback gain as in (1.5). A major difference between LQR and MPC is the consideration of state and control constraints. The MPC framework can directly include these into the optimization problem, while LQR has trouble coping with constraints. Further, in the MPC formulation, it is straightforward to modify the cost objective. We leverage this fact for active learning experiments that minimize uncertainty in addition to the control task.

We utilize a Kalman filter as a parameter filter to track the system parameters. Through the probabilistic nature of the filter, we directly obtain a posterior distribution that we use for testing.

Event-triggered Learning 3 (for Model Predictive Control)**.** *The following ETL algorithm was proposed in Schlor et al. (2022) and is based on a Kalman filter that tracks the system parameters:*

*1) **test signal:** we consider the outcome of the parameter filter as the random variables of interest—these are estimates of the system parameters;*

*2) **learning trigger:** based on the probabilistic formulation of the filter, we use the posterior distribution for testing. Due to the construction, the estimates are normal distributed and thus, we can apply a standard χ^2-test. In particular, we test the current model against the point estimate of the filter. The model is not updated during testing.*

*3) **learning algorithm:** whenever there is a significant deviation between filter and model, we start learning by using the recursive nature of the filter. However, we first excite the system to ensure an accurate learning outcome. Learning is realized by minimizing the uncertainty ellipsoid through the MPC formulation. Once, the uncertainty is small enough, the new point estimate is used as the new model.*

An important aspect is that the filter outcomes are only used for testing outside of the learning experiments. Only if there is a significant deviation, learning is triggered and the model is updated. This specific ETL algorithm demonstrates nicely how the learning algorithm can benefit from the trigger. Conversely, we can also show how learning permanently can lead to issues since the parameter estimates perform a random walk in a potentially large uncertainty ellipsoid when learning is not terminated eventually.

1.4.3 Comparing Dynamical Systems

In the last part, we consider the underlying core question of comparing dynamical systems directly from data, which will eventually pave the way towards a general nonlinear event-triggered learning framework. The general nonlinear problem with the dynamics

$$x_{k+1} = f(x_k) + \epsilon_k \tag{1.6}$$

is extremely challenging and requires more sophisticated tools. In particular, we avoid putting more structure on the dynamics f. Instead, in Solowjow et al. (2020), we propose to compare stationary distributions $\mathbb{P}$ of the states and design a kernel-based statistical test for this task. In particular, we consider kernel mean embeddings that lift the distributions into a reproducing kernel Hilbert space. Further, we leverage the Hilbert-Schmidt independence criterion, to address correlations within trajectories. Ultimately, we obtain the distance to an independent distribution in terms of the maximum mean discrepancy. Together with the kernel two-sample test, which compares the distributions of independent data in the same metric, we obtain a test for dynamical systems.

Depending on the goal, unnecessary complexity might be introduced by estimating intermediate objects that could be avoided, which is also known as Vapnik's principle: "When solving a problem of interest, do not solve a more general problem as an intermediate step." Here, we strive for a widely applicable framework with few assumptions that should serve as the basis for a general nonlinear event-triggered learning algorithm. Essentially, we address the questions of a suitable test object and statistical test by comparing the stationary distributions through a kernel two-sample test for dynamical systems.

Natural choices for the learning part are Gaussian processes dynamics in an active learning setting (Buisson-Fenet et al., 2020), conditional kernel mean embeddings, or deep learning-based approaches. While there is work on this, it still remains to combine these ideas with the developed statistical tests to efficiently relearn and update dynamics models.

1.5 Overview over Publications

This doctoral thesis is divided into three parts. In the first two, we develop instantiations of ETL algorithms for specific problems, i.e., saving communication in NCSs and model-based optimal control. In Part III, we consider the core question of comparing dynamical systems directly from state measurements. Eventually, the proposed test will pave the way towards nonlinear ETL algorithms.

Part I: Networked Control Systems

In Part I, we develop ETL for networked control systems with the goal of reducing communication. This part is based on the following publications:

- **Friedrich Solowjow** and Sebastian Trimpe, "Event-triggered learning", Automatica 117, © Elsevier 2020.
- **Friedrich Solowjow**, Dominik Baumann, Jochen Garcke, and Sebastian Trimpe, "Event-triggered learning for resource-efficient networked control", American Control Conference (ACC), © AACC 2018.

The second paper contains additional hardware experiments on an inverted pendulum that are not presented here. Instead, we present more mature experimental results that were developed in the following publications:

- Jonas Beuchert, **Friedrich Solowjow**, Sebastian Trimpe, and Thomas Seel, "Overcoming bandwidth limitations in wireless sensor networks by exploitation of cyclic signal patterns: An event-triggered learning approach", Sensors 20.1 (2020).
- Jonas Beuchert, **Friedrich Solowjow**, Jörg Raisch, Sebastian Trimpe, and Thomas Seel, "Hierarchical event-triggered learning for cyclically excited systems with application to wireless sensor networks", IEEE Control Systems Letters 4.1, ©IEEE 2020.

For the example of human walking, ETL was utilized to reduce communication in sensor networks, where limited bandwidth and energy consumption are serious problems. Through the new ETL algorithms, both issues could be addressed by reducing the overall communication load by up to 70%.

Part II: Optimal Control

In Part II, we first develop a learning trigger for the linear quadratic regulator problem. The learning trigger is at its heart based on the distribution of the cost functional. The derivation of the moment generating function itself is new and a contribution. Designing learning triggers based on Chernoff bounds goes even further and yields a test that is tailored to the distribution of the control cost. The learning trigger is validated in simulations and experiments on an inverted pendulum.

- Henning Schlüter[1], **Friedrich Solowjow**[1], and Sebastian Trimpe, "Event-triggered learning for linear quadratic control", IEEE Transactions on Automatic Control, 66(10), ©IEEE 2020.

  Next, we consider model predictive control and derive a different type of learning trigger that directly considers the estimates of the model parameters. Most contributions here are of a conceptual nature by tightly integrating and synergizing the learning trigger and learning algorithm.

- Sebastian Schlor[1], **Friedrich Solowjow**[1], and Sebastian Trimpe, "Parameter filter-based event-triggered learning", IEEE Transactions on Control Systems Technology (under review), 2022.

The presented framework is independent of the control architecture and can be applied to a wide range of linear learning-control systems. Model predictive control serves as a well-suited example since we can efficiently deal with relearning the dynamics. In addition to the control objective, we trigger a dual control objective when detecting change and minimize the uncertainty of the parameter estimates as well.

Part III: Comparing Dynamical Systems

The last part of this thesis, Part III, is focused on the core question of comparing the stationary distributions of dynamical systems. The proposed statistical test extends well-established kernel two-sample tests to dynamical systems. In particular, we generalize critical independence assumptions of the test to a new type of mixing that can be estimated from data. The proposed type of mixing has rich theoretical properties and we show relations to other standard notions of mixing. In particular, it is applicable to deterministic and stochastic systems. Further, we validate the test in extensive simulations and on experimental human walking data.

- **Friedrich Solowjow**, Dominik Baumann, Christian Fiedler, Andreas Jocham, Thomas Seel, and Sebastian Trimpe, "A kernel two-sample test for dynamical systems", Transactions on Machine Learning Research (under review), 2022.

Additional Publications by the Author

In addition to the publications listed above, the author has also contributed to the following publications:

- **Friedrich Solowjow**, Arash Mehrjou, Bernhard Schölkopf, and Sebastian Trimpe, "Efficient Encoding of Dynamical Systems through Local Approximations", IEEE Conference on Decision and Control (CDC), ©IEEE 2018.
- Dominik Baumann, **Friedrich Solowjow**, Karl Henrik Johansson , and Sebastian Trimpe, "Event-triggered pulse control with model learning (if necessary)", American Control Conference (ACC), © AACC 2019.

[1]Equally contributing

- Ravi N. Haksar, **Friedrich Solowjow**, Sebastian Trimpe, and Mac Schwager, "Controlling heterogeneous stochastic growth processes on lattices with limited resources", IEEE Conference on Decision and Control (CDC), ©IEEE 2019.
- Mona Buisson-Fenet, **Friedrich Solowjow**, and Sebastian Trimpe, "Actively learning Gaussian process dynamics", Learning for Dynamics and Control, 2020.
- Dominik Baumann, **Friedrich Solowjow**, Karl Henrik Johansson , and Sebastian Trimpe, "Identifying causal structure in dynamical systems", Transactions on Machine Learning Research, 2022.
- Pierre-François Massiani, Steve Heim, **Friedrich Solowjow**, and Sebastian Trimpe, "Safe value functions", IEEE Transactions on Automatic Control (in press), ©IEEE 2022.
- Katharina Ensinger, **Friedrich Solowjow**, Sebastian Ziesche, Michael Tiemann, and Sebastian Trimpe, "Structure-preserving Gaussian Process Dynamics", Joint European Conference on Machine Learning and Knowledge Discovery in Databases (accepted), 2022.

These publications are not part of this thesis but the author contributed to all of them during his time as a PhD student.

1.6 Contributions by the Author

This thesis is based on several publications that are the basis for Chap. 2—6. The authors' contribution is reflected by the order of names in the publications. The first author has contributed the most, however, all authors were involved in formulating the problems, discussing solutions, evaluating the results, and writing the paper. For some papers, the first and second author are equally contributing.

The publications and experimental results that are presented in Chap. 3 are the results of a collaboration with Jonas Beuchert and Thomas Seel, who have picked up the idea of event-triggered learning. In particular, the algorithm that was presented in Solowjow and Trimpe (2020) was implemented for sensor networks and parts of the algorithms were improved. The author contributed to formulating the problems, discussing solutions, evaluating the results, and writing the paper.

The publications presented in Chap. 4 and Chap. 5 have emerged from masters' thesis projects that were supervised by the author, which resulted in an equal contribution between the first and second authors.

1.7 Literature Overview

Next, we connect the contributions of this thesis to related work. In Sec. 1.7.1, we discuss conceptually similar ideas to event-triggered learning. We focus on concepts from cognitive science, to show that ETL shows similarities to learning in humans, and selected concepts from machine learning although they lack many properties that are a consequence of learning-control systems. Afterward, we focus on the domain-specific details of the instantiations of ETL. In Sec. 1.7.2, we discuss how ETL is conceptually similar to the idea of communicating information in multi-agent systems only when necessary. There are several methods in control theory that are closely related to ETL. For example, dual, robust, and adaptive control. We discuss the difference and similarities in Sec. 1.7.3. Sec. 1.7.4 discusses different possibilities for comparing dynamical systems.

The discussions are based on the corresponding publications (Solowjow and Trimpe, 2020; Schluter et al., 2020; Schlor et al., 2022; Solowjow et al., 2020).

1.7.1 Event-triggered Learning

The objective of detecting change has been around for a long time and has been developed in several communities. First, we discuss general similarities to methods from cognitive science and machine learning and afterward, move on to more domain-specific related work from networked control systems, optimal control, and systems theory.

Comparing expected behavior with observed realizations is also common in cognitive science to model human learning (Butz et al., 2003a). This effect is often quantified with the aid of internal models and anticipation along surprisal boundaries (Butz et al., 2003b). Interestingly, from an abstract point of view, the idea of comparing expected with observed behavior is similar to ETL. Yet, the approaches differ in the considered systems, applied methods, and concrete implementations. Nonetheless, there are also additional ideas that could greatly benefit future research in ETL such as hierarchical models and inductive learning biases as discussed in Butz (2021).

In machine learning, concept and distributional drifts are problems, where the underlying probability distribution that generates the data can change (Lu et al., 2018; Gama et al., 2014). While the general frame is similar to ETL, the applications and goals differ. Concept drift is usually closely related to big data problems and prediction systems, which is very different from the herein considered ETL applications that are focused on dynamical systems. Many techniques are window-based, such as the method presented in Klinkenberg and Joachims (2000) and discussed in Gama et al. (2014). However, there are also ideas based on hypothesis testing (Lu et al., 2018) that are conceptually closely related to ETL and could lead to new beneficial developments.

1.7.2 Part I: Networked Control Systems

Various event-triggered control (Lemmon, 2010; Heemels et al., 2012; Miskowicz, 2015) and state estimation (Lemmon, 2010; Shi et al., 2015; Trimpe and Campi, 2015; Trimpe, 2017) algorithms have been proposed for improving resource usage in NCSs. Approaches differ, among others, in the type of models that are used for predictions. The send-on-delta protocol (Miskowicz, 2006) triggers data transmission when the difference between the current and last communicated value passes a threshold. This protocol is extended to linear predictions in Suh (2007), which are obtained by approximating the signal derivative from data. More elaborate protocols use dynamics models of the observed process, which typically leads to more effective triggering (Trimpe and D'Andrea, 2011; Sijs et al., 2014; Trimpe and D'Andrea, 2014; Trimpe and Baumann, 2019; Sijs and Lazar, 2012; Wu et al., 2013; Battistelli et al., 2018; Han et al., 2015).

Recent articles proposed to improve and augment typical event-triggered state estimation (Shi et al., 2014; Battistelli et al., 2018; Huang et al., 2017) and control algorithms (Vamvoudakis and Ferraz, 2018; Vamvoudakis et al., 2019; Baumann et al., 2018; Funk et al., 2021; Narayanan and Jagannathan, 2017) with data-based techniques. In these works, learning is used to approximate intractable conditional probability densities that arise in distributed problems or to obtain tractable solutions to Hamilton-Jacobi-Bellman equations that yield optimal control policies, e.g., with model-free methods such as Q-learning, or based on neural networks. However, none of these works considers principled decision making on model learning to improve prediction accuracy, as we do with ETL.

In order to obtain effective learning triggers, we take a probabilistic view on inter-communication times (i.e., the time between two communication events) and trigger learning experiments whenever the expected communication differs from the empirical. A similar probabilistic interpretation of inter-communication times is considered in Xu and Hespanha (2004) and Rabi et al. (2008), where NCSs are modeled as jump-diffusion processes, and the expected value of inter-communication times is considered. We analyze the statistical properties in a more general context and propose the design of learning triggers based on inter-communication times and concentration inequalities.

1.7.3 Part II: Optimal Control

The LQR framework is a popular control method with many recent applications (see for instance (Marco et al., 2016; Trimpe and D'Andrea, 2012; Mason et al., 2014; Borrelli and Keviczky, 2008)). Usually, the expected value of the cost function is minimized, however, there are also approaches that consider its variance such as minimum variance control (see (Åström, 2012) for details). Closed-form solutions for the variance of the cost were derived in Bijl et al. (2016). Taking this approach one step further, we consider the full distribution of the cost functional. In particular, we characterize the distribution in terms of the moment generating function. To our knowledge, this is the first such characterization of the LQR cost. Further, we develop learning triggers that perform goodness of fit tests that are closely designed to the problem at hand: model-based statistical properties of the cost are compared to observed empirical data. In particular, we leverage Chernoff bounds to derive confidence intervals that contain a predefined portion of the probability mass. Learning experiments are triggered whenever the empirical cost is not contained within these bounds. Further, we show that it is not sufficient to analyze the mean and higher moments, since there are many inherent challenges, such as auto-correlations and unbounded control costs.

Adaptive control (see (Åström and Wittenmark, 2013; Ioannou and Sun, 2012; Bradtke et al., 1994) and references therein) seeks to continuously update system parameters or controllers in order to cope with changing environments. Updating the parameters permanently makes adaptive control algorithm potentially fast and flexible, however, the convergence of such algorithms is usually tightly connected to persistently excited signal vectors (Bradtke et al., 1994), which is not necessarily satisfied. It is well known that some adaptive control schemes suffer from temporary instability, so-called bursting (Anderson, 2005). It can occur if the system converged to a steady state, and the measured signals are not persistently exciting. Then, divergence may become a dangerous problem. Further, there are results from statistical literature that simultaneous parameter estimation and testing might lead to distorted test statistics and different asymptotic distributions (Kac et al., 1955) of statistical tests. There exist statistical tests that explicitly take the distribution of the estimator into account (Babu and Rao, 2004), however, the dependency is often highly non-trivial. In our approach, we propose to separate control from learning. Furthermore, learning is only triggered when there is a significant difference to the expected behavior and hence, a difference in the signal we ultimately care about. Thus, we only update models and controllers when needed, which is conceptually very different from adaptive control.

Dual control circumvents divergence issues by using a twofold objective: optimizing control performance and minimizing model uncertainty. A survey on dual control can be found in Unbehauen (2000), and a general textbook on adaptive dual control is given in Filatov and Unbehauen (2004). Heirung et al. (2012) proposed a dual MPC scheme that takes the error covariance of the estimated model parameters into account in the cost function to be minimized. A similar control objective has already been introduced in Wittenmark (1975). In these approaches, however, model uncertainty is artificially increased over time to enforce excitation of the system. Intuitively, the controller permanently forces the system to move in order to continuously update the model parameters. Our approach for the MPC problem is similar, however, we rely on statistical tests that quantify the model accuracy and detect significant deviations. In situations, where permanent updates are undesirable, our ETL approach avoids unnecessary excitation of the system while providing theoretical guarantees.

Robust control (cf. (Zhou and Doyle, 1998) and references therein) is also related to the proposed method, but has a different objective. The goal of robust control is designing control policies, which work decently for a variety of system parameters without changing the controller. In the event-triggered learning approach, we keep the controller fixed as long as the system parameters are not changing significantly. However, when there is a significant change in the system, we propose to update the model automatically. Thus, the proposed method possess enough flexibility to adapt to new environments, while still being efficient and robust, in particular during times with no changes in the system.

Change and fault detection (Isermann, 1984, 2006; Looze et al., 1985; Zhan and Jiang, 1999) have been addressed in the control theory literature, and there are many methods that can be applied to the considered problem. This also involves closed-loop performance assessment and monitoring (Qin, 1998; Huang et al., 1997; Swanda and Seborg, 1999; Dong and Qin, 2018), which is conceptually related to our work, however, usually analyzes minimum variance control and does not consider more complex statistical objects such as moment generating functions or parameter filters. Instead, closed-loop data is used to evaluate the control performance in terms of minimizing the output variance (Qin, 1998; Harris, 1989). The outcome of this analysis of variance (ANOVA) is then compared to a benchmark performance. Often, minimum variance control serves as this benchmark; however, also user-specific criteria can define the desired performance if minimum variance control is not achievable nor desired (Jelali, 2006). In Julien et al. (2004), a performance benchmark for MPC is derived. Also, the benefit of updating the controller model is estimated. Further, model-plant mismatches are purely deduced on variance assessment. All of the above methods still struggle in industrial applications due to difficulties in interpreting the results and a lack of guidance for corrective action (Bauer et al., 2016). We address these shortcomings by leveraging the inherent parameter filter uncertainty ellipsoid to derive optimal excitation signals. These signals produce the required insights into the system's behavior for appropriate model updates.

There are many system identification techniques for estimating models of linear systems from data. A broad overview is given in Ljung (2009). For the LQR part, we use a standard least squares technique, however, for the MPC problem, we consider a more sophisticated Kalman filter that estimates the model parameters. In Garcia and Antsaklis (2016), a Kalman filter for state-space system identification is used in an iterative adaptive control scheme. We consider a similar Kalman filter. However, here, the Kalman filter is used not only to estimate the system's parameters but also to derive trigger conditions and guiding the subsequent learning. There has been a lot of research in designing optimal excitation signals to minimize the error covariance (Ljung, 2009; Bombois et al., 2011). We leverage the shape of the estimated uncertainty ellipsoid to design control inputs that minimize said ellipsoid. The general idea of parameter filters is standard and has been investigated, e.g., in Goodwin and Payne (1977); Ljung and Söderström (1983). However, combining these filters with statistical tests and utilizing the beneficial posterior properties for learning on necessity is new in ETL and one of the key contributions here.

Augmenting control architectures with machine learning techniques is a popular research direction. There are many advances such as in reinforcement learning (Recht, 2019; Lee and Hu, 2018), networked control (Yoo and Johansson, 2019; Eisen et al., 2018; Gatsis and Pappas, 2018), or model predictive control Chen et al. (2018); Nubert et al. (2020). There is also recent work from a statistical point of view on LQR and sample complexity (Dean et al., 2019; Krauth et al., 2019; Mania et al., 2019) that analyzes the convergence speed of popular methods. However, most work revolves around learning dynamics models and data-driven improvements of controllers. In contrast to this, our work investigates the question of *when to learn*, derives learning triggers, and provides theoretical guarantees for these triggers.

1.7.4 Part III: Comparing Dynamical Systems

There is only limited literature that explicitly investigates the question of how to compare dynamical systems from data. One possibility is the embedding of dynamical systems as infinite-dimensional objects into reproducing kernel Hilbert spaces (RKHS) with specifically designed kernels such as Binet-Cauchy kernels (Vishwanathan et al., 2007) or generalizations thereof as proposed in Ishikawa et al. (2018) and Ishikawa et al. (2019). A similar function-analytical approach is considered in Mezic (2016) and Klus et al. (2020), where the authors consider Koopman and Perron-Frobenius operators to obtain linear dynamics in an infinite-dimensional space. These articles leverage specifically designed kernels and linear operators associated with dynamical systems to obtain an embedding. However, none of the above articles proposes a statistical test that compares dynamical systems. This would require further finite sample and error bounds on the approximations of the infinite dimensional operators, which is far from trivial.

The critical technical issue for dealing with dynamical systems is non-i.i.d. data. There are several extensions of kernel two-sample tests that have been developed (Zaremba et al., 2013; Gretton et al., 2012b; Doran et al., 2014; Lloyd and Ghahramani, 2015; Chwialkowski et al., 2014; Chwialkowski and Gretton, 2014) to make them applicable to a broader range of problems, where non-i.i.d. data is also an issue. However, the strong mixing properties that are typically postulated limit the applicability of the results to dynamical systems. Surprisingly, there is only very limited work that addresses the estimation of mixing coefficients from data, as also acknowledged and emphasized in (McDonald et al., 2011). The approach proposed in (McDonald et al., 2011) is different from our work, as mixing is considered with respect to the total variation norm, which requires the estimation of complex intermediate objects while we estimate mixing properties directly from data. We propose a novel mixing notion that ideally fits the kernel two-sample test and that can also be *estimated* from data (in contrast to most other mixing notions).

Also in the systems and control community, approaches for comparing more general, nonlinear systems have been considered. For example, in (Umlauft and Hirche, 2019) the authors consider the question of when to trigger model updates. In robust control, there is the notion of the gap metric (Zhou and Doyle, 1998), which compares the closed-loop behavior of dynamical systems. These approaches are particularly promising to quantify the similarities between dynamical systems when trying to achieve effective transfer learning, as shown in (Sorocky et al., 2020). However, they usually rely on a given model and a certain linear structure in the system. Depending on the prior system information, there exist more model-based approaches that compare dynamical systems. But estimating such models of nonlinear systems can be difficult in practice (Schoukens and Ljung, 2019; Schön et al., 2011; Brunton et al., 2017; Ljung, 2009). Similarly, estimating the stationary measure of a dynamical system is also a highly nontrivial problem (Hang et al., 2018; Luzzatto et al., 2005). In our approach, we do not require any intermediate objects. Instead, we compare stationary distributions of dynamical systems directly from data.

Part I

Networked Control Systems

Chapter 2

Networked Control Systems

In this chapter, we present ideas from the conference publication (Solowjow et al., 2018) and the journal publication (Solowjow and Trimpe, 2020). We construct the first types of learning triggers to detect deviations between a model $\hat{\Theta}$ and the ground truth parameters Θ. Due to the specific problem setting, we can leverage standard statistical i.i.d. results that directly synergize with well-known objects such as the expected value, higher moments, and the cumulative distribution function (CDF).

In the end, we summarize experimental applications to sensor networks that are presented in the journal publications (Beuchert et al., 2020a,b)

2.1 Introduction to NCSs

Networked control systems are an active field of research and there are many details to focus on. Here, we consider a special case of estimation problems, where measurements are communicated over a wireless network with a limited bandwidth. Ideally, every agent should be able to communicate continuously. However, this is wasteful in terms of energy and also, overloads the network. In order to be more resource efficient, there are model-based algorithms that predict the states of all agents. Clearly, the effectiveness of these model-based algorithms heavily depends on the accuracy of the model $\hat{\Theta}$. At the same time, it can also be problematic and potentially wasteful to constantly learn models and communicate them over the network, which leads us back to the idea of event-triggered learning.

Next, we introduce a typical event-based state estimation (EBSE) algorithm, as used in, e.g., (Lemmon, 2010; Trimpe and D'Andrea, 2014), and afterward, augment it with ETL. In Fig. 2.1, an abstraction of such an EBSE algorithm is depicted in blue and in green, we see the new ETL part that sits on top of the EBSE framework. On the right side, there is a receiving agent that needs the state information $X(t)$

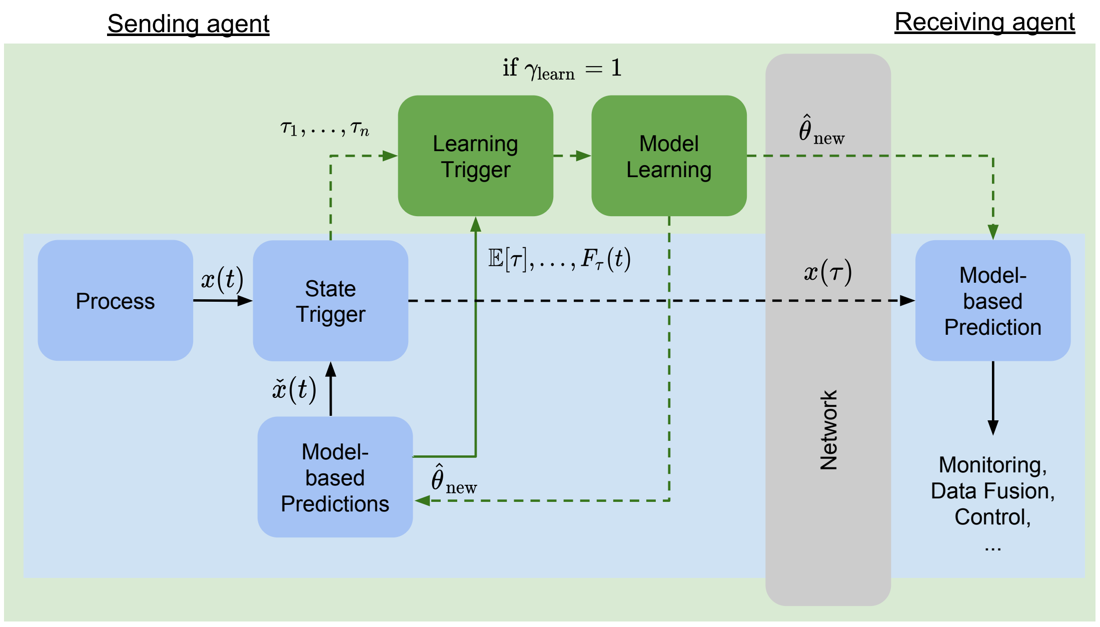


Figure 2.1: Proposed event-triggered learning architecture for a networked control problem between two agents. Based on a typical event-triggered state estimation architecture (in blue), we propose event-triggered learning (in green) to improve predictions and lower communication between Sending and Receiving agents. Learning experiments are themselves triggered as necessary by comparing empirical with expected inter-communication times.

of a sending agent, which is on the left side. Both agents are connected through a network and can communicate. Instead of communicating the state constantly, the receiving agent utilizes model-based predictions of the state $\hat{X}(t)$. Simultaneously, the sending agent also runs the same predictions and additionally, keeps track of the error $\|X(t) - \hat{X}(t)\|_2$. Whenever the error exceeds a predefined threshold δ, the sending agent communicates the true state $X(\tau)$ to the receiving agent and both update the prediction $\hat{X}(\tau)$ to the ground truth. The time of communication is denoted with τ.

The quality of the predictions heavily depends on the accuracy of the model $\hat{\Theta}$. Even if models are initialized close to the ground truth $\hat{\Theta} \approx \Theta$ there might be unexpected changes in the systems and the system might change, which results in a poor model and thus, poor predictions. With the aid of learning techniques it is possible to update the model again. This exemplary problem leads to the driving question of this thesis: *when should we learn?*

Analogously as in Fig. 2.1, where it is bad idea to constantly communicate states, it is also intractable to transmit models all the time over a network. This lead to the first event-triggered learning algorithms that only learn and update models when there is a significant error between model and ground truth. Here, however, the ground truth parameters are unknown and thus, we design statistical tests on a different target signal – the intercommunication times τ. We will show later that it is possible to characterize the intercommunication times as random variables (stopping times). The distribution of these stopping times is fully characterized by the system parameters Θ and the threshold δ. Therefore, it is possible to infer system changes by analyzing the distribution of τ. As indicated in Fig. 2.1, we compare model-induced statistical properties (i.e., expected value, variance, higher moments, or the CDF).

There are also deeper and more fundamental problems with *learning all the time*, which we will discuss in the later chapters. In the context of NCSs, the limited bandwidth is already reason enough to limit learning to the necessary instances since the updated model parameters need to be transmitted as well.

2.2 Problem Formulation

In this section, we make the problem of event-triggered learning precise for linear Gaussian time-invariant systems. The framework is developed in the context of NCSs and primarily focuses on limited bandwidth. Information exchange over networks is abstracted to be ideal in the sense that there are no packet drops or delays. First, we state the problem formulation for continuous time systems. We then address the discrete time case separately since the technical details differ slightly. In Sec. 2.7.1, the problem is extended to output measurements and, in particular, the Kalman filter setting.

2.2.1 Continuous Time Formulation

As introduced in Sec. 1.3, let $(S, \mathcal{F}, (\mathcal{F}_t)_{t\in\mathbb{R}^+}, \mathbb{P})$ be a filtered probability space and $X(t) \in \mathbb{R}^n$ a stochastic process, indexed by time $t \geq 0$. Furthermore, assume $X(t)$ (cf. 'Process' block in Fig. 2.1) is a solution to the following linear stochastic differential equation (SDE): $\mathrm{d}X(t) = \mathcal{A}X(t)\mathrm{d}t + \Sigma_{\mathrm{X}}\,\mathrm{d}W(t)$ with $X(0) = x_0$. We denote the system parameters as $\Theta = (\mathcal{A}, \Sigma_{\mathrm{X}})$ and models as $\hat{\Theta} = (\hat{\mathcal{A}}, \hat{\Sigma}_{\mathrm{X}})$.

For the model-based predictions ('Model-based Predictions' in Fig. 2.1), we use the expected value of system (1.3), which coincides with the open-loop predictions of the deterministic system $\mathrm{d}\check{X}(t) = \hat{\mathcal{A}}\check{X}(t)\mathrm{d}t$, with $\check{X}(0) = x_0$. Due to the stochasticity of the system, the prediction error will almost surely leave any predefined domain after sufficient time. Event-triggered communication ('State Trigger' in Fig. 2.1) bounds the prediction error by resetting the open-loop predictions $\check{X}(t)$ to the current state $X(t)$. Further, the binary event trigger

$$\gamma_{\text{state}} = 1 \iff \|X(t) - \check{X}(t)\|_2 \geq \delta, \tag{2.1}$$

is only activated when the error threshold $\delta > 0$ is crossed and hence, limits communication to necessary instances. The corresponding inter-communication time is defined as

$$\tau \coloneqq \inf\{t \in \mathbb{R} : \|X(t) - \check{X}(t)\|_2 \geq \delta\}, \tag{2.2}$$

and realizations of this random variable are denoted as $\tau_1, \dots, \tau_n$ and can be directly measured as the time between communication instances.

Assumption 2. *We assume $\tau \leq \tau_{\max} < \infty$.*

Bounded communication times are usually implemented in real-world applications to detect defect agents, which never communicate. Hence, communication is enforced after $\tau_{\max}$. For the design of the final learning trigger to be derived in this article, the assumption can be omitted. However, it is useful for intermediate results, such as the expectation-based learning trigger.

We address the problem of designing learning triggers ('Learning Trigger' in Fig. 2.1) based on inter-communication time analysis. Since the probability distribution of τ can be fully parameterized by Θ, we can derive an expected distribution based on the model $\hat{\Theta}$ and test if empirical inter-communication times are drawn from that distribution. Further, this statistical analysis yields theoretical guarantees, which are obtained from concentration inequalities and ensure that the derived learning triggers are effective. Therefore, we design a method to perform dedicated learning experiments on necessity and update models $\hat{\Theta}$ in an event-triggered fashion.

2.2.2 Discrete Time Formulation

Since processing on microcontrollers or sensors mostly happens on synchronously sampled data, we provide an alternative discrete time formulation of the considered problem. In principle, the problem formulation does not change. However, some essential details differ; for example, the inter-communication times from (2.2) need to be treated differently due to discontinuities in the states.

As introduced in Chap. 1, the discrete time analogue to (1.3) is

$$x_{k+1} = Ax_k + \epsilon_k, \quad x(0) = x_0, \tag{2.3}$$

with discrete time index $k \in \mathbb{N}$ and state $x_k \in \mathbb{R}^n$. Furthermore, we assume $A \in \mathbb{R}^{n \times n}$ has all eigenvalues strictly within the unit sphere since we do not consider any control here to stabilize the system. The model-based predictions are obtained through the equation $\check{x}_{k+1} = \hat{A}\check{x}_k$, which yields the trigger

$$\gamma_{\text{state}} = 1 \iff \|x_k - \check{x}_k\|_2 \geq \delta. \tag{2.4}$$

We define the system parameters and model as $\Theta = (A, \Sigma_x)$ and $\hat{\Theta} = (\hat{A}, \hat{\Sigma}_x)$. Hence, we obtain the inter-communication times

$$\tau^{\text{d}} := \min\{k \in \mathbb{N} : \|x_k - \check{x}_k\|_2 \geq \delta\}. \tag{2.5}$$

2.3 Communication as Stopping Times

In this section, we characterize inter-communication times (Eq. (2.2)) as stopping times of the prediction error process. The inter-communication time τ is a random variable and depends on the stochastic system (1.3). We seek to compare model-based expectations to observed data in order to detect significant inconsistencies between Θ and $\hat{\Theta}$. The core idea of the learning triggers comes down to deriving expected stopping time distributions based on the model $\hat{\Theta}$ and then analyzing how likely it is that observed stopping times $\tau_1, \ldots, \tau_n$ are drawn from this distribution.

Assuming $\hat{\Theta} = \Theta$, we derive model-based statistical properties of τ. Later on, we will test the hypothesis that empirical inter-communication times are indeed drawn from the derived distribution of τ—if not, this will indicate $\hat{\Theta} \neq \Theta$; that is, the model does not match reality.

2.3.1 Theoretical Properties

We define the error process as $Z(t) := X(t) - \check{X}(t)$. Due to linearity, it follows immediately that $Z(t)$ is an OU process as well and that $Z(0) = 0$. Next, we introduce inter-communication times with respect to the stochastic process $Z(t)$. Assume $\mathcal{F}_t = \sigma(Z_s : s \leq t)$ is the natural filtration on the given probability space and τ a stopping time with respect to $\mathcal{F}_t$. In particular, we consider the first exit time of the stochastic process $Z(t)$ from a sphere with radius δ, i.e., $\tau := \inf\{t \in \mathbb{R} : \|Z(t)\|_2 > \delta\}$, which precisely coincides with (2.2). Hence, we use the terms stopping times and inter-communication times synonymously in this article.

After each communication instance, we reset the process $Z(t)$ and set it to zero again by correcting $\check{X}(t)$ to $X(t)$. The sample paths of the process $Z(t)$ are (almost surely) continuous between two inter-communication times, which follows from the existence and uniqueness theorem of solutions to SDEs (cf. (Øksendal, 2003)). Therefore, we can precisely quantify the moment when the error threshold (2.1) is crossed. Further, it is possible to quantify statistical properties of τ such as the expected value (Pavliotis, 2014, Sec. 7.2) or the distribution (Patie and Winter, 2008) with the aid of certain boundary value problems. In particular, there are existence and uniqueness theorems (Patie and Winter, 2008) that imply that τ is mathematically well behaved.

2.3.2 Monte Carlo Approximations

Next, we describe how we obtain statistical properties of τ such as expected value $\mathbb{E}[\tau]$, variance, and cumulative distribution function (CDF) $F(t)$ with the aid of sample-based methods and hence, without solving nonlinear boundary value problems. Given the system parameters Θ, we can simulate trajectories of the stochastic process (1.3) with the aid of numerical sample methods such as the Euler-Maruyama scheme (cf. (Kloeden and Platen, 2011, Sec. 10.2) for an introduction to numerical solutions of SDEs).

In order to obtain independent and identically distributed (i.i.d.) samples $\tau_1, \ldots, \tau_n$, we sample the process $Z(t)$ and restart from zero after reaching the threshold δ. Alternatively, we could also simulate $X(t)$ and $\check{X}(t)$ and set the predictions $\check{X}(\tau)$ to the true value $X(\tau)$ when communication is triggered. The statistical properties of the corresponding stopping times do not differ because the processes $Z(t)$ and $X(t) - \check{X}(t)$ are indistinguishable. Further, stable OU processes are stationary and satisfy the strong Markov property, which generalizes the Markov property to stopping times.

For given i.i.d. random variables, we can approximate the expected value with $\frac{1}{n}\sum_{i=1}^{n} \tau_i$ and the CDF with $F_n(t) := \frac{1}{n}\sum_{i=1}^{n} 1_{\tau_i \leq t}$, where 1 is the indicator function. Quantifying the convergence speed of the above approximation will be vital in designing learning triggers.

2.4 Learning Trigger Design for Continuous Time

In this section, we design the learning trigger γ_{learn} (cf. Fig. 2.1) to detect a mismatch between model and true dynamics based on the inter-communication time τ.

2.4.1 Concentration Inequalities

The following results will form the backbone of the later derived learning triggers. Concentration inequalities quantify the convergence speed of empirical distributions to their analytical counterparts. In particular, Hoeffding's inequality bounds the expected deviation between mean and expected value. Further, we also consider the

Dvoretzky-Kiefer-Wolfowitz (DKW) inequality, which compares empirical and analytical CDF functions, and bounds the error between them uniformly. Essentially, we test if observed data fits the distribution, which is induced by the model $\hat{\Theta}$; that is, which was derived with $\hat{\Theta} = \Theta$ (cf. Sec. 2.3). If the distributions do not match, we conclude an unfit model and update $\hat{\Theta}$ through model learning.

Lemma 3 (Hoeffding (1963))**.** *Let $\tau_1, \dots, \tau_n$ be i.i.d. bounded random variables, s. t. $\tau_i \in [0, \tau_{\max}]$. Then*

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n}\tau_i - \mathbb{E}[\tau]\right| > \kappa\right] \leq 2\exp\left(-\frac{2n\kappa^2}{\tau_{\max}^2}\right). \tag{2.6}$$

We will first design learning triggers around the Hoeffding's inequality and later move on to richer statistical information. Therefore, we also want to analyze the convergence speed of the empirical CDF function.

Lemma 4 (DKW Inequality (Massart, 1990))**.** *Assume $\tau_1, \dots, \tau_n$ are i.i.d. random variables with CDF $F(t)$ and empirical CDF $F_n(t)$. Then*

$$\mathbb{P}\left[\sup_{t\in\mathbb{R}} |F_n(t) - F(t)| > \kappa\right] \leq 2\exp(-2n\kappa^2). \tag{2.7}$$

2.4.2 Expectation-based Learning Trigger

We propose a first learning trigger γ_{learn} based on the expected value $\mathbb{E}[\tau]$.

Exact Learning Trigger

Based on the foregoing discussion, we propose the following learning trigger:

$$\gamma_{\text{learn}} = 1 \iff \left|\frac{1}{n}\sum_{i=1}^{n}\tau_i - \mathbb{E}[\tau]\right| \geq \kappa_{\text{exact}}, \tag{2.8}$$

where $\gamma_{\text{learn}} = 1$ indicates that a new model shall be learned; $\mathbb{E}[\tau]$ is the analytical expected value, which is based on the model $\hat{\Theta}$; and $\tau_1, \tau_2, \dots, \tau_n$ are the last n empirically observed inter-communication times (τ_i the duration between two state triggers (2.1)). The horizon n is chosen to yield robust triggers in the sense that a larger time horizon allows the detection of smaller changes. However, it also increases the delay until the n samples are actually observed. The threshold parameter κ_{exact} quantifies the error we are willing to tolerate. There are some examples where it is possible to compute $\mathbb{E}[\tau]$ analytically. In general, however, it is intractable. Hence, we also propose the approximated learning trigger, which takes the approximations for the statistical analysis into account. We denote (2.8) as the *exact learning trigger* because it involves the exact expected value $\mathbb{E}[\tau]$, as opposed to the trigger derived in the next subsection, which is based on a Monte Carlo approximation of the expected value.

Even though the trigger (2.8) is meant to detect inaccurate models, there is always a chance that the trigger fires not due to an inaccurate model, but instead due to the randomness of the process (and thus randomness of inter-communication times τ_i). Such false positives are inevitable due to the stochastic nature of the problem. However, we obtain a confidence interval, which contains the empirical mean with high confidence. If observations violate the derived confidence interval, we conclude that distributions do not match, and learning is beneficial. Therefore, we propose to choose κ_{exact} to yield effective triggering with a user-defined confidence level. We then have the following result for the trigger (2.8):

Theorem 5 (Exact learning trigger)**.** *Assume τ and $\tau_1, \ldots, \tau_n$ are i.i.d. random variables and the parameters α, n, and $\tau_{\max}$ are given. If the trigger* (2.8) *gets activated ($\gamma_{\text{learn}} = 1$) with*

$$\kappa_{\text{exact}} = \tau_{\max} \sqrt{\frac{1}{2n} \ln \frac{2}{\alpha}}, \tag{2.9}$$

then

$$\mathbb{P}\left[\left| \frac{1}{n} \sum_{i=1}^{n} \tau_i - \mathbb{E}[\tau] \right| \geq \kappa_{\text{exact}} \right] \leq \alpha. \tag{2.10}$$

Proof. Substituting κ_{exact} into the right-hand side of Hoeffding's inequality yields the desired result. □

The theorem quantifies the expected convergence rate of the empirical mean to the expected value for a perfect model. This result can be used as follows: the user specifies the desired confidence level α, the number n of inter-communication times considered in the empirical average, and the maximum inter-communication time $\tau_{\max}$. By choosing κ_{exact} as discussed, the exact learning trigger (2.8) is guaranteed to make incorrect triggering decisions (false positives) with a probability of less than α.

Approximated Learning Trigger

As discussed in Sec. 2.3, obtaining $\mathbb{E}[\tau]$ can be difficult and computationally expensive. Instead, we propose to approximate $\mathbb{E}[\tau]$ by sampling τ. For this, we simulate the process $Z(t)$ and average the obtained stopping times $\hat{\tau}_1, \ldots, \hat{\tau}_m$. This yields the *approximated learning trigger*

$$\gamma_{\text{learn}} = 1 \iff \left| \frac{1}{n} \sum_{i=1}^{n} \tau_i - \frac{1}{m} \sum_{i=1}^{m} \hat{\tau}_i \right| \geq \kappa_{\text{approx}}. \tag{2.11}$$

The Monte Carlo approximation leads to a choice of κ_{approx}, which is different from κ_{exact} for small m. For $m \to \infty$ we see that κ_{approx} converges to κ_{exact}.

Theorem 6 (Approximated Learning Trigger)**.** *Assume $\tau_1, \ldots, \tau_n$, and $\hat{\tau}_1, \ldots, \hat{\tau}_m$ are i.i.d. random variables. If the trigger* (2.11) *gets activated ($\gamma_{\text{learn}} = 1$) with*

$$\kappa_{\text{approx}} = \tau_{\max} \sqrt{\frac{n+m}{2nm} \ln \frac{2}{\alpha}}, \tag{2.12}$$

then

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n} \tau_i - \frac{1}{m}\sum_{i=1}^{m} \hat{\tau}_i\right| \geq \kappa_{\text{approx}}\right] \leq \alpha. \tag{2.13}$$

Proof. First, we introduce an alternative formulation of Hoeffding's inequality (2.6)

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n} \tau_i - \frac{1}{m}\sum_{i=1}^{m} \hat{\tau}_i - (\mathbb{E}[\tau] - \mathbb{E}[\hat{\tau}])\right| > \kappa_{\text{approx}}\right] \leq 2\exp\left(-\frac{2\kappa_{\text{approx}}^2}{(m^{-1} + n^{-1})\tau_{\max}^2}\right),$$

which was already stated in the original article by Hoeffding (Hoeffding, 1963) as a corollary (Eq. 2.6). Here, we assume that τ and $\hat{\tau}$ are identically distributed and, therefore, the analytical expected values cancel out. Rearranging for κ_{approx} yields the desired result. □

2.4.3 Density-based Learning Trigger

Analyzing the expected values is, in general, not enough to distinguish random variables since higher moments such as variance can differ. Therefore, we propose to look at the CDF, build learning triggers around the DKW inequality (2.7), and thus use richer statistical information. We propose the following learning trigger:

$$\gamma_{\text{learn}} = 1 \iff \sup_{t \in \mathbb{R}} |F(t) - F_n(t)| > \kappa_{\text{exact}}. \tag{2.14}$$

The density-based learning trigger has the following property:

Theorem 7 (Exact Density Learning Trigger)**.** *Assume $\tau_1, \ldots, \tau_n$ are i.i.d. random variables with CDF $F(t)$ and empirical CDF $F_n(t)$. If the learning trigger* (2.14) *gets activated ($\gamma_{\text{learn}} = 1$) with*

$$\kappa_{\text{exact}} = \sqrt{\frac{1}{2n} \ln \frac{2}{\alpha}}, \tag{2.15}$$

then

$$\mathbb{P}\left[\sup_{t \in \mathbb{R}} |F(t) - F_n(t)| > \kappa_{\text{exact}}\right] \leq \alpha. \tag{2.16}$$

Proof. Follows directly from the DKW Inequality. □

Finally, we can follow the reasoning as before and obtain the sample-based version of the trigger (2.14)

$$\gamma_{\text{learn}} = 1 \iff \sup_{t\in\mathbb{R}} |\hat{F}_m(t) - F_n(t)| > \kappa_{\text{approx}}, \tag{2.17}$$

where $\hat{F}_m(t) = \frac{1}{m}\sum_{i=1}^m 1_{\hat{\tau}_i \le t}$ and $\hat{\tau}_i$ are obtained by creating samples based on the model $\hat{\Theta}$. This trigger is essentially the well established two-sample Kolmogorov-Smirnov (KS) test (Hodges, 1958).

Theorem 8 (Two-sample KS Learning Trigger)**.** *Assume* $\tau_1, \dots, \tau_n$ *and* $\hat{\tau}_1, \dots, \hat{\tau}_m$ *are i.i.d. random variables with empirical CDFs* $F_n(t)$ *and* $\hat{F}_m(t)$*. If the trigger* (2.17) *gets activated with*

$$\kappa_{\text{approx}} = \sqrt{\frac{n+m}{2nm} \ln\left(\frac{2}{\alpha}\right)}, \tag{2.18}$$

then

$$\mathbb{P}\left[\sup_{t\in\mathbb{R}} |\hat{F}_m(t) - F_n(t)| > \kappa_{\text{approx}}\right] \le \alpha. \tag{2.19}$$

Proof. Follows from the two-sample KS test. □

The density-based learning triggers do not depend on $\tau_{\max}$ and consider richer statistical information, which can be an advantage and will be discussed in detail in the experimental sections.

2.5 Learning Trigger Design for Discrete Time

Based on the previous discussion, we will now highlight how to apply the derived learning triggers to discrete time systems (2.3). The random variables τ (cf. Eq. (2.2)) and τ^{d} (cf. Eq. (2.5)) can differ significantly due to discretization effects. Intuitively, this effect can be thought of as the continuous time process crossing the δ-threshold and returning within the discretization time. Therefore, the discrete-time process has no possibility of observing the crossing, and hence, stopping times tend to be larger for discrete time systems. For small time steps, the difference tends to be negligible, and τ^{d} converges to τ in the limit.

In this section, we show that the *approximated* learning triggers transfer without any modification to the discrete time system. It is important to adjust the system parameters $\Theta = (A, \Sigma_x)$ and the model $\hat{\Theta} = (\hat{A}, \hat{\Sigma}_x)$ to discrete time (cf. Sec. 2.2.2) in order to sample from the correct distribution (i.e., sampling from the continuous time model, while the true dynamics are discrete, or vice versa). Only based on statistical tests, irrelevant of the actual shape, we decide if they coincide.

Theorem 9 (Discrete Time Learning Trigger)**.** *Assume Θ and $\hat{\Theta}$ correspond to the discrete time system* (2.3)*. Then, the previously derived approximated learning triggers* (2.11) *and* (2.17) *are applicable without any further modification.*

Proof. The derived learning triggers test if given observations of inter-communication times $\tau_1, \dots, \tau_n$ are drawn from the same distribution as $\hat{\tau}_1, \dots, \hat{\tau}_m$ and therefore, if $\Theta = \hat{\Theta}$. The concrete shape of the distribution is irrelevant for the test. □

Remark 10. *It is also possible to consider more complex noise models such as colored noise. The main challenge lies in identifying the system and, in particular, the noise model from data.*

2.6 Numerical Example – Reduced Communication

The learning triggers derived in the previous two sections are the core element in the proposed ETL architecture (Fig. 2.1, block 'Learning Trigger'). For 'Model Learning' in the context of linear Gaussian systems considered herein, one can use standard techniques for linear systems identification (Ljung, 2009), which we do not elaborate further. Thus, all components of the proposed ETL method in Fig. 2.1 are complete, and we present a first numerical example to illustrate the main ideas of the developed learning triggers.

2.6.1 Setup

Next, we introduce the system, and afterward, we apply the learning trigger in order to demonstrate how to detect an inaccurate model. We consider the first-order dynamical system $x_{k+1} = 0.9x_k + \epsilon_k$, with noise $\epsilon_k \sim \mathcal{N}(0, 1)$. Further, we assume the disturbed model $\hat{\Theta} = (0.8, 1)$ and hence, we obtain the predictions with the equation $\check{x}_{k+1} = 0.8\check{x}_k$. To bound the prediction error, we deploy the state trigger (2.4) with $\delta = 3$. In Fig. 2.2, we can see in the first graph a trajectory of states x_k as a black dashed line and the model-based predictions $\check{x}_k$ in blue. Whenever $\gamma_{\text{state}} = 1$, we set $\check{x}_\tau$ to x_τ. The error signal never crosses the δ-threshold and is depicted in the second graph. The communication instances are shown in the third graph. The distances between two consecutive communication instances corresponds to the inter-communication times. Further, we set $\alpha = 0.05$, $\tau_{\max} = 100$, $n = 300$, and $m = 100\,000$.

2.6.2 Expectation-based Learning Trigger

The proposed learning triggers analyze the statistical properties of the observed inter-communication times and compare them to model-induced quantities. For instance, in Fig. 2.3, the corresponding average inter-communication rate is shown (yellow line). The first n inter-communication times are stored in a buffer, and the empirical mean is computed. Based on the model $\hat{\Theta}$ and with the aid of m Monte Carlo simulations, we derive $\mathbb{E}[\hat{\tau}] \approx 28.6$ (dashed blue line). The model-based confidence interval is obtained with the aid of Theorem 6. After the buffer is successfully filled (at $k = 4961$), the learning trigger (2.11) compares if the buffered average inter-communication rate (yellow line) lies outside the expected confidence interval, which is the case here. Therefore, the learning trigger discovers an inaccurate model and triggers a learning experiment. Here, we abstract learning and set model $\hat{\Theta}$ to the true parameters Θ. A more detailed discussion on the learning aspect of ETL can be found in (Solowjow et al., 2018), where we demonstrated the effectiveness of ETL in hardware experiments on a cart-pole system.

After updating the model (at $k = 4961$), we empty the buffer, start collecting new stopping times, and reset the average inter-communication time accordingly. This causes the initial fluctuation of the signal. However, we see fast convergence to the model-based expectation. Further, the average inter-communication time was increased after updating the model (yellow line converges to a larger value), which results in less communication.

The test statistic is also depicted in Fig. 2.4 for the initial inaccurate model and in Fig. 2.5 for the exact model. The dashed blue line again represents the model-based expected value, while the dashed red line depicts the empirical mean at the moment of triggering.

2.6.3 Density-based Learning Triggers

Next, we will discuss the density-based learning trigger (2.17), which is also illustrated in Fig. 2.4 and Fig. 2.5. The solid blue line represents the model-based CDF function $\hat{F}_m(t)$, and the solid red line is the empirical CDF $F_n(t)$ based on observed inter-communication times. Here, both triggers detect the inaccurate model (cf. Fig. 2.4) and have high confidence in the true model since the model-based and empirical quantities coincide, which is depicted in Fig. 2.5. The confidence interval is derived with Theorem 8 and tighter than the expectation-based. Hence, inaccurate models can be detected faster.

2.6.4 Expected Value is not Enough

Here, we assume the model $\hat{\Theta} = (0.5, 1.7)$ with $n = m = 10\,000$. Intuitively, process noise and stability have opposite effects on the communication behavior. We construct the counterexamples by creating a hypersurface of models $\hat{\Theta}$, where the noise and stability effects cancel out. Figure 2.6 shows the expected and empirical expected values of inter-communication times, which are almost identical – bad performance is expected and also realized due to the bad prediction model. The CDF-based trigger is still able to detect the inaccuracy, which is a big advantage over the expectation-based learning triggers. Clearly, the highest inter-communication time is realized when $\hat{\Theta} = \Theta$, which can be observed when comparing Fig. 2.5 with Fig. 2.6, where the inter-communication time is twice as high.

Clearly, the model has also to be communicated at some point. However, this happens very rarely and, in particular, when there is a significant change in the system. We conclude that both learning triggers are effective in detecting a mismatch between model and true dynamics. Also, average communication was successfully reduced after updating the model.

2.7 Extensions and Insights

So far, we assumed that perfect measures of the full state x_k are available at the sending agent. In the following, we will drop this assumption and consider systems where only part of the state can be measured.

2.7.1 Output Measurements

Assume the following system

$$x_{k+1} = Ax_k + \epsilon_k, \quad y_k = Cx_k + \nu_k, \tag{2.20}$$

with output measurements $y_k \in \mathbb{R}^m$. Further, let $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{n\times m}$. The system is again assumed to be stochastic with process noise $\epsilon_k \sim \mathcal{N}(0, \Sigma_x)$ and observation noise $\nu_k \sim \mathcal{N}(0, R)$, which are independent of each other. We also assume that A is stable, and the pair (A, C) is observable. Hence, the system is parameterized by $\Theta = (A, C, \Sigma_x, R)$ and modeled by $\hat{\Theta} = (\hat{A}, \hat{C}, \hat{\Sigma}_x, \hat{R})$. To reconstruct the full state, we use a Kalman filter (KF), which is the optimal filter for linear Gaussian systems with exact models (Anderson and Moore, 2012). Here, we consider the steady-state KF and obtain

$$\hat{x}_{k+1} = \hat{A}\hat{x}_k + K\left(y_{k+1} - \hat{C}\hat{A}\hat{x}_k\right), \tag{2.21}$$

where $K \in \mathbb{R}^{m\times n}$ is the corresponding gain. Further, assume the process has already converged to stationarity.

Ideally, we want to use the KF states $\hat{x}_k$ on the receiving agent's site. However, this would require periodic communication of the estimates $\hat{x}_k$, or the measurements y_k, which we try to avoid. Exactly as in Sec. 2.2, we run a model-based prediction step in the absence of data, obtain $\check{x}_{k+1} = \hat{A}\check{x}_k$, and employ the state trigger $\|\hat{x}_k - \check{x}_k\|_2 \geq \delta$. Hence, the inter-communication time is defined as $\tau^{\mathrm{o}} := \min\{k \in \mathbb{N} : \|\hat{x}_k - \check{x}_k\|_2 \geq \delta\}$. Extending ETL to output measurements is based on treating the KF sequence as a stochastic process in its own right and investigating the distribution of $\hat{x}_k$. With the aid of the innovation sequence, we can derive an auto-regressive structure. The innovation of the KF is defined as $I_k = y_k - C\hat{x}_k$. Furthermore, it is well known that $I_1, \dots, I_n$ are independent normal distributed random variables with $I_k \sim \mathcal{N}(0, S)$ (Anderson and Moore, 2012). The covariance is given by $S = CPC^{\intercal} + R$, where P is the stationary error covariance matrix of the KF and can be obtained by solving the corresponding Riccati equation (Anderson and Moore, 2012, Equation (4.4)). Hence, we reformulate the KF as

$$\hat{x}_{k+1} = A\hat{x}_k + KI_k, \tag{2.22}$$

and regard $I_k \sim \mathcal{N}(0, S)$ as a random variable. By regarding KI_k as process noise, we are back to the previously discussed problem (cf. (2.3)) and can apply the derived tools and learning triggers. Hence, we can effectively analyze the distribution of the corresponding stopping time with the previously derived tools – sampling (2.22) to obtain model-based stopping times $\hat{\tau}^{\mathrm{o}}$.

2.7.2 Better Models may Result in more Communication

More accurate models result in better predictions. Thus, one may expect that improved models also lead to reduced communication of state information from sender to receiver (cf. Fig. 2.1). While this is indeed the case for perfect state measurements (as has been observed in the example of Sec. 2.6), it may actually be the opposite for the KF setting. Here, we present an example that demonstrates this rather unexpected effect – better models may lead to more communication. The reason is as follows: better models increase the KF performance, and thus, it is possible to track the unobserved states better. Therefore, it is possible to construct examples where communication increases, which is desirable for performance, though counterintuitive. Consider system (2.20) with the matrices

$$A = \begin{pmatrix} 1.000 & 0.010 & -0.005 & 0.000 \\ 0.017 & 1.027 & -0.301 & -0.061 \\ 0.000 & 0.000 & 0.997 & 0.009 \\ 0.046 & 0.067 & -0.507 & 0.850 \end{pmatrix}, C^\intercal = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{2.23}$$

which is obtained by linearizing the closed-loop dynamics of a stabilized inverted pendulum. We assume process noise $\epsilon_k \sim \mathcal{N}(0, 0.1I_4)$ and observation $\nu_k \sim \mathcal{N}(0, 0.1I_2)$, where I_n is the identity matrix of dimension n. Further, we assume that $\hat{\nu}_k \sim \mathcal{N}(0, 0.5I_2)$ and that the model otherwise coincides with the true system parameters. We consider the KF states $\hat{x}_k$ (cf. (2.21)), the predictions $\check{x}_k$, and the state trigger

$$\gamma_{\text{state}} = 1 \iff \|\hat{x}_k - \check{x}_k\|_2 \geq 1. \tag{2.24}$$

We initialize $x(0) = \hat{x}(0) = \check{x}(0) = 0$ and obtain the distribution over stopping times depicted in Fig. 2.7. The expected model-based communication is derived via Monte Carlo simulation of the innovation process (2.22), where we set $\tau_{\max} = 100$ and $m = n = 5000$.

In the first graph in Fig. 2.7, we can see that the empirical inter-communication times are higher than the model-based. Updating the model actually reduces the average inter-communication time (more communication), which is because the KF improves and tracks the states x_k better, which is illustrated in Fig. 2.8. The first plot shows the tracking performance when a perfect model is used $\hat{\Theta} = \Theta$ (KF states in yellow). For the second plot, we changed the covariance of ν_k to $\mathcal{N}(0, 0.5I_2)$, as discussed above. In the third plot, we exaggerated this effect even further by assuming $\nu_k \sim \mathcal{N}(0, 10I_2)$ in the model. In all three plots, we consider the first $k = 150$ time steps and stop afterward. In the third graph, we can see that the KF states deviate a lot from the true underlying states. However, they are still close to the open-loop predictions, and hence, there is very little communication. Despite the counterintuitive link between model accuracy and average communication, the example shows that the derived learning triggers are effective in detecting model mismatch.

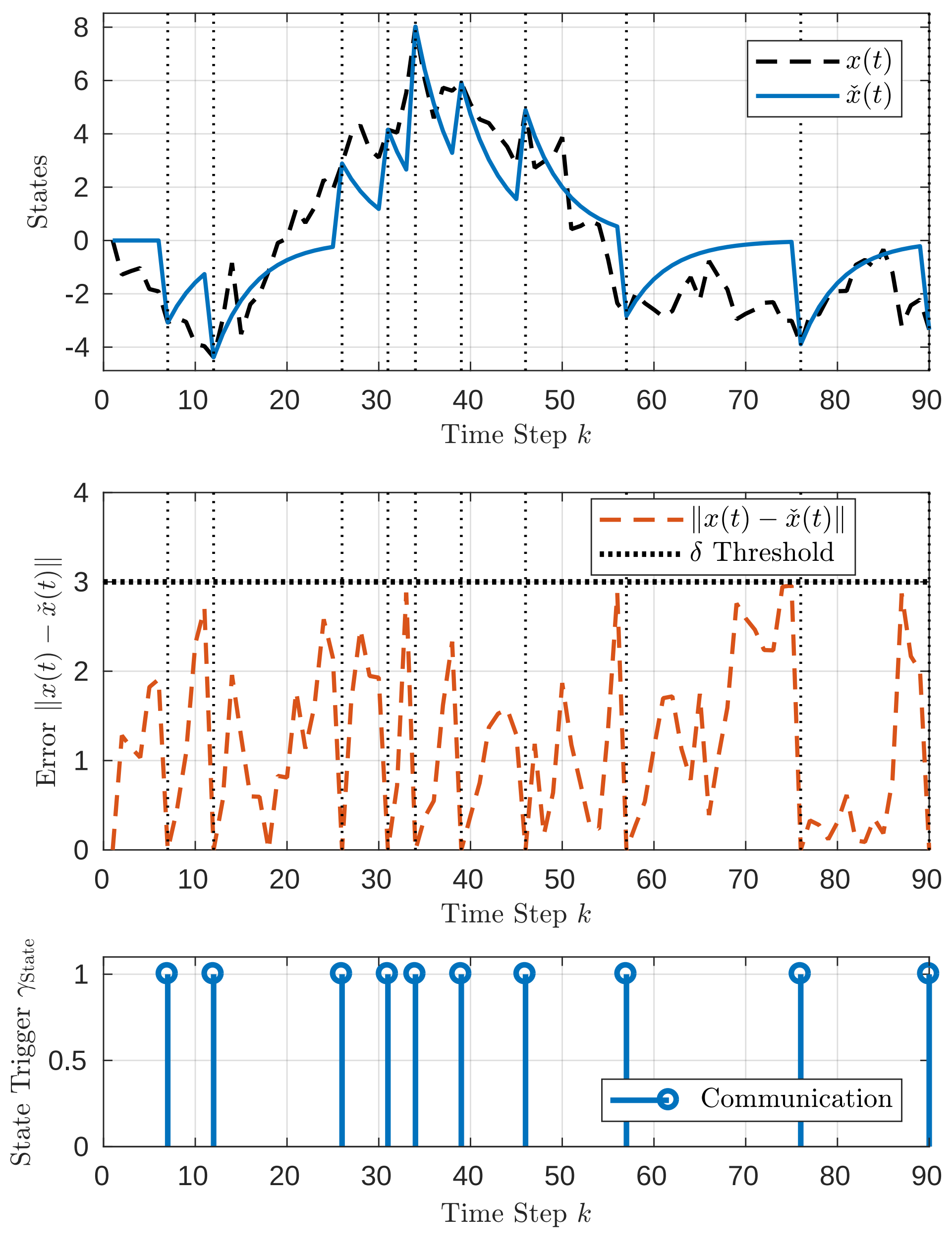


Figure 2.2: Example of model-based state predictions, which are reset to the exact state whenever the error would exceed the predefined threshold $\delta = 3$. States, error signals, and the first ten communication instances τ_i are depicted in the three graphs (top to bottom).

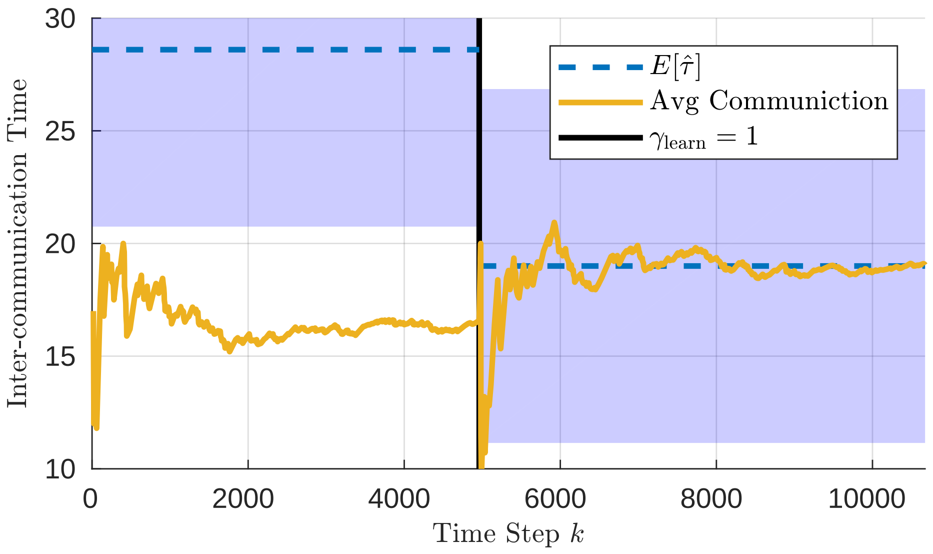


Figure 2.3: Average inter-communication times. The dashed blue line illustrates the model-based expected value and the yellow line the empirical mean. The shaded blue area around the dashed blue line indicates the confidence interval that should contain the empirical mean with 95% probability. Every new communication instant is added to a buffer of size $n = 300$. Afterward, the learning trigger compares the expected value with empirical mean and updates the model if the yellow line is outside the blue area. A new model-based expected value is computed, and the empirical mean coincides with it. Further, the inter-communication times increase, which corresponds to a decrease in communication.

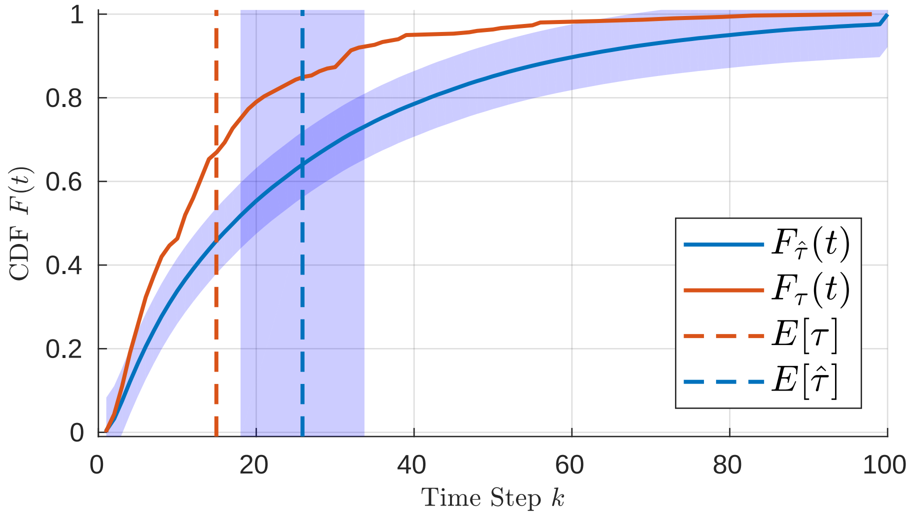


Figure 2.4: Statistical properties of expected communication $\hat{\tau}$ (blue) and observed inter-communication times τ (red). The dashed lines capture the expected values, and the shaded blue region is a confidence interval that should contain the dashed red line with a 95% probability (cf. (2.11)). As there is a significant deviation between observation and expectation, the learning trigger initiates to relearn the model. The solid lines represent the CDF functions, and also here, the empirical distribution is not contained within the confidence bounds, which triggers learning (cf. (2.17)).

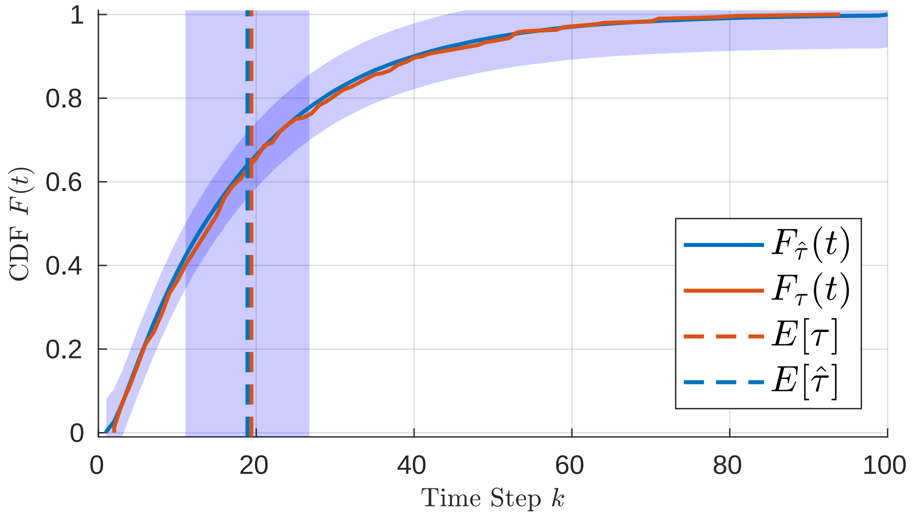


Figure 2.5: After updating the model $\hat{\Theta}$ to the true system parameters Θ, the estimated stopping times $\hat{\tau}$ coincide with empirical stopping times τ. As a direct consequence, communication behavior is improved as the stopping times increase.

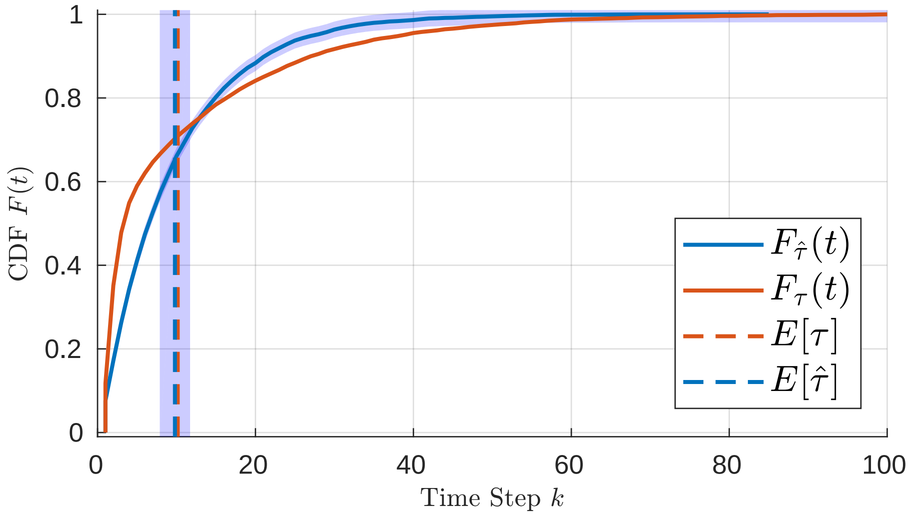


Figure 2.6: Counterexample showing that the expected value of inter-communication times (learning trigger (2.11)) may not be sufficient to detect an inaccurate model. Furthermore, the CDF-based trigger (2.17) detects the mismatch reliably.

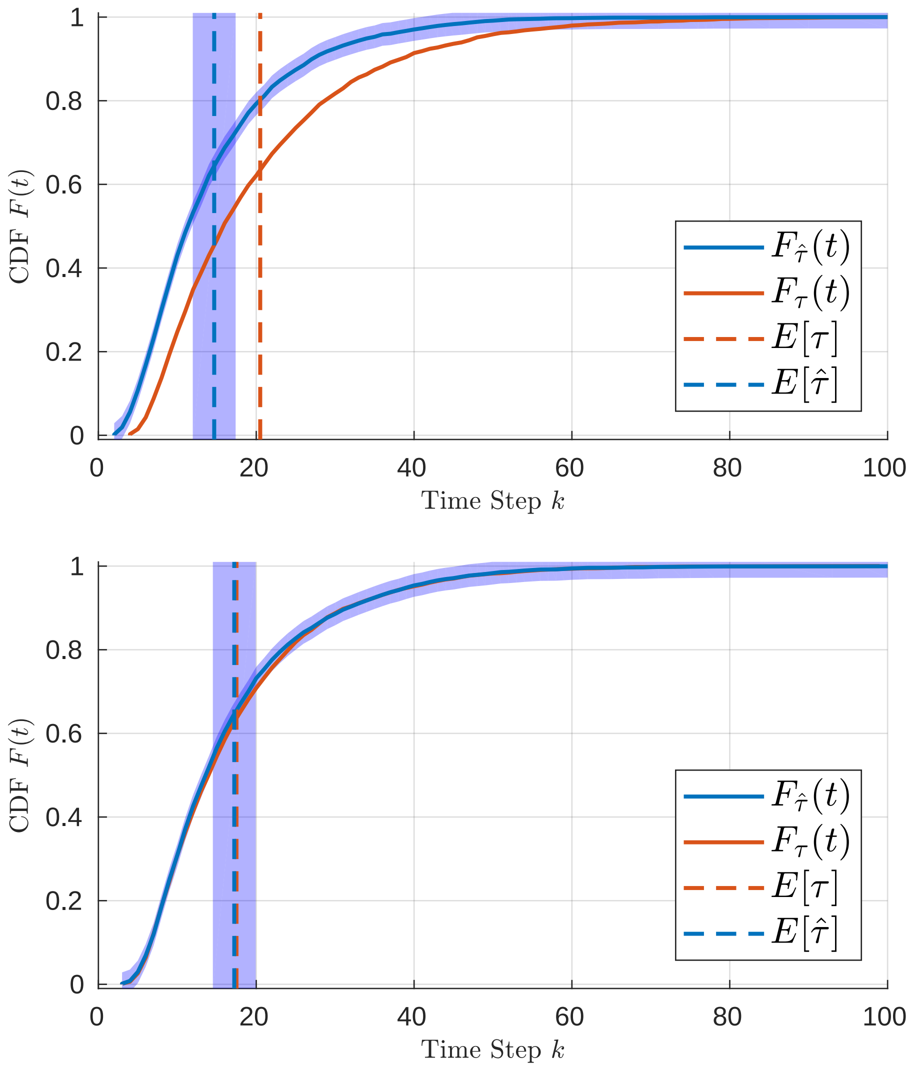


Figure 2.7: Communication behavior of a system with output measurements. In the first graph, we see the test statistic for an inaccurate model and in the second for $\hat{\Theta} = \Theta$. Both learning triggers ((2.11) and (2.17)) are effective in detecting the model mismatch. Interestingly, updating the model results in more communication, as can be seen by a decrease in the actual average inter-communication time $E[\tau]$ (dashed red) between top and bottom. With the improved model, the KF tracks the true states better, and thus, we obtain more communication.

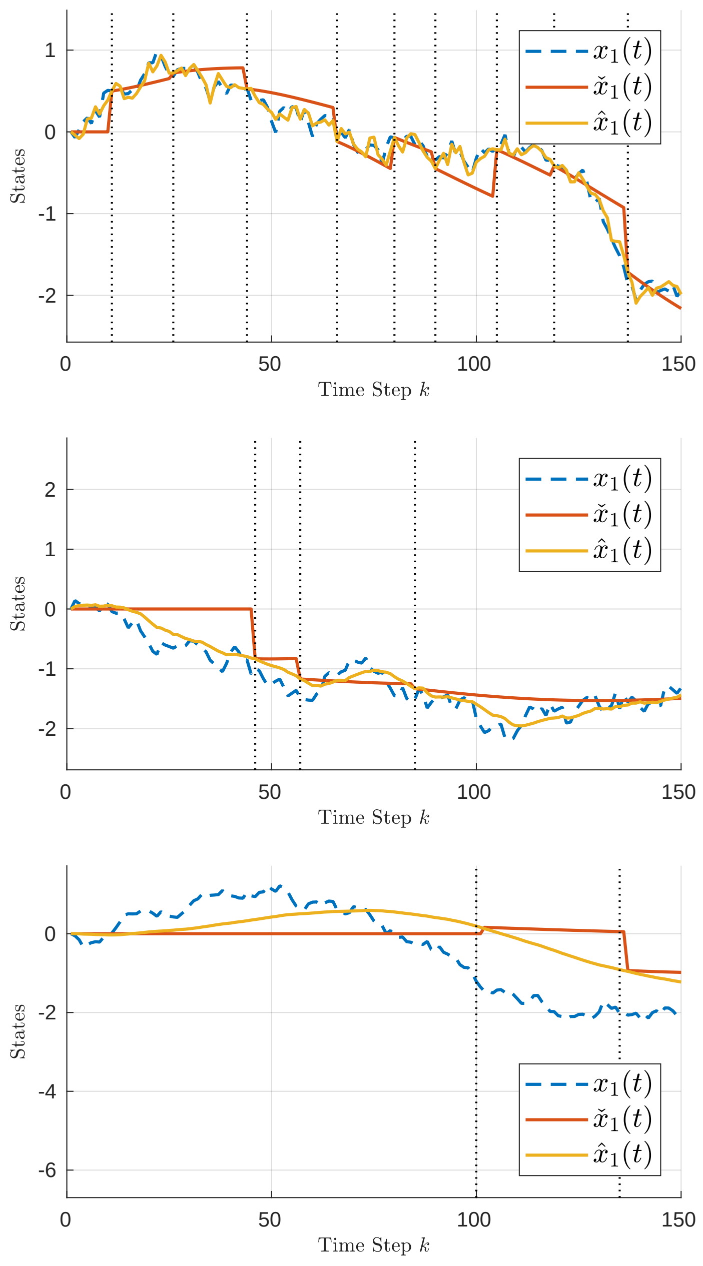


Figure 2.8: State trajectories of the first dimension of the four-dimensional system with output measurements (2.23). The state trigger (2.24) is applied and therefore, communication triggered when $\hat{x}_k$ and $\check{x}_k$ deviate by δ. Communication instances are depicted with dotted vertical lines. In the first graph, we can see how well the KF $\hat{x}_k$ (yellow) tracks the true states x_k (dashed blue). The red line depicts the open-loop predictions $\check{x}_k$. By worsening the model from the first to the second graph, the KF performance gets worse, which results in less communication. From the second to the third graph, we worsen the model even more, which results in even less communication, because the KF is not able to track the states.

Chapter 3

Experimental Application to Gait Data

The following part is based on the journal publications (Beuchert et al., 2020a,b), where event-triggered learning was applied for the first time to real-world sensor networks. In particular, these networks monitor human gait data and through event-triggered learning, the communication was effectively reduced. The publications contain additional conceptual contributions by Jonas Beuchert such as different levels of model learning, i.e., small and full model updates, which will not be covered here.

3.1 Application Example

Many applications require real-time transmission of signals over communication channels with bandwidth limitations. A typical example is given by wireless sensor networks in feedback-controlled systems. The number of agents (i.e., network nodes) and their communication rate is limited by the amount of information the wireless network can transmit in real-time. It is, therefore, desirable to reduce the communication load without compromising the accuracy of the transmitted signals.

When a signal is approximately constant or linear in time, it seems straightforward to accurately estimate the current sample from previously measured ones. However, if the signal varies in a less easily predictable manner, it is more difficult to find signal properties that can be exploited. In the present contribution, we consider signals that are approximately locally periodic. Approximately means that the signal can be well approximated by repetitions of a periodic pattern. Locally means that this pattern might change slowly or even suddenly, and for some time periods, there might be no periodic pattern at all. Many physiological signals exhibit this approximate local periodicity, for example respiratory, cardiopulmonary, and EMG data, as well as inertial measurement data from periodic motions such as walking, cycling, and swimming. Even though the shape of an ECG curve or the motion of the leg during swimming might change slowly or even suddenly, a very large portion of the signals is typically well described by cyclic patterns, and completely irregular episodes are rare. We will aim at exploiting this approximate local periodicity by identifying the patterns in real-time and transmitting only the data samples that cannot be adequately estimated from that pattern and previously transmitted data. To this end, we will use the proposed ETL methods.

3.2 Results

In summary, the communication load was reduced by 70% while maintaining small prediction errors, i.e., the foot angle was smaller than 2°. Thus, the predictions are sufficiently accurate for potential down stream applications such as control tasks. The measurement sampling rate was conservatively set to 50 Hz. For many applications, the sampling rates are substantially higher, which would result in even greater savings. Through the reduced communication load, it would be possible to triple the amount of sensors with the same bandwidth.

The papers (Beuchert et al., 2020a,b) contain in depth descriptions of the experiments and different settings. In Beuchert et al. (2020a), the focus was on a single foot during running, while Beuchert et al. (2020b) considers multiple sensors in a full body sensor network.

Further, ETL enables state-of-the-art prediction methods to adapt to changing running styles and speeds. While this is already a satisfying result, there is still the potential to improve the algorithms further and tailor them to the task at hand. One step in this direction was achieved through small updates that only update certain parameters, while full updates relearn the whole model. Extending the model class from linear models to Gaussian process regression was also considered in Beuchert et al. (2020b) and shown to be effective. A different extension would be to save the models and reuse them in a smart way and potentially with the model class that yields the lowest communication load.

Part II

Optimal Control

Chapter 4

Linear Quadratic Control

Next, we consider a different application for event-triggered learning – model-based control. The results presented in this chapter are based on the journal publications (Schluter et al., 2020) for the Linear quadratic regulator (LQR) and (Schlor et al., 2022) for the Model-predictive control (MPC) part.

4.1 Introduction

Linear quadratic regulator problems are well understood in literature and yield tractable and well-behaved solutions (see, for example, (Åström, 2012; Anderson and Moore, 2007) and references therein). Because of this, they are frequently used in practice, and even applications to nonlinear problems are possible with the aid of iterative methods that linearize the system dynamics (Todorov and Li, 2005). While LQR has favorable robustness properties (Lee et al., 2012), the performance of the controller naturally depends on the accuracy of the underlying model. Thus, just like any other model-based design, LQR will generally benefit from a precise model, both in terms of performance and robustness.

We propose to improve the model during operation from data *when needed.* Clearly, the idea of data-driven model updates is not new (Hou and Wang, 2013), however, principled decision making on when to learn is a novel approach. Learning permanently can be wasteful from a resource point of view and may suffer from divergence issues when the system is standing still, and there is no persistent excitation in the data (Åström and Wittenmark, 2013; Anderson, 2005). Hence, we propose to separate the process of learning from the nominal behavior of the system and investigate the question of *when to learn.* By automatically detecting the instances where learning is beneficial, we maintain the advantages of both data-driven and optimal control approaches by performing learning in a controlled environment and

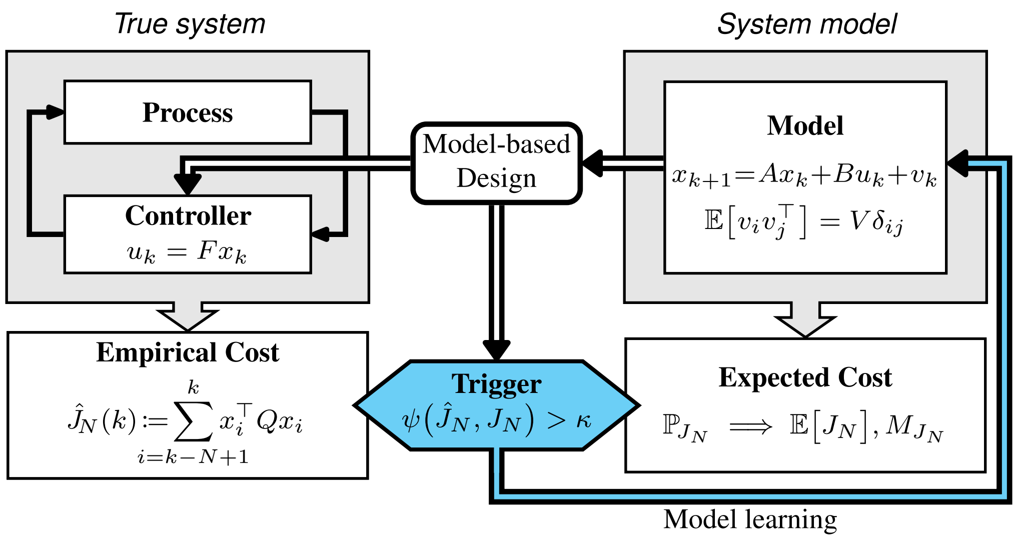

True system
System model
Process
Controller
$u_k = Fx_k$
Model-based Design
Model
$x_{k+1} = Ax_k + Bu_k + v_k$
$\mathbb{E}[v_i v_j^\top] = V\delta_{ij}$
Empirical Cost
$\hat{J}_N(k) := \sum_{i=k-N+1}^{k} x_i^\top Q x_i$
Trigger
$\psi(\hat{J}_N, J_N) > \kappa$
Expected Cost
$\mathbb{P}_{J_N} \implies \mathbb{E}[J_N], M_{J_N}$
Model learning

Figure 4.1: Proposed event-trigger learning architecture. The classic model-based controller architecture is extended by introducing the triggering of model learning when needed (blue) in order to improve performance. Learning experiments are triggered whenever there is a significant difference between the empirical cost, which is observed from data, and the theoretically expected cost, which is analytically derived from the model. After identifying an improved model, a new controller and trigger are derived based on said updated model. Thus, the model-based controller is closer to the underlying dynamics and, therefore, yields a reduced cost.

afterward, applying the rich optimal control framework to learned models. However, the crucial difficulty lies in deciding *when to learn*, which we address herein with the aid of an event-triggered learning (ETL) approach, whose architecture is depicted in Fig. 4.1.

The key contribution of this part lies in designing dedicated learning triggers that compare empirical costs to a model-induced distribution. Our presented approach is specifically designed for linear Gaussian systems and the signal we care about, which is the control cost. Thus, we obtain an efficient algorithm with tight confidence bounds that are based on the herein derived expression for the moment generating function of the cost.

Contributions

To summarize, this chapter makes the following contributions:

- introducing the concept of control performance-based event-triggered learning for linear Gaussian systems – the model and therefrom derived quantities are only updated when there are significant changes in the online performance;

- characterization of the full distribution of the LQR cost functional via moment generating functions (Thm. 13);
- design of an effective learning trigger based on the full distribution (Sec. 4.3) of the LQR cost with additional theoretical guarantees, which are derived with the aid of concentration inequalities and demonstration that considering moments (i.e., the expected value) is not sufficient; and
- validation and comparison of the derived triggers in simulation and hardware experiments, in which we demonstrate fast, reliable, and robust detection.

4.2 Event-Triggered Learning for LQR control

In this section, we formulate the problem of event-triggered learning for linear quadratic control and present the main ideas.

4.2.1 Problem setup

We assume again the linear dynamics

$$x_{k+1} = A_{\mathrm{o}} x_k + B u_k + \epsilon_k, \tag{4.1}$$

with discrete-time index $k \in \mathbb{N}$, state $x_k \in \mathbb{R}^n$, control input $u_k \in \mathbb{R}^q$, system matrix $A_{\mathrm{o}} \in \mathbb{R}^{n \times n}$, input matrix $B \in \mathbb{R}^{n \times q}$ and independent identically distributed (i.i.d.) Gaussian noise $\epsilon_k \sim \mathcal{N}(0, \Sigma_x)$ with $\mathbb{E}\left[\epsilon_i \epsilon_j^\top\right] = \Sigma_x \delta_{ij}$. Further, the system is assumed to be (A_{o}, B)-stabilizable. Hence, stable closed-loop dynamics can be achieved through state feedback

$$u_k = -F x_k + u_{\mathrm{ref},k}, \tag{4.2}$$

where $F \in \mathbb{R}^{q \times n}$ is the feedback gain and $u_{\mathrm{ref},k}$ is a known reference, which can be used to track a trajectory or excite the system in order to generate informative data.

A stabilizing feedback gain can be obtained, for instance, via LQR design (Anderson and Moore, 2007). In particular, we can use Riccati equations to find analytical solutions to the optimal control problem with the quadratic cost function

$$J = \lim_{N \to \infty} \frac{1}{N} \mathbb{E}\left[\sum_{j=0}^{N-1} x_j^\top Q_{\mathrm{LQR}} x_j + u_j^\top R_{\mathrm{LQR}} u_j \right], \tag{4.3}$$

where Q_{LQR} and R_{LQR} are symmetric and positive definite matrices with compatible dimensions. In the following, we consider the empirical cost over a finite horizon N, which we will denote at time step k by

$$\hat{J}_N(k) = \sum_{j=k-N+1}^{k} x_j^\top Q_{\mathrm{LQR}} x_j + u_j^\top R_{\mathrm{LQR}} u_j \,. \tag{4.4}$$

A normalization is not needed here since the cost will remain finite when considering a finite horizon. Thus, we will drop the normalization for notational convenience since it has no theoretical influence on the later obtained results.

To further ease the notation, we write

$$x_{k+1} = Ax_k + \epsilon_k, \tag{4.5}$$

with $A = (A_o - BF)$ and obtain

$$\hat{J}_N(k) = \sum_{j=k-N+1}^{k} x_j^\top Q x_j, \tag{4.6}$$

where $Q = \Big(Q_{\text{LQR}} + F^\top R_{\text{LQR}} F\Big)$.

It is well-known that the states of a stable system (such as (4.5)) converge to a stationary Gaussian distribution. In particular, the steady-state covariance $X^V := \lim_{k\to\infty} \mathbb{E}[x_k x_k^\top]$ can be computed as the solution to the Lyapunov equation (e.g., (Kuvaritakis and Cannon, 2014, Lemma 2.1))

$$AX^V A^\top - X^V + \Sigma_x = 0. \tag{4.7}$$

The stationary state covariance X^V is a key object for the following technical development, and thus, we want to explicitly point out the technical assumptions that are necessary.

Assumption 11. *The closed-loop model (4.5) is stable in the sense that* $|\lambda_{\max}(A)| < 1$.

This assumption is not very restrictive, as we only require the feedback law (4.2) to stabilize the open-loop model (4.1).

Assumption 12. *The system has converged to a steady state, in the sense that* $\mathbb{E}[x_k] = 0$ *and the covariance* $\mathbb{E}[x_k x_k^\top] = X^V$ *are time-invariant.*

Given Assumption 11, it follows directly that the system converges exponentially to a steady-state Gaussian distribution (Kumar and Varaiya, 1986, Sec. 3.1). The problem can easily be generalized to $\mathbb{E}[x_k] = \mu$ by subtracting the constant mean from given data. Thus, the assumptions made here are not very strong.

In the following, we distinguish between the model-induced cost J_N, which is a random variable, and the empirical cost $\hat{J}_N(k)$, which is sampled from the system. For the random variable J_N, we can drop the dependency on k. This follows directly from assuming stationary states in Assumption 12. Since we are considering quadratic transformations of stationary random variables (the states) and the summation over a fixed window of length N, the random variable J_N is itself stationary. Thus, we can omit the index k.

4.2.2 Problem and Main Idea

In this work, we systematically analyze the question of *when to learn* a new model of the dynamical system (4.1), which is later on utilized to synthesize a controller. Due to the structure of the problem, we are able to quantify how well the controller should perform in terms of expected value, variance, or a distributional sense. The statistical testing is carried out under the null hypothesis that model and ground truth coincide. Thus, by checking if theoretically derived properties actually coincide with empirically observed cost values, we are able to detect significant mismatches between the current model and the ground truth dynamics.

This idea leads to the proposed ETL architecture shown in Fig. 4.1. The core piece of the proposed method is the binary event trigger γ_{learn} for learning a new model and the corresponding test statistic ψ that quantifies how likely it is that empirical samples $\hat{J}_N(k)$ coincide with the model-induced random variable J_N. Given a level of confidence α, we are able to compute critical thresholds κ and, thus, trigger learning experiments on necessity. Since we are considering linear systems here, the main emphasis is on the design of the test statistic ψ. Identifying linear systems is not the focus here and has been extensively discussed in previous work (see (Ljung, 2009) for an overview). After a new model is identified, we propose to compute a new controller and derive new trigger thresholds. We thus summarize the core problem addressed here.

Learning Trigger (LQR)**.** *Detect, when the model has changed, by comparing the deviation of model-induced cost properties to empirical costs, thus, yielding the learning trigger*

$$\psi\Big(\hat{J}_N, J_N\Big) > \kappa \Leftrightarrow \gamma_{\text{learn}} = 1, \tag{4.8}$$

where ψ is an appropriate test statistic, κ is the computed critical threshold and γ_{learn} is a binary indicator for whether a model update is required ($\gamma_{\text{learn}} = 1$) or not ($\gamma_{\text{learn}} = 0$).

Due to the Gaussian process noise, the proposed trigger will also exhibit an expected probabilistic behavior. In particular, it is impossible to entirely avoid false positive learning decisions. Therefore, we take this explicitly into account when designing the learning trigger by choosing κ such that

$$\mathbb{P}\Big[\psi\Big(\hat{J}_N, J_N\Big) > \kappa\Big] < \alpha\,, \tag{4.9}$$

i.e., the probability of the trigger misfiring is less then desired confidence level α.

First, we develop a learning trigger incorporating the entire distribution in the form of a moment generating function in Sec. 4.3, which allows for an efficient implementation to detect system changes. Then, we demonstrate that the straightforward approaches using single moments of the cost (Sec. 4.4) pose theoretical challenges. For the case of the expected value, we show that the trigger based on moment generating function with its superior theoretical properties also yields better empirical performance and reliability.

4.3 Distribution-based Trigger

In this section, we derive a learning trigger, which is directly based on confidence bounds of the empirical cost. The idea is to apply the Chernoff bound to the empirical cost in order to obtain a likely range of values. Then, the trigger can easily detect values outside of this interval. Before we can apply the Chernoff bound, we first need to derive the moment-generating function (MGF) of the cost function.

4.3.1 Moment Generating Functions

The moment-generating function (MGF) $M_X(\xi) := \mathbb{E}[e^{\xi X}]$ of a random variable X – if it exists – is a powerful tool to characterize the distribution (see, e.g., (Gut, 2013, Chapter 4) for more details on MGFs). It is moment-generating in the sense that for all $n \in \mathbb{N}$, the n-th moment $\mathbb{E}[X^n]$ can be obtained by computing $\frac{\mathrm{d}}{\mathrm{d}\xi} M_{J_N}(\xi)|_{\xi=0}$.

Next, we will compute the MGF of the cost J_N and afterward, combine it with Chernoff bounds to obtain a powerful trigger.

Theorem 13 (Moment Generating Function of the Cost)**.** *Assuming the state sequence $z = (x_0, x_1, \dots, x_{N-1})^\top$ is a jointly normally distributed random variable with mean μ and covariance matrix Σ. The moment generating function of the cost $J_N = z^\top \Omega z = \sum_{k=0}^{N-1} x_k^\top Q x_k$ is given by*

$$M_{J_N}(\xi; \mu, \Sigma, \Omega) = \frac{\exp\Big(\frac{1}{2}\mu^\top \Big((\mathrm{I} - 2\xi\Omega\Sigma)^{-1} - \mathrm{I}\Big)\Sigma^{-1}\mu\Big)}{\sqrt{\det(\mathrm{I} - 2\xi\Omega\Sigma)}}, \tag{4.10}$$

where $\xi \in \Big[-\infty, \frac{1}{2\lambda_{\max}(\Omega\Sigma)}\Big)$ and $\Omega = \operatorname{diag}(Q, \dots, Q)$ with weight matrix Q.

Proof. It is a known fact that there exits an $m \times m$ matrix T such that $\det T \neq 0$, $T^\top \Sigma^{-1} T = \mathrm{I}$, $T^\top \Omega T = \Lambda$, and $\Sigma\Omega = T^{-\top}\Lambda T^\top$, where Λ has the eigenvalues λ_i of $\Sigma\Omega$ on the diagonal, given that Σ and Ω are symmetric and Σ is positive definite. As both Σ and Ω fulfill this requirement by definition, we can use this to obtain a transformation matrix T. Let F_z denote the cumulative distribution function of z, i.e., of a normal distribution. It then follows by definition that

$$M_{J_N}(\xi) = \mathbb{E}\left[e^{\xi z^\top \Omega z}\right] = \int_{\mathbb{R}^{Nn}} \exp\Big(\xi x^\top \Omega x\Big) \mathrm{d}F_z(x)$$

$$= \det\,(2\pi\Sigma)^{-\frac{1}{2}}\!\int \exp\!\Big(\xi x^\top \Omega x - \tfrac{1}{2}\,(x-\mu)^\top \Sigma^{-1}(x-\mu)\Big)\mathrm{d}x,$$

where $\int \mathrm{d}x$ is an m-fold integral over the domain of z, i.e., $\mathbb{R}^{Nn}$. Applying the transformation $x = Ty + \mu$ with $c = T^{-1}\mu = (c_1, \dots, c_m)$, we rewrite as

$$\begin{aligned} M_{J_N}(\xi) &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(\xi\lambda_i(y_i + c_i)^2 - \frac{1}{2}y_i^2\right)\mathrm{d}y_i \\ &= \left[\prod_{i=1}^{m} \frac{1}{1-2\xi\lambda_i}\right] \cdot \left[\exp \sum_{i=1}^{m} \frac{1}{2} c_i^2 \frac{2\xi\lambda_i}{1-2\xi\lambda_i}\right] \\ &= \frac{\exp\Big(\frac{1}{2}\mu^\top \Big((\mathrm{I} - 2\xi\Omega\Sigma)^{-1} - \mathrm{I}\Big)\Sigma^{-1}\mu\Big)}{\sqrt{\det(\mathrm{I} - 2\xi\Omega\Sigma)}} \end{aligned}$$

□

By Assumption 11 and 12, we have $\mu = 0$ and $\Sigma = \text{const}$. This yields the simplified form

$$M_{J_N}(\xi; \Sigma, \Omega) = \det(\mathrm{I} - 2\xi\Omega\Sigma)^{-\frac{1}{2}}\,, \tag{4.11}$$

which is time-invariant as it only depends on constant model parameters. Thus, it is also well suited for the design of a learning trigger.

Based on the MGF, it is straightforward to compute the moments.

Lemma 14. *For $\mathbb{E}[x] = 0$, the expected value and variance for the cost J_N of a trajectory x, as derived from the moment-generating function $M_{J_N}(\xi)$, are given by*

$$\mathbb{E}\big[J_N\big] = \frac{\mathrm{d}}{\mathrm{d}\xi} M_{J_N}(\xi)\Big|_{\xi=0} = \operatorname{trace}\Omega\Sigma \tag{4.12a}$$

$$\mathbb{E}\big[J_N^2\big] = \frac{\mathrm{d}^2}{\mathrm{d}\xi^2} M_{J_N}(\xi)\Big|_{\xi=0} = 2\operatorname{trace}(\Omega\Sigma)^2 + \operatorname{trace}^2\Omega\Sigma\,. \tag{4.12b}$$

Proof. The result follows from using Jacobi's formula (Magnus and Neudecker, 1999, Sec. 8.3)

$$\tfrac{\mathrm{d}}{\mathrm{d}t} \det A(t) = \det A(t)\operatorname{trace}\Big(A^{-1}\tfrac{\mathrm{d}}{\mathrm{d}t}A(t)\Big)$$

to compute the derivatives of the moment-generating function. □

4.3.2 Chernoff Trigger

In order to obtain an effective trigger with theoretical guarantees, we need sophisticated concentration results, which make use of the whole distribution.

Next, we introduce the Chernoff bound and utilize it to derive the trigger threshold κ.

Theorem 15 (Chernoff Bound (Chernoff, 1952, Thm. 1))**.** *Given the moment-generating function $\mathbb{E}\left[e^{\xi X}\right]$ of the random variable X, for any real number $\xi > 0$, it holds that*

$$\mathbb{P}[X \geq \kappa] \leq \frac{M_X(\xi)}{e^{\xi\kappa}} \qquad \mathbb{P}[X \leq \kappa] \leq \frac{M_X(-\xi)}{e^{-\xi\kappa}}. \tag{4.13}$$

In particular, it holds that

$$\mathbb{P}[X \geq \kappa] \leq \inf_{\xi>0} \frac{M_X(\xi)}{e^{\xi\kappa}} \qquad \mathbb{P}[X \leq \kappa] \leq \inf_{\xi<0} \frac{M_X(\xi)}{e^{\xi\kappa}}. \tag{4.14}$$

Proof. Follows from Markov's inequality applied to $e^{\xi X}$. □

Remark 16. *These bounds are often specialized to sums of independent random variables. We avoid this independence requirement, by considering the MGF of the entire sum as a random variable. This enables us to tailor the Chenoff trigger to the full distribution of J_N taking any dependence over the horizon into account. This is in contrast to the moment-based trigger designed in the next section.*

Thus, we can state the main theorem of this part, which is the full distributional analog to Thm. 26.

Theorem 17 (Chernoff Trigger)**.** *Let the parameter $N \in \mathbb{N}$ be given and Assumptions 11 and 12 hold. Then, we can obtain for any time-index k an upper bound $\alpha \in (0, 1)$ for the probability*

$$\mathbb{P}\Big[J_N(k) \notin \big(\kappa^-, \kappa^+\big)\Big] \leq \alpha, \tag{4.15}$$

where the thresholds are chosen in the following

$$\kappa^+ = \inf_{\xi\in\left(0,\frac{1}{2\lambda_{\max}}\right)} \chi(\xi) \qquad \kappa^- = \sup_{\xi\in(-\infty,0)} \chi(\xi) \tag{4.16}$$

$$\chi(\xi) = -\tfrac{1}{\xi}\ln\tfrac{\alpha}{2} - \tfrac{1}{2\xi}\ln\det(\mathrm{I} - 2\xi\Omega\Sigma) \tag{4.17a}$$

$$= -\frac{1}{\xi}\ln\frac{\alpha}{2} - \frac{1}{2\xi}\sum_{j=0}^{Nn}\ln(1-2\xi\lambda_j)\,. \tag{4.17b}$$

Further, λ_j are the eigenvalues of $\Omega\Sigma$, the state covariance matrix is denoted as Σ (as introduced in Lemma 24), and the weight matrix $\Omega = \operatorname{diag}(Q, \ldots, Q)$.

Proof. We distribute the tail probability α symmetrically to both sides of the interval. Thus,

$$\inf_{\xi>0} \frac{M_X(\xi)}{e^{\xi\kappa^+}} = \inf_{\xi<0} \frac{M_X(\xi)}{e^{\xi\kappa^-}} = \frac{\alpha}{2},$$

which has to be solved for $\kappa^\pm$. For κ^+, we get

$$\frac{\alpha}{2} = \inf_{\xi>0} \frac{M_X(\xi)}{e^{\xi\kappa^+}}$$

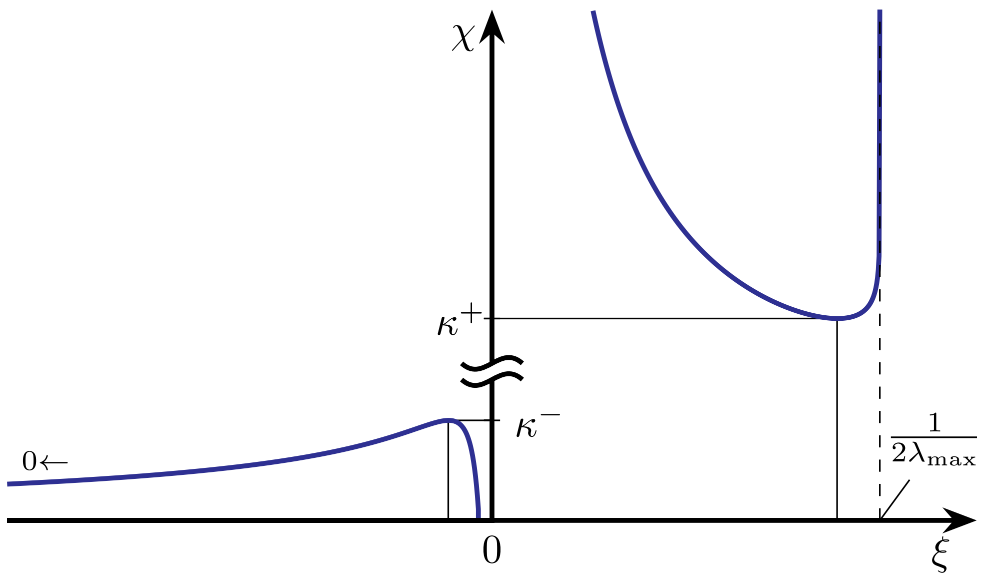


Figure 4.2: Illustration of the shape of the function $\chi(\xi)$, which has to be optimized for the Chernoff trigger. The scales of the left- and right-hand side of the graph differ and have been adjusted for better visualization. The bounds of the Chernoff trigger, κ^- and κ^+, are found through straightforward maximization (over $\xi < 0$) and minimization ($\xi > 0$), respectively.

$$
\begin{aligned}
&\Leftrightarrow \quad 0 = \inf_{\xi>0} \ln M_X(\xi) - \xi\kappa^+ - \ln \tfrac{\alpha}{2} \qquad \Big| \div \xi \\
&\Leftrightarrow \quad \kappa^+ = \inf_{\xi>0} \tfrac{1}{\xi} \ln M_X(\xi) - \tfrac{1}{\xi} \ln \tfrac{\alpha}{2}
\end{aligned}
$$

We can proceed similarly for κ^-, just that the infimum turns into a supremum, when we divide by ξ as $\xi < 0$. By inserting the simplified MGF from (4.11) into the equation, we obtain the statement. □

Next, we introduce the trigger design, discuss the main advantages, and, finally, elaborate on how to obtain the thresholds $\kappa^\pm$.

The Chernoff trigger is defined as follows:

$$\hat{J}_N(j) \notin \left(\kappa^-, \kappa^+\right) \Leftrightarrow \gamma_{\text{learn}} = 1, \tag{4.18}$$

with $\kappa^\pm$ as introduced in Thm. 17.

In order to obtain the thresholds $\kappa^\pm$, we need to solve the two optimization problems (4.16). However, this is easily tractable online due to the following properties of the objective function χ. Intuitively this can also be seen in Fig. 4.2, where the general shape of the objective function is illustrated.

Theorem 18. *The function $\chi(\xi)$ is strictly convex in the range for κ^+ and thus has only one minimum on the interval.*

Proof. We first consider the strict convexity on the interval $0 < \xi < \frac{1}{2\lambda_{\max}}$. For all ξ from that interval, $\frac{1}{\xi}$ is convex, thus also $-\frac{1}{\xi}\ln\frac{\alpha}{2}$ is convex as $\alpha \in (0,1)$ implies that the logarithm is negative. The second derivative of the second part is

$$\frac{\mathrm{d}^2}{\mathrm{d}\xi^2}\left[-\frac{\ln(1-2\xi\lambda_j)}{2\xi}\right] = -\frac{\ln(1-2\xi\lambda_j)}{\xi^3} + \frac{2\lambda_j}{\xi^2\,(1-2\xi\lambda_j)} + \frac{4\lambda_j^2}{2\xi\,(1-2\xi\lambda_j)^2}.$$

In the considered interval, we have $\xi > 0$, $0 < 1-2\xi\lambda_j < 1$ and $\ln(1-2\xi\lambda_j) < 0$, therefore, the second derivative is positive in this range. Hence, all summands of $\chi(\xi)$ are strictly convex. Thus, $\chi(\xi)$ is strictly convex on the interval. This immediately implies that there is only one minimum on the interval. $\square$

Remark 19. *Even if the optimization does not yield the optimal value, any sub-optimal value will still fulfill the Chernoff bound. Thus the trigger remains valid, just with a more conservative threshold.*

4.4 Mean-based Learning Trigger

The previous design considers only a single sample of the cost function. Intuitively one may want to use moving averages and higher moments of the cost since it is a stochastic process. Similar ideas, which are based on the expected values and Hoeffding's inequality were presented in Solowjow and Trimpe (2020); Gatsis and Pappas (2018); Baumann et al. (2019). While conceptually simpler, we illustrate that moment-based triggers involve theoretical obstacles and limitations arising from this concept.

The idea for this trigger is to derive a threshold κ on the deviation from the expected value $\mathbb{E}[J_N]$, leading to

$$\left|\sum\nolimits_{j\in\mathfrak{L}(k)} \hat{J}_N(j) - L\,\mathbb{E}[J_N]\right| \geq \kappa \Leftrightarrow \gamma_{\text{learn}} = 1\,, \tag{4.19}$$

where $\mathfrak{L}(k)$ is a summation set of cardinality L that achieves approximately uncorrelated samples and will be discussed later (cf. (4.23)). We will first provide a derivation of the expected value, then discuss how to design the threshold κ in order to obtain a confidence interval corresponding to a given probability.

4.4.1 Expected Value

Analogously to the continuous-time solution provided in Bijl et al. (2016), we will next derive the expected value of the cost $\mathbb{E}[J_N]$ for the discrete-time case.

Lemma 20 (Expected Cost)**.** *Under Assumption 11 and 12, the expected value for the cost J_N with respect to the system (4.5) is given by*

$$\begin{aligned}\mathbb{E}[J_N] &= \mathbb{E}\left[\sum_{k=0}^{N-1} x_k^\top Q x_k\right] = \operatorname{trace}\left(\sum_{k=0}^{N-1} S_k Q\right)\\ &= \operatorname{trace}\left(\left(S_0 - S_N + N\Sigma_x\right)\bar{X}^Q\right)\\ &= \operatorname{trace}\left(N\Sigma_x \bar{X}^Q\right), \end{aligned} \tag{4.20}$$

with $S_k = \mathbb{E}\left[x_k x_k^\top\right]$, and $\bar{X}^Q$ the solution to the Lyapunov equation $A^\top \bar{X}^Q A - \bar{X}^Q + Q = 0$.

Proof. We first note that $\mathbb{E}[J_N] = \mathbb{E}\left[\sum_{k=0}^{N-1} x_k^\top Q x_k\right] = \mathbb{E}\left[\sum_{k=0}^{N-1} \operatorname{trace}\left(x_k x_k^\top Q\right)\right] = \operatorname{trace}\left(\sum_{k=0}^{N-1} \mathbb{E}\left[x_k x_k^\top\right] Q\right) = \operatorname{trace}\left(\sum_{k=0}^{N-1} S_k Q\right)$. Then, let $Y(N) := \sum_{k=0}^{N-1} S_k$. Next, we can find $Y(N)$ as the solution to a discrete Lyapunov equation by reordering the difference of initial and final second moment

$$\begin{aligned} S_N - S_0 &= \sum_{k=0}^{N-1} (S_{k+1} - S_k) = \sum_{k=0}^{N-1} \left(A S_k A^\top - S_k + \Sigma_x\right)\\ &= A\left(\sum_{k=0}^{N-1} S_k\right)A^\top - \left(\sum_{k=0}^{N-1} S_k\right) + \sum_{k=0}^{N-1} \Sigma_x\\ &= AY(N)A^\top - Y(N) + N\Sigma_x. \end{aligned}$$

One can show by substituting the Lyapunov equations, that $\mathbb{E}[J_N] = \operatorname{trace}(Y(N)Q) = \operatorname{trace}\left(\left(S_0 - S_N + N\Sigma_x\right)\bar{X}^Q\right)$ with $Y(N)$ and $\bar{X}^Q$ being the solution to $0 = AY(N)A^\top - Y(N) + S_0 - S_N + N\Sigma_x$ and $0 = A^\top \bar{X}^Q A - \bar{X}^Q + Q$. With Assumption 12 the covariance is time-invariant, thus the result simplifies to $\mathbb{E}[J_N] = \operatorname{trace}\left(N\Sigma_x\bar{X}^Q\right)$. □

This result is equivalent to the previous result from Lemma 14 for the first moment. However, here we avoid explicitly computing Σ, which is more computationally efficient for individual moments than the previous direct solution, where Σ is explicit.

Next, we will consider how to design the threshold κ.

4.4.2 Statistical Guarantees

The trigger (4.19) leverages the fact that the empirical mean converges to the expected value. Even for finite sample sizes, it is possible to quantify the expected deviation, which can be done with the well-known Hoeffding inequality. The trigger threshold κ can be regarded as a confidence bound, i.e., it is chosen such that with confidence level α, the deviation term does not exceed κ. Therefore, observing deviations larger than κ can not be sufficiently explained by noise and random fluctuations. Thus, we trigger model learning whenever this happens.

Theorem 21 (Hoeffding's Inequality (Hoeffding, 1963, Thm. 2))**.** *Assume $X_1, X_2, \ldots, X_n$ are independent random variables and $a_i \leq X_i \leq b_i$ $(i = 1, 2, \ldots, n)$, then we obtain for all $\kappa > 0$*

$$\mathbb{P}\left[\sum_{i=1}^{n} X_i - n\mu \geq \kappa\right] \leq \exp\left(\frac{-2\kappa^2}{\sum_{i=1}^{n} (b_i - a_i)^2}\right). \tag{4.21}$$

Comparing with (4.19), the cost samples $\hat{J}_N$ corresponds to the random variables X_i and $\mathbb{E}[J_N]$ to the mean μ. However, there are two challenges when applying Hoeffding directly in this way. First, J_N is unbounded, as it is directly influenced by Gaussian noise. Second, the cost samples are not independent, as they are part of the same state trajectory. In the following, we introduce two modifications to cope with these issues and make Hoeffding's inequality applicable.

In order to obtain an upper bound on J_N, we will assume state constraints and, for the sake of simplicity, we shall assume linear constraints.

Lemma 22. *Assume the states are constrained by $\|W^{-1}x_k\| < \rho$ for all k, where $W \in \mathbb{R}^{n\times n}$ is invertible. Then, the cost function J_N is bounded by*

$$0 \leq J_N \leq \sup_{\|W^{-1}x_k\|_2 < \rho} J_N = \rho^2 N \lambda_{\max}\left(W^\top Q W\right). \tag{4.22}$$

Proof. The lower bound follows immediately from the positive definiteness of Q, as $x^\top Q x \geq 0$ for all x. For the upper bound we use the convexity of the cost function. The supremum of a convex function on an open set is attained at the maximum on the boundary. Hence,

$$\begin{aligned} J_N \leq \sup_{\|W^{-1}x_k\|_2 < \rho} J_N &= \max_{\|W^{-1}x_k\|_2 = \rho} \left[\sum_{k=0}^{N-1} x_k^\top Q x_k\right] \\ &= N \max_{\|y\|_2 = \rho} \left\|\sqrt{Q}\, W y\right\|_2^2 = \rho^2 N \left\|\sqrt{Q}\, W\right\|_2^2 \\ &= \rho^2 N \lambda_{\max}\left(W^\top Q W\right). \end{aligned}$$

□

Remark 23. *Even for naturally unconstrained system, considering Assumptions 11 and 12, it is reasonable to assume that the state stays within some sufficiently large, but finite, region around the origin.*

Next, we investigate how to cope with the dependence in the cost samples. First, we note that consecutive samples $J_N(k-1)$ and $J_N(k)$ are dependent, as they overlap in the states they sum over. Also, adjacent sample $J_N(k-N)$ and $J_N(k)$ are dependent, since the first state in $J_N(k)$ just follow the last state in $J_N(k-N)$. In order to find *approximately independent* samples $J_N(j)$, we first need to consider the correlation between states in a trajectory.

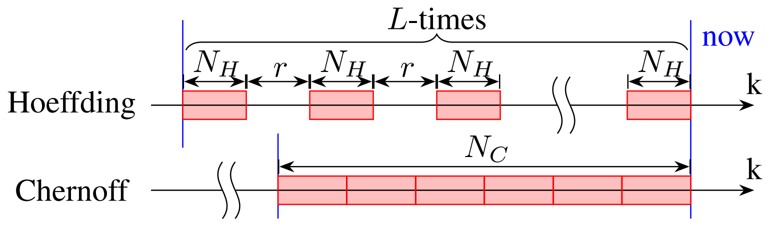


Figure 4.3: Sampling intervals for Hoeffding (Sec. 4.4) and Chernoff trigger (Sec. 4.3). Over each red interval, the quadratic cost (4.6) is computed yielding the sampled $\hat{J}_N(j)$, while the data in-between remains unused. Hence, the Hoeffding discards a significant part of the collected data, in order to ensure approximate independence between samples of J_N. In contrast to this, the Chernoff trigger uses all data points by taking a single cost sample over a longer horizon.

Lemma 24. *By Assumption 12 we have $x_0 \sim \mathcal{N}\left(0, X^V\right)$. Then, the joint distribution of a sequence of states $(x_0, x_1, \dots, x_N)$ is a multivariate Gaussian distribution with mean $\mu = 0$ and symmetric block-Toeplitz covariance matrix*

$$\Sigma = \begin{pmatrix} X^V & X^V A^\top & X^V (A^2)^\top & \cdots & X^V (A^N)^\top \\ AX^V & X^V & X^V A^\top & \ddots & X^V (A^{N-1})^\top \\ A^2 X^V & AX^V & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & X^V & X^V A^\top \\ A^N X^V & \cdots & A^2 X^V & AX^V & X^V \end{pmatrix}.$$

Proof. The covariance X^V is invariant under the system equation, thus $\mathbb{E}[x_i x_i^\top] = X^V$ for all $i = 0, \dots, N$. Computing the cross-covariance for two states x_i and x_j, for $i < j$ yields $\mathbb{E}[x_i x_j^\top] = A^{i-j} X^V$. As the joint distribution over multivariate Gaussians, i.e., the states, is also multivariate Gaussian, the statement follows. □

Lemma 25. *Under Assumptions 11 and 12, and with arbitrarily small $\varepsilon > 0$, there exist an r_0 such that $\left|\left[A^{r_0} X^V\right]_{i,j}\right| < \varepsilon$ for all matrix-entries (i, j). Hence, for any large enough $r > r_0$ the state x_k is approximately independent from the state x_{k-r}.*

Proof. Using Lemma 24, we obtain $\mathbb{E}\left[x_k x_{k-r_0}^\top\right] = A^{r_0} X^V$ as the cross-covariance for the jointly multivariate normal distributed states. For multivariate normal distributions, we have that zero cross-covariance is equivalent to independence. Since, as A is Schur-stable by Assumption 11, i.e., $\lambda_{\max}(A) < 1$, the term A^{r_0} approaches zeros as $r_0 \to \infty$ (Oldenburger, 1940). Hence, by definition of the limit, there exists an r_0 such that the absolute value of the cross-covariance is elementwise smaller than ε. Furthermore, the same holds trivially true for any $r > r_0$. Therefore, the states from the same trajectory with distance r are approximately independent. □

Thus, we ensure approximately independent samples by waiting for r data points between each N data points long cost sample (cf. Fig. 4.3). The horizon of the last cost-sample ends at the current time step, which allows the trigger to take the most recent data into account. Therefore, we obtain the summation set

$$\mathfrak{L}(k) := \{k - (N+r)i | i \in 0, \dots, L-1\} \tag{4.23}$$

for the sample average in (4.19), which by construction has cardinality L. Considering the definition of the cost (4.6), the indices in $\mathfrak{L}(k)$ mark the end of each red interval in Fig. 4.3. In total, the trigger needs the last $L(N+r)-r$ states as data, of which it only uses LN for approximating the mean.

Strictly speaking, approximate independence still does not allow us to apply Hoeffding (Thm. 21). Therefore, for the time being, we assume the approximation is exact and then apply Hoeffding to obtain the thresholds. The previously introduced Chernoff trigger (Sec. 4.3) solved this issue in an elegant and clean way by incorporating the correlations directly into the trigger via the MGF.

Theorem 26 (Hoeffding Trigger)**.** *Let Assumptions 11 and 12 hold, assume the conditions of Lemma 22 are satisfied, and the samples $J_N(k-(N+r)i), i = 0, \dots, L-1$ are mutually independent. Further, let α denote the desired confidence level and κ is chosen as*

$$\kappa = \sup_j \{J_N(j)\} \sqrt{-\frac{L}{2} \ln \frac{\alpha}{2}} = \rho^2 N \lambda_{\max}\left(W^\top Q W\right) \sqrt{-\frac{L}{2} \ln \frac{\alpha}{2}}. \tag{4.24}$$

Then, the probability of triggering with (4.19), while the model coincides with the ground truth, is bounded by

$$\mathbb{P}\left[\left|\sum_{i=0}^{L-1} J_N(k-(N+r)i) - L\,\mathbb{E}[J_N]\right| \geq \kappa\right] \leq \alpha. \tag{4.25}$$

Proof. By construction, we can apply Hoeffding's inequality (Thm. 21) to J_N at the sampling instances $\mathfrak{L}(k)$. The bound is given by Lemma 22 as $b_i \equiv 0$ and $a_i \equiv \sup J_N$. As the same inequality can also be applied to $-J_N$, we get the combined inequality

$$\mathbb{P}\left[\left|\sum_{i=0}^{L-1} J_N(k-(N+r)i) - L\mu\right| \geq \kappa\right] \leq 2\exp\left(\frac{-2\kappa^2}{L\left(\sup J_N\right)^2}\right) = \alpha.$$

We set α to coincide with the upper bound and rearrange for κ. Thus, we obtain

$$\alpha = 2\exp\left(\frac{-2\kappa^2}{L\left(\sup J_N\right)^2}\right)$$
$$\kappa^2 = -\left(\sup J_N\right)^2 \tfrac{L}{2} \ln \tfrac{\alpha}{2}.$$

Then, the result is obtained by taking the square root and inserting the value for $\sup J_N$ from the Lemma 22. □

In practice, r_0 is chosen large enough so that Lemma 25 ensures approximate independence of the samples $J_N(k-(N+r)i)$. Thus, we analyze if the empirical mean actually converges to the analytically derived expected value, while the technical details ensure that we avoid distorting the cost with random short term effects.

Remark 27. *Following the same principles, it is possible to derive alternative triggers that consider different error terms or higher moments. However, we did not observe any improved performance of such triggers compared to the mean-based Hoeffding trigger. Considering relative instead of absolute errors is also possible, however, it does not improve the triggering behavior. We confirmed this in numerical investigations (analogous to Sec. 4.5) for a variance-based Hoeffding trigger, with similar theoretical guarantees, and showed that it yields no significant advantage over the mean trigger. The Chernoff trigger does not suffer from the same limitations as the Hoeffding-based design.*

4.5 Numerical Simulation

Next, we will numerically study the trigger architecture, as shown in Fig. 4.1. We will illustrate the triggering behavior and, in particular, investigate how well model change is detected with each trigger.

4.5.1 Setup

Initially, a 5 dimensional system $(A_\mathrm{o}, B, \Sigma_x)$ is randomly generated, by sampling the matrices $A_\mathrm{o} - \mathrm{I} \in \mathbb{R}^{5\times 5}$, $B \in \mathbb{R}^{5\times 1}$, and $\sqrt{\Sigma_x} \in \mathbb{R}^{5\times 5}$ elementwise from a uniform distribution between ± 1. The initial state is sampled from the asymptotic distribution of the closed-loop system, in order to fulfill Assumption 12.

Next, we introduce the model $(\tilde{A}_\mathrm{o}, \tilde{B}, \tilde{\Sigma}_x)$, which is used to compute the feedback controller and to derive the triggering thresholds. Initially, we set the model to the exact system parameters in order to demonstrate that the cost behaves as expected. Later on, we will distort the system dynamics $(A_\mathrm{o}, B, \Sigma_x)$ to create a gap between model and true system parameters. For the model-based controller, we use LQR state feedback with unity weight matrices.

The system is simulated for 50 000 time steps. At each time step, the cost and trigger value is computed as described below for each trigger. If the trigger detects a system change, then the model is set to the true parameters, i.e., $(\tilde{A}_\mathrm{o}, \tilde{B}, \tilde{\Sigma}_x) \leftarrow (A_\mathrm{o}, B, \Sigma_x)$. Thus, we abstract for the time being the actual model learning to setting the model parameters to the true values. While this, of course, is not possible in reality, for the simulation we are for now mainly interested in the behavior of the trigger. The learning part will be considered later in Sec. 4.6.

In order to simulate system changes, which the trigger should detect, we alter the system every 10 000 time steps without adjusting the model, trigger, nor controller. First, we tried sampling the new system dynamics (A'_o, B', Σ'_x) exactly the same way as for the initial system. However, these changes are usually quite significant and easy to detect. Thus, we bound the change with the aid of an additive model increment

$$\Delta = \beta \frac{(A'_\mathrm{o}, B', \Sigma'_x) - (A_\mathrm{o}, B, \Sigma_x)}{\|(A'_\mathrm{o}, B', \Sigma'_x) - (A_\mathrm{o}, B, \Sigma_x)\|_2}, \tag{4.26}$$

where $\beta \in (-0.1, 0.1)$ is also sampled from an uniform distribution. Thus, the new system is obtained by adding Δ to the old system. If the resulting system is uncontrollable, a different increment is generated by sampling again. We do not enforce stability after altering the system since any threshold κ^+ will be reached eventually, and thus, triggering is trivial when the system is unstable.

The Chernoff trigger is computed from the model as described in (4.18) with a horizon of $N = 200$ and $\alpha = 1\%$. A system change is detected, when the trigger value $\psi = J_N(\cdot)$ leaves the interval (κ^-, κ^+). With the resulting model update, we have to recompute the trigger thresholds $\kappa^\pm$.

The Hoeffding trigger uses the simpler design (4.19) with the increased misfire probability $\alpha = 25\%$. For the moving average sampling we use $N = r = 60$ and $L = 20$ (cf. Fig. 4.3). The adjustment of N and α compared to the Chernoff trigger are required for detection with this trigger due to the significant conservatism and theoretical shortcomings of the design. The required state-bound W is set to the covariance of the state and $\rho = 18$. These bounds are constant throughout the simulation. Thus, the threshold κ remains constant as well, while the mean $\mathbb{E}[J_N]$ is the only part of the trigger that changes with model updates.

For both triggers, we use the same random seed allowing for direct comparison of the result as shown in Fig. 4.5 and Fig. 4.4.

4.5.2 Results

Next, we look at the numerical performance for both triggers. The shown roll-out was pick as it displays many interesting effects observed throughout various roll-outs and is a good representation of the observed behavior. In this run neither instability, significant violation of the upper bound on the cost for the Hoeffding trigger, nor the issues that arise from the inactive trigger after a detection are shown. Their effects are obvious and observed in other roll-outs.

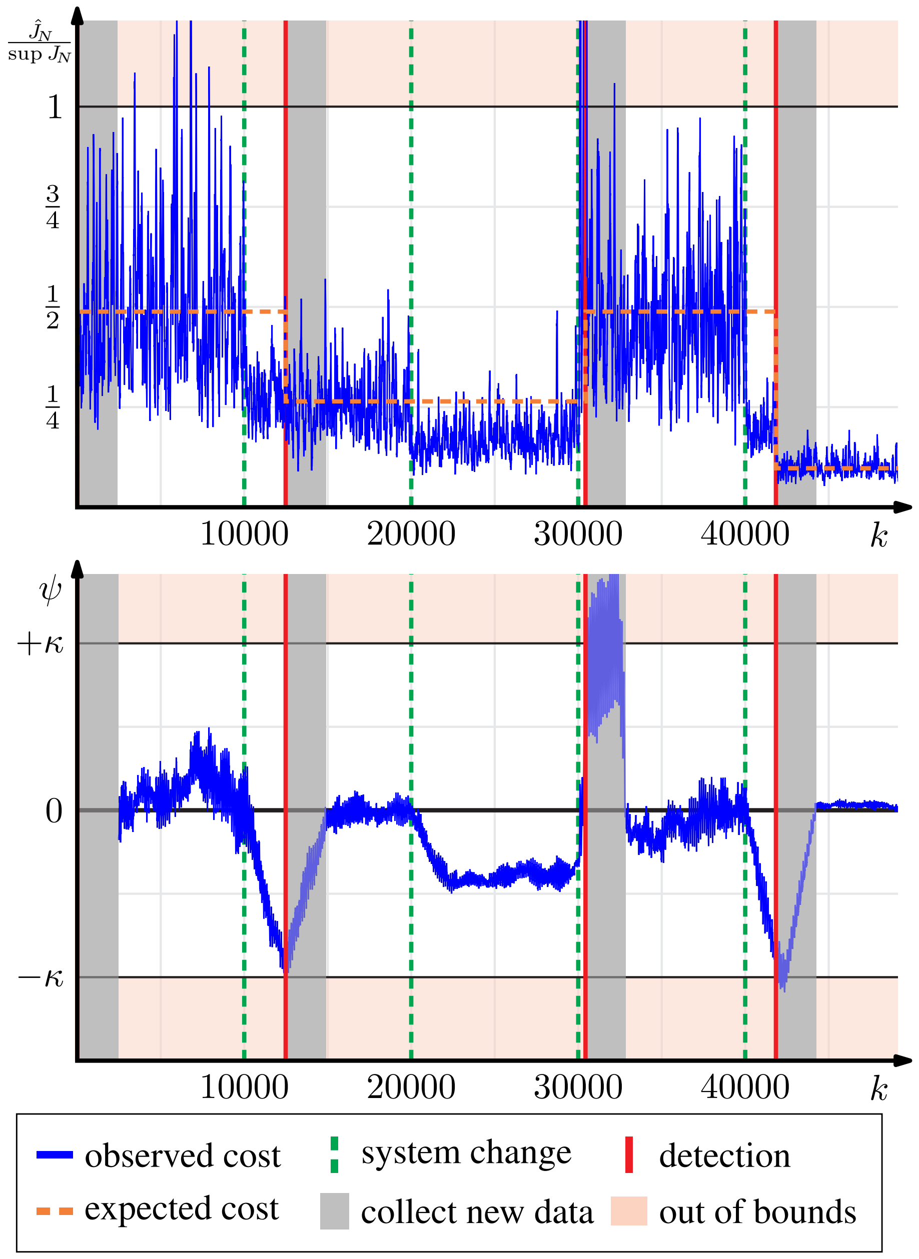


Figure 4.4: Numerical simulation of the Hoeffding trigger on a randomly generated 5-dimensional system. In the lower graph the trigger statistic ψ is shown (blue line). Every 10 000 time steps (green lines), the system is randomly altered in order to simulate change. Leaving the confidence bounds $(-\kappa, \kappa)$ triggers learning (red line). Then, the model is set to the true system parameters, a new feedback gain computed, and a new value for $\mathbb{E}[J_N]$ derived. In the upper graph, the normalized cost is shown (in blue) and the model-based expected value $\mathbb{E}[J_N]$ (orange line).

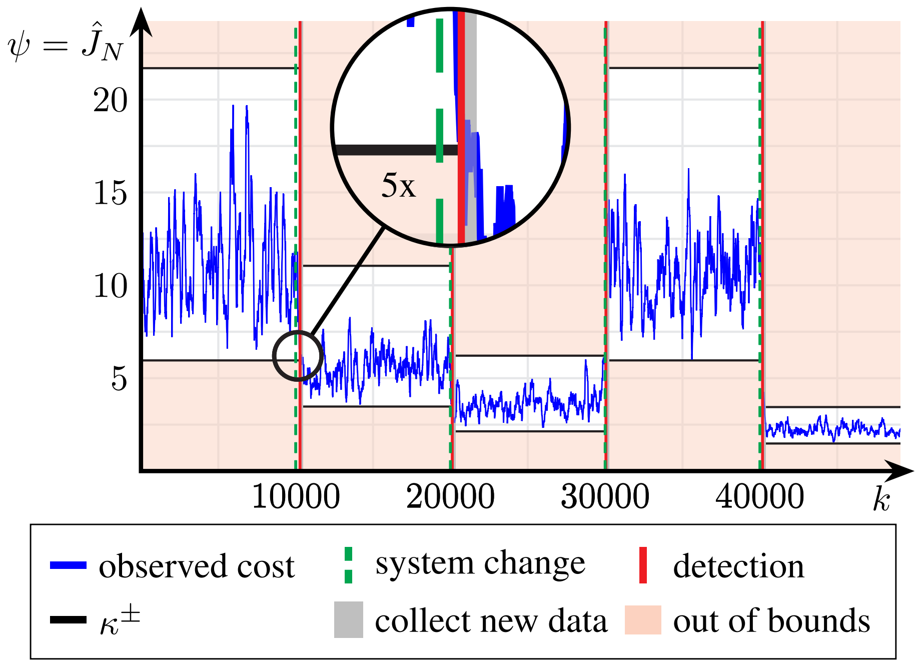


Figure 4.5: Numerical simulation of the Chernoff trigger on a 5-dimensional system with random A_o and B matrices. At the indicated time step (green), the entries of the A and B-matrices are randomly altered in order to simulate a change in the dynamics. This change is detected at the red lines, at which point the model, the feedback controller, and the thresholds $\kappa^{\pm}$ are updated.

In the upper plot of Fig. 4.4, we see the normalized cost, and in the lower one, the trigger statistic ψ for the Hoeffding trigger. The Chernoff trigger requires only a single plot in Fig. 4.5, since it utilizes the cost $\hat{J}_N$ directly as trigger statistic ψ. The system is distorted every 10 000 time steps (green dashed line). A detection (red line) occurs when the trigger value ψ (in blue) hits either threshold κ, which can be seen in the (lower) plot for trigger statistic. Recall that each trigger uses a different horizon for the cost, hence due to the longer horizon, the cost for the Chernoff trigger looks smoother.

Foremost, this example illustrates the shortcomings of the mean-based design in contrast to the normal operation of the Chernoff trigger. In the following, we consider the individual system changes and their detection in detail.

At $k = 10\,000$, the first system change occurs, which is detected after 2 480 steps with the Hoeffding trigger. This is a significant delay between changing the dynamics and detecting said change. However, these transient effects are not surprising due to the large amount of data required by the trigger with its window of length $L(N+r) = 2400$. Thus, it might take some time (cf. Fig. 4.3) until the new dynamics affect the entire horizon of the moving average. This effect can also be observed at $k = 40\,000$, where the detection takes 1 838 steps. The Chernoff trigger, in contrast, does not suffer from this issue. While the achieved detection times of between 50 and 280 steps in this simulation are only possible due to the better bounds, of course, the short window of states considered also expedites the detection.

The Hoeffding trigger does not detect the change at $k = 20\,000$, which demonstrates the downsides of this trigger design. The upper bound is too conservative, and thus, the change cannot be detected with the given amount of data. Considering more data would be possible, however, it would also increase the delays even further. Smaller upper bounds are not possible since these were designed for the initial system by hand. In principle, it would be possible to change the bounds during triggering, however, it is not trivial to do so automatically. The Chernoff is designed to deal with possible unbounded cost, insofar the order of magnitude of the cost has no impact on the triggering performance.

At $k = 30\,000$, there is a fast increase of the cost and a lot of deviation within the signal. This allows even the Hoeffding trigger a fast detection after only 435 steps. Yet, the Chernoff trigger is still significantly faster in detecting the change within 50 steps.

Secondly, for the Chernoff trigger, we can see that the bounds are tight in the sense that the cost stays within the confidence interval, but also comes close to the edges. Thus, the probability mass is distributed as intended. Even though the trigger uses little data, it rarely misfires – not once in this run. In the next section, we will present a large scale experiment to further investigate false positives and trigger delays. We obtain a misfire rate of less than 0.01% over the simulated four billion time steps, which is, as designed, less than $\alpha = 1\%$.

4.5.3 Discussion

Hoeffding Trigger

First of all, the Hoeffding trigger does a decent job at detecting change. However, there are some downsides and limitations.

The large amount of data required by the trigger affects detection. Clearly, there is a significant delay in the detection, which corresponds to the magnitude of the time window. Further, it prevents the detection of quick changes since new data has to be gathered after each model update.

Furthermore, bounding the cost derived from Lemma 22 is an issue. In order to apply Hoeffding's inequality rigorously, we need the bound to hold everywhere. However, this has a significant impact on the detection rate since the possibility of high costs increases the possible confidence interval. Additionally, Hoeffding's inequality is per se based upon the worst-case distribution and, thus, not very tight.

Nonetheless, there is sufficient information in the cost signal to detect changes in the system dynamics reliably. Yet, this trigger only exploits a small part of the available data and, hence, it only achieves sub-optimal detection times and misses some changes.

Chernoff Trigger

Since the trigger thresholds are tailored to the actual distribution, we can see a superior performance. In particular, the adaptivity of the thresholds to different magnitudes of process noise can be clearly seen in Fig. 4.5. For instance, at $k = 40\,000$, and afterward, there is little deviation in the cost, and this is also captured in the bounds. However, between $k = 0$ and $k = 10\,000$ there are strong oscillations in the signal. Nonetheless, the interval fits nicely.

Furthermore, the shorter time-window of only 200 instead of 2 400 steps results in faster and more reactive triggering. However, therein lies a trade-off with random fluctuations and unmodelled disturbances. These can have large impacts on the trigger value ψ, as they are not averaged out. Therefore, we will discuss further possibilities to robustify the trigger in Sec. 4.6, where we consider a hardware experiment.

4.5.4 Detection Delay

To study the detection delays of this trigger (i.e., the time between changing the system and the trigger detecting the change), we ran large scale Monte Carlo simulations using the same setup as before. However, we ignored unstable changes and resampled when this happened. Each roll-out was simulated an hour of wall-time before a new roll-out with a different random seed was started. The restarts are required as the used pseudo-random number generator for the noise and system changes has only limited entropy. Eight roll-outs were computed in parallel on an Intel® Xeon® W-2123 3.6 GHz 8-core processor, for a total of two weeks accumulating a total of 3 976 360 000 simulated time steps with 397 636 system changes.

A System Change Metric

Our hypothesis is that the detection delay depends on the *size* of the system change. Thus, we require a metric to quantify this. For this purpose, we compare the system norm before and after the change. Considering the stochastic nature of the problem, using the H_2-norm seemed suitable. This norm is closely linked to the steady-state covariance of the system when driven by white noise input. In detail, the H_2-norm for an input-output system $\mathcal{G}$ with input w and output y is defined as

$$\begin{aligned} \left\|\mathcal{G}\right\|_{H_2} &= \sqrt{\lim_{k\to\infty} \operatorname{trace} \mathbb{E}\Big[\, y_k^\top y_k \,\Big]}, \\ \mathbb{E}\left[w_i\right] = 0\,, \quad & \mathbb{E}\Big[\, w_i w_j^\top \,\Big] = \delta_{ij}\, \mathrm{I}\ . \end{aligned} \tag{4.27}$$

Further, we decided to measure the system change as

$$\delta_{\text{sys}} := \frac{\left\| \left[\begin{array}{c|c} A_{\text{new}} & \sqrt{\Sigma_{x,\text{new}}} \\ \hline \mathrm{I} & 0 \end{array}\right] \right\|_{H_2}}{\left\| \left[\begin{array}{c|c} A_{\text{old}} & \sqrt{\Sigma_{x,\text{old}}} \\ \hline \mathrm{I} & 0 \end{array}\right] \right\|_{H_2}}, \tag{4.28}$$

with the old and new closed-loop system matrix A and the square root of process noise covariance Σ_x. The $[\,\begin{array}{c|c} & \\ \hline & \end{array}\,]$-notation represents the system and is commonly used in the robust control community (Zhou and Doyle, 1998, Sec. 3). We use the square root of Σ_x as input since this will transform the white noise of the norm into the actual Gaussian noise.

Estimating Probability Density $\mathbb{P}\Big(T_D \Big| \delta_{sys}\Big)$

Additionally, we need to clarify how we measure the detection delay T_D. We define the delay as the number of time steps from the system change, which is instantaneous in the simulation, to the first time step, the trigger threshold is passed. In particular, we do not consider detection only after the threshold is passed for some time, as implemented on hardware (cf. Sec. 4.6).

The Monte Carlo simulation yields samples from the joint probability $\mathbb{P}\left(T_D, \delta_{\text{sys}}\right)$ of the detection delay T_D and system change δ_{sys}. From these samples, we compute an estimate for the probability density function using a Gaussian kernel smoothing with the MATLAB®-command[1] `ksdensity`[2] on $\left(T_D, \log_{10} \delta_{\text{sys}}\right)$. Applying the logarithm is beneficial from a numerical point of view.

Clearly, the system changes δ_{sys} are not uniformly distributed, and thus, the joint probability density is difficult to interpret. Hence, we condition on δ_{sys} to obtain the conditional probability $\mathbb{P}\left(T_D \middle| \delta_{\text{sys}}\right)$. In order to do that we need to compute the density function for the marginal probability $\mathbb{P}\left(\delta_{\text{sys}}\right)$, for which we again apply a Gaussian kernel smoothing with `ksdensity` on the logarithm of δ_{sys}. The conditional probability is then computed by division.

Results

In Fig. 4.6, the obtained density function for $\mathbb{P}\left(T_D \middle| \delta_{\text{sys}}\right)$ is shown. For the visualization the graph renormalized such that $\forall \vartheta : \max_\tau \left\{\mathbb{P}\left(T_D = \tau \middle| \delta_{\text{sys}} = \vartheta\right)\right\} = 1$. We can see that a change is most likely detected after N time step, as we observed earlier when considering just a single roll-out.

Since we are using a relative metric, a value of 1 implies that there was no change in the system. We can clearly see in Fig. 4.6 that the probability mass is rather concentrated for significant changes in the system (i.e., $\delta_{\text{sys}} \ll 1$ and $\delta_{\text{sys}} \gg 1$). Moving towards $\delta_{\text{sys}} = 1$, we can observe that the detection time increases and also the variance. More and more probability mass is pushed towards large detection times. Exactly at $\delta_{\text{sys}} = 1$, the triggering should be purely due to false positives. However, we did not record any data points exactly at $\delta_{\text{sys}} = 1$ since this event has probability zero. Further, there are some smoothing effects in Fig. 4.6, in particular, around $\delta_{\text{sys}} = 1$.

4.6 Hardware Experiment: Rotary Pendulum

While the previous numerical examples showed the effectiveness of the proposed triggers, we now investigate their efficacy under real-world and, thus, non-ideal conditions. We consider the pole-balancing performance of a rotary pendulum.

[1]MATLAB®, SIMULINK® are registered trademarks of The MathWorks, Inc.
[2]https://www.mathworks.com/help/stats/ksdensity.html

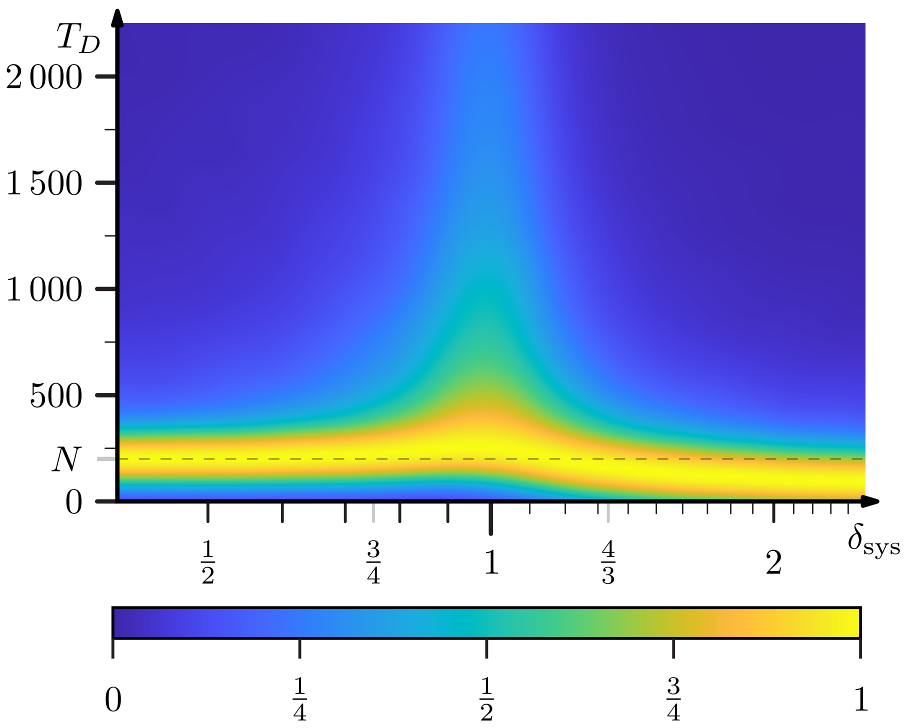


Figure 4.6: The probability density estimate $\mathbb{P}(T_D|\delta_{\text{sys}})$ of the detection delay T_D of the Chernoff trigger conditioned on the relative change δ_{sys}. The estimate is obtained with a Gaussian smoothing kernel from a Monte Carlo simulation with 397 636 samples using the setup described in Sec. 4.5.1. The maximal value of the density for any fixed δ_{sys} is normalized to one, thus the most likely delay for a given system change δ_{sys} is colored in yellow.

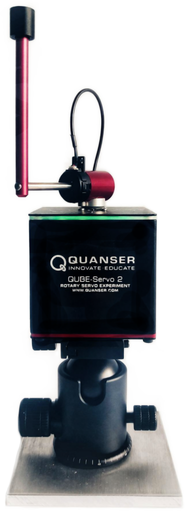


Figure 4.7: The experiment setup consists of a Quanser QUBE Servo 2 rotary pendulum that is mounted on top of a tripod. Here, the plant is shown just after swing-up.

We focus on the Chernoff trigger, which proved to be superior in the numerical experiments and has better theoretical properties.

4.6.1 Experimental Setup

We implemented the proposed learning trigger on a modified Quanser rotary pendulum (J. Apkarian, 2016), as shown in Fig. 4.7. We denote the axial rotation as θ and the angle between the vertical position and the pendulum as α. The setup allows us to directly measure both angles and the velocity in θ. For the velocity in α, we use the provided high-pass filter to approximately differentiate the angle.

The pendulum has been modified in such a way that allows for changing the dynamics in two distinct ways. First, the base of the platform is mounted on top of a tripod. Using its ball joint, we can tilt the pendulum freely in any direction. Additionally, a magnet is attached to the top end of the pendulum, which allows us to change the inertia by adding magnetic weights. By varying these two, we can change the system dynamics and validate if the trigger is able to detect these.

Using SIMULINK®[1], we implemented a switched controller running at a sampled rate of 500 Hz. An LQRI controller, i.e., an LQR with an additional integrator, is used to stabilize the upright position. While outside the approximately linear region ($\pm 20°$), a nonlinear swing-up controller is used to bring the inverted pendulum back to the linear region. Including the artificial integrator on θ as an extra state $e := \int \theta \, dt$ in the plant model yields a five dimensional system with the state $x^\top = \left[\theta, \alpha, \dot{\theta}, \dot{\alpha}, e\right]$.

4.6.2 Event-triggered Learning Design

On this linear controller, we apply the proposed event-triggered learning strategy (i.e., the Chernoff trigger (4.18)) as shown in Fig. 4.1. The empirical cost $\hat{J}_N(k)$ is computed at every time step with a horizon of $N = 200$. In order to avoid detecting an instantaneous disturbance, like for example, the jerk introduced by enacting a system change via tilting or weights, we modify the trigger slightly. We introduce the additional condition that the threshold has to stay surpassed for more than 10 seconds. Thus, we achieve more robust detection against strong short term disturbances.

When the trigger detects a change, a learning experiment is started. For this, the trigger and integrator are disabled, as they would react to the learning excitation. However, this introduces an initial disturbance, which we overcome by waiting a few seconds until the system returns to steady-state.

For the learning experiment, an artificial excitation signal is added to the control input. Choosing a signal that is both sufficiently exciting (Ljung, 2009) for possibly changed dynamics of the pendulum and avoids the hardware constraints in θ turned out to be a nontrivial problem on this experiment. In general, it is difficult to design well-behaved excitation signals a priori. Here, we apply a carefully tuned chirp signal, which first increasing and then decreasing frequency.

Learning itself is performed using prediction error minimization from the MATLAB® System Identification toolbox[1,3], with an initial guess based on least square estimation.

Due to the nonlinear, state-dependent, non-white noise of the actual pendulum, we cannot use the data to estimate the process noise directly. Instead, we record a few seconds of steady-state behavior with the new controller, including the integrator, and estimate the covariance. Thereby, we obtain a linear Gaussian approximation of the process noise around the steady-state.

With the linear model and process noise, we can compute new trigger thresholds $\kappa^{\pm}$ (cf. Thm. 17 and Equation (4.16)), which completes the model update.

4.6.3 Results

In Fig. 4.8, we can see the (measured) cost of an exemplary run of the Chernoff trigger on the hardware setup. The setup has been initialized with a sightly incorrect model. That is, the parameters of the first-principle model, provided by Quanser, have been changed slightly. Both the initial controller and the initial bounds of the trigger have been computed based on this faulty model. The main goal is to show

[3]https://de.mathworks.com/help/ident/ref/pem.html

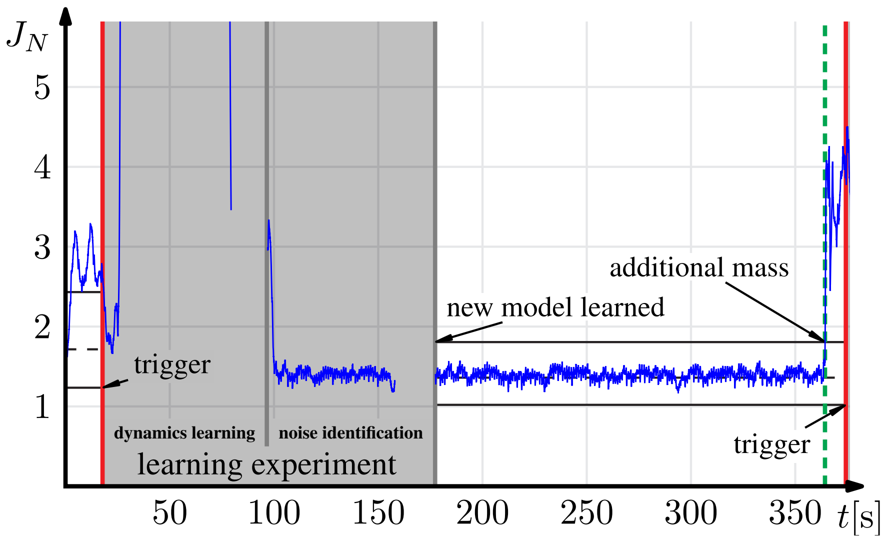


Figure 4.8: Experimental run of the Chernoff trigger on the rotary pendulum (Fig. 4.7). The black lines indicate κ^+ and κ^-, respectively. Additionally, the model-derived expected value (cf. Lemma 14) of the cost is shown as a dashed line. At the red lines, a change is detected, and thus, a learning experiment triggered. During this model update, the trigger is offline, as indicated in grey. At the green dashed line, the physical system is changed by adding a weight.

that we are able to detect change systematically and, thus, effectively reduce the cost by updating the controller.

As we can see at the very beginning of Fig. 4.8, the measured cost does not lie within the interval (κ^-, κ^+). Since we used an inaccurate model to design the feedback controller, this is to be expected. The cost quickly rises above the threshold, however, at 7.758 s, it has a down crossing caused by random effects. Hence, the change is only detected after 17.758 s.

After the model has been updated (end of the learning experiment), we see that the cost lies within the new trigger interval. Indeed, it oscillates nicely around the computed expected value.

Most importantly, we effectively reduce the cost. Before we triggered learning, the cost signal was significantly higher (roughly two times on average) than after updating the model and controller. Further, we also obtain new thresholds to detect an additional change in the dynamics.

After approximately six minutes, we add a weight to the pendulum, which is indicated by the green line in Fig. 4.8. At 374.272 s, the trigger detects this change. We want to emphasize again that this detection is not due to the initial disturbance. Instead, it is due to the change in dynamics and, thus, a different cost distribution, which we successfully detect.

4.6.4 Hardware and Implementation

Around 80 s and 160 s, there are two chunks of missing data, which is an artifact of our implementation. During these times, the updates of the system matrices and trigger thresholds $\kappa^{\pm}$ were computed. During these computations, no data was collected. However, this did not influence the controller and, thus, the presented results.

The required connection wire between the rotating sensor for measuring α and the base introduces a time-variant nonlinearity into the setup. As this wire randomly twists during operation and swing-up, the equilibrium state may change in θ. Additionally, the wire applies a force towards some θ, which may not be zero. While these effects have little impact on the controller, they pose a problem for linear system identification and especially the noise estimation. Thus, we can only consider runs, where the wire remains close to its correct state.

Also, tilting the pendulum in any direction by at least one degree yields an interesting problem. Detecting the change is straightforward with our approach. However, the new system is at least affine and does not have an upper equilibrium without input. Thus, it is challenging to identify new bounds for the trigger. Handling such a system might be possible but requires some adjustment in our approach.

4.7 Conclusion

In this chapter, we propose a Chernoff type learning trigger to trigger model learning when needed based on the distribution of LQR cost function. Thus, we obtain a highly flexible control scheme that leverages well-known results from LQR and combines them with tools from statistical learning theory. By explicitly computing the moment generating function of the LQR cost function, we are able to tailor learning triggers tightly to the problem at hand.

The derived learning triggers are extensively validated in numerical simulation and yield the expected results. Furthermore, we show in a hardware experiment that the approach can be applied to a real system, where it effectively detects the change and reduces the incurred control cost and steady-state variance.

Chapter 5

Model Predictive Control and Parameter Filtering

This chapter presents the contents of the journal publication (Schlor et al., 2022).

5.1 Introduction

Never change a running system is a popular heuristic to ensure the dependable operation of engineering systems. At the same time, learning-based techniques are becoming increasingly popular in the control community, and learning usually requires some form of exploration to the system; that is, change. Despite the popularity, success stories are still rare. One key issue are the opposing objectives of many control tasks and the requirements for successful learning outcomes. On the one hand, we usually aim for controllers that stabilize a system with the goal of avoiding deviations from a setpoint or reference. But, if a system is well regulated and barely moving, data is mainly dominated by noise. Using this data for learning is a bad idea and leads to profound theoretical issues that might result in divergence of parameters and damage to the system. Therefore, excitation is a critical requirement to learn something meaningful; however, often not desired as it deteriorates the control performance. Hence, learning permanently can be problematic. Instead, we want to detect the instances when updates are useful and – only then – effectively execute learning over a limited period of time.

The main idea of our event-triggered learning (ETL) framework is depicted in Figure 5.1. In the top left corner, we see the controlled system that, during normal operation, will have little excitation and thus yield uninformative data in most cases. The parameter filter (bottom left) estimates the system's parameters to test if the model utilized for control is still reliable. So far, there is no learning involved.

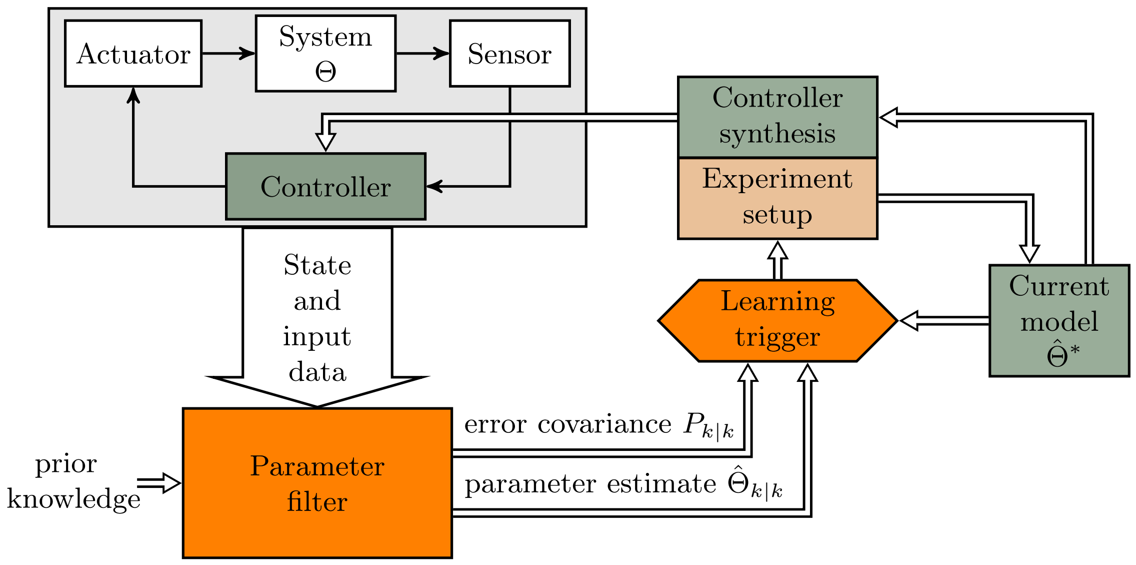


Figure 5.1: The figure schematically depicts the event-triggered learning framework. A model is used to synthesize a controller, which is applied to the system. The parameter filter outputs current estimates of model parameters with its uncertainty, which is obtained from the measured data. The learning trigger compares the current model belief with the parameter filter estimate of the model. Updates are only triggered if there is a significant discrepancy between them.

Changes in the dynamics or environment might occur naturally and deteriorate the control performance or lead to constraint violations. When there is an actual change in the system's behavior, the learning trigger shall detect this and triggers learning experiments. During this learning phase, the system is actively excited to ensure accurate learning outcomes. After reaching sufficient accuracy, we stop the excitation, and the learning phase terminates. We update the model including the controller, and the parameter filter keeps monitoring the parameters. Unless there is more change in the future, there is no more need for further learning.

The proposed ETL approach yields the following three main improvements: 1) a statistical test that directly acts on the model quality; 2) point estimates that can be used as a new model; and 3) uncertainty quantification of the new model. In contrast to previous ETL, 2) allows for efficient online learning, and 3) gives us a notion about whether the learned model is sufficient or further excitation is required.

The proposed design is flexible and can be combined with any model-based downstream algorithm or control architecture. To demonstrate the flexibility of the proposed method, we consider a model predictive control task. Due to the optimization-based formulation of the controller architecture, we can readily incorporate learning objectives in the optimization, thus arriving at an active learning framework. We show the benefits of the proposed method for a DC motor use case where we simulate and successfully detect changes in the dynamics.

5.1.1 Contributions

In summary, this chapter makes the following contributions:

- proposing a parameter filter-based ETL approach that yields point estimates and uncertainty ellipsoids that can directly be combined into a powerful statistical test;
- leveraging the induced uncertainty to decide if we need more data for suitable model updates and designing optimal input signals based on the point estimate; and
- incorporating robustness margins and prior knowledge into the statistical test to target specific parameters.

5.2 Problem and Main Idea

Next, we introduce the considered system and control architecture. This is followed by a precise problem formulation.

5.2.1 System Dynamics

We consider a linear, discrete-time system

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{5.1}$$

with the state $x_k \in \mathbb{R}^n$, the control input $u_k \in \mathbb{R}^m$, the random disturbance $w_k \in \mathbb{R}^n$ at discrete-time $k \in \mathbb{N}$, and the unknown system matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ for some $n, m \in \mathbb{N}$. Furthermore, the disturbances $\{w_k\}$ are assumed to be independent and identically distributed (i.i.d.) and drawn from a normal distribution with mean 0 and known covariance Σ_w, which is denoted by $w_k \sim \mathcal{N}(0, \Sigma_w)$. Thus, the unknown parameters of the system can be fully parameterized by the stacked matrix $\Theta^\top = \begin{bmatrix} A & B \end{bmatrix} \in \mathbb{R}^{n\times(n+m)}$. Since the true system parameters Θ are unknown, we consider a model of the system, which is denoted by $\hat{\Theta}^* = \begin{bmatrix} \hat{A}^* & \hat{B}^* \end{bmatrix}$. Here, this model will be used to design a controller (cf. controller synthesis block in Figure 5.1). Further, assume that the states and inputs are subject to constraints $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$ for all $k \in \mathbb{N}$.

To ensure a well-behaved control algorithm, we require the following mild assumptions.

Assumption 28. *1) The pair (A, B) is stabilizable.*

2) The sets $\mathcal{X}$ and $\mathcal{U}$ are compact, convex and contain the origin.

The standard assumption of stabilizability ensures that the states of the system remain bounded under suitable control. The second assumption states that the equilibrium of the system is located in the admissible state space.

5.2.2 Objective: ETL with Recursive Learning

We assume that the parameters Θ for system (5.1) can change. Further, changes are assumed to be rare and sudden. The main problem we consider here is detecting these changes and subsequently updating the static model $\hat{\Theta}^*$ accordingly.

Clearly, we should leverage all of the available information to be as efficient as possible. Despite a change in the dynamics, the old model still contains valuable information that can be leveraged to relearn the model $\hat{\Theta}^*$ efficiently.

5.2.3 Main Idea: ETL Architecture

We propose to utilize a parameter filter (Sec. 5.3) that outputs a point estimate $\hat{\Theta}_{k|k}$ of the unknown system parameters Θ and additionally yields a posterior variance $P_{k|k}$. Usually, a Kalman filter is used to estimate unknown states from observations. Here, we estimate unknown system parameters from data. Due to the underlying normal distributions in the filter, it is possible to design the learning trigger (Sec. 5.4) based on well-established statistical tests for the null hypothesis

$$H_0 : \hat{\Theta}_{k|k} = \hat{\Theta}^*. \tag{5.2}$$

Whenever we reject the hypothesis, we trigger learning to adapt to the changes in the dynamics.

The filter output $\hat{\Theta}_{k|k}$ is usually subject to a high variance $P_{k|k}$ when there is only little movement in the system. Therefore, it is problematic to use it for directly updating the model $\hat{\Theta}^*$. Instead, we first shrink the ellipsoid through optimized control inputs (Sec. 5.5). This allows us to ensure that we relearn accurate models $\hat{\Theta}^*$. Due to the recursive nature of the filter, we also obtain an effective way to incorporate prior knowledge. The interactions between the components are depicted in Figure 5.1.

5.3 Recursive System Identification

Identifying linear systems is a classical problem that has been addressed in many textbooks, e.g., Ljung (2009) and Ljung and Söderström (1983), specifically for recursive system identification. Nonetheless, the identification of linear systems is still subject of recent research, e.g., Simchowitz et al. (2018, 2019). In our work, we discuss the question of *when to learn* by leveraging a state-space formulation of the parameter filter, which we have not found elsewhere.

We start by summarizing recursive system identification approaches that are primarily based on Goodwin and Payne (1977). These estimators provide point estimates and corresponding uncertainty ellipsoids. Further, depending on often implicitly made assumptions, the estimators have different properties. At the same time, these differences are critical when deciding what filter to use for an ETL approach. Therefore, we first present the most common approaches to recursive identification – i) least squares and ii) Kalman filter-type estimators. Afterward, we show why the latter is most beneficial for the problem at hand.

For least squares-type approaches, we usually assume time-invariant dynamics as introduced in (5.1). We show how to estimate the static model parameters with standard techniques and afterward consider addressing potential changes through forgetting. For example, we can reduce the influence of old data points through appropriate scaling.

Alternatively, the change can explicitly be taken into account and directly encoded into the estimation procedure through appropriate assumptions. Mathematically, we obtain the structure

$$x_{k+1} = A_k x_k + B_k u_k + w_k \tag{5.3}$$

and aim for time-varying estimates $\hat{A}_k$ and $\hat{B}_k$ of the nominal parameters A_k and B_k. This approach leads to a Kalman filter-style parameter filter.

Vectorized Process Model

To ease and unify notation, we introduce a vectorized system, i.e., the model parameters will be contained in a common vector. The general system (5.3) can be rewritten as

$$x_{k+1}^\top = d_k^\top \Theta_k + w_k^\top \,, \tag{5.4}$$

where $\Theta_k^\top = \begin{bmatrix} A_k & B_k \end{bmatrix} \in \mathbb{R}^{n\times(n+m)}$ is the stacked matrix containing all the system parameters and $d_k^\top = \begin{bmatrix} x_k^\top & u_k^\top \end{bmatrix} \in \mathbb{R}^{n+m}$ contains the state and input at time k.

Further, we can vectorize the system matrix using the vectorization operator $\mathrm{vec}(\cdot)$, which stacks the columns of a matrix to one vector. We write $z_k = \mathrm{vec}(\Theta_k) \in \mathbb{R}^{n(n+m)}$. The linear system (5.4) can now be written in vector form as

$$x_{k+1} = C_k z_k + w_k \,, \tag{5.5}$$

with $C_k = \mathrm{I}_n \otimes d_k^\top \in \mathbb{R}^{n\times n(n+m)}$, and $\otimes$ the Kronecker product.

5.3.1 Least Squares Estimators

The objective of least squares parameter estimation is to find estimated model parameters $\hat{z}_k$, which minimize the squared prediction error at the current time as

$$\hat{z}_k = \arg\min_{\hat{z}} \mathbb{E}\left[(x_{k+1} - C_k\hat{z})^\top (x_{k+1} - C_k\hat{z})\right]. \tag{5.6}$$

Clearly, the estimator highly depends on the available data C_k and, additionally, on the (often implicitly made) assumptions on the parameters z_k. Let

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}, \qquad C = \begin{bmatrix} C_0 \\ \vdots \\ C_{k-1} \end{bmatrix} \tag{5.7}$$

be the stacked data vectors and matrices generated by the system from time 0 to k. First, let us consider the standard time-invariant problem, i.e.,

$$z_{k+1} = z_k \quad \forall k. \tag{5.8}$$

Then, the analytic solution to (5.6) is given by

$$\hat{z}_k = \left(C^\top C\right)^{-1} C^\top X. \tag{5.9}$$

This batch least squares estimate can be reformulated recursively as

$$K_k = P_{k-1}C_{k-1}^\top \left(\mathrm{I} + C_{k-1}P_{k-1}C_{k-1}^\top\right)^{-1} \tag{5.10}$$

$$\hat{z}_k = \hat{z}_{k-1} + K_k(x_k - C_{k-1}\hat{z}_{k-1}) \tag{5.11}$$

$$P_k = (\mathrm{I} - K_kC_{k-1})P_{k-1} \tag{5.12}$$

with the starting values $\hat{z}_0$ and P_0 (see e.g., Goodwin and Payne (1977)).

In this estimator, the assumption of a constant model leads to a monotonically decreasing error covariance matrix P_k. With increasing amounts of data, the estimate converges, and the gain matrix K_k decreases with respect to a suitable norm. Thus, new measurements only have a small impact on the estimated parameters. The recursive least squares estimator eventually loses its adaptivity and stops updating. It weights all samples equally and does not give preference to more recent data. While this is desirable for constant model parameters, there will be issues if the system changes over time.

As stated earlier, one approach to making recursive least squares adaptable to changing parameters is the recursive least squares estimator with exponential forgetting Goodwin and Payne (1977). Here, observations that are l time steps before the current time k are weighted by λ^{k-l}, with the constant $\lambda \in (0, 1)$ (typically

$\lambda = 0.9, \ldots, 0.99$). Thus, increasing the influence of recent samples on the estimator. This is equivalent to adapting (5.5) and (5.8) with a weighted output equation

$$\lambda^{k-l} x_{l+1} = \lambda^{k-l} C_l z_l + w_l \qquad w_l \sim \mathcal{N}\left(0, \Sigma_w\right) . \tag{5.13}$$

The scaling with $\lambda^{k-l} < 1$ decreases the absolute value of the state. Hence, the squared prediction error is directly effected, and the influence of the corresponding point is lowered. This can be interpreted as artificially decreasing the signal-to-noise ratio for old data since the additional disturbance w_k is not weighted. Even though we keep z constant, the influence of past data is reduced because data with a lower signal-to-noise ratio is less informative. As a consequence, the underlying model (5.13) is equivalent to an unweighted observation with disturbances w_k whose covariance matrix is time-varying. We can simply divide both sides of (5.13) by λ^{k-l} and substitute $\left(\frac{1}{\lambda}\right)^{k-l} w_l$ by w_l. Then, we obtain for all times $l \leq k$

$$\begin{aligned} z_{l+1} &= z_l \\ x_{l+1} &= C_l z_l + w_l \qquad w_l \sim \mathcal{N}\left(0, \left(\frac{1}{\lambda^2}\right)^{k-l} \Sigma_w\right) \end{aligned} \tag{5.14}$$

as the data generation model for which recursive least squares with exponential forgetting yields the optimal estimate.

5.3.2 Kalman Filter Approach

In order to explicitly address potential change in the dynamics (cf. (5.5)), we assume the additional structure for the changes

$$\begin{aligned} z_{k+1} &= z_k + \Delta z_k \qquad && \Delta z_k \sim \mathcal{N}\left(0, \Sigma_z\right) \\ x_{k+1} &= C_k z_k + w_k \qquad && w_k \sim \mathcal{N}\left(0, \Sigma_w\right) , \end{aligned} \tag{5.15}$$

where the random variable Δz_k induces the change in the system parameters. By Σ_z we denote the covariance of the assumed model changes Δz_k, which can be used as a tuning parameter. Here, one may also incorporate prior knowledge of the possible system changes, e.g., which parameters can be affected in case of a load change, etc. We further consider the following standard assumptions.

Assumption 29. *1) The disturbance sequences $\{\Delta z_k\}$ and $\{w_k\}$ are i.i.d. and* $\mathbb{E}\left(\Delta z_k w_k^\top\right) = 0$.
2) The initial value z_0 is independent of Δz_k and w_k.

For such a system, the Kalman filter is the optimal state estimator Matisko and Havlena (2012). In particular, it is the optimal Bayesian estimator that keeps track of the full posterior distribution.

Typically, the Kalman filter is used to estimate the state vector $\hat{x}_k$ of a linear state-space model when only the observations y_k are available, and the state x_k is hidden. In our case, we assume access to the whole state of the system, but the systems transition matrices A_k and B_k, respectively z_k, are unknown and might change over time. Our goal is a filter that yields the system parameters.

Essentially, we lift the standard estimation problem one level higher to estimate the process model from given states instead of estimating the state from given measurements. Thus, for this vectorized system, the Kalman filter is given by

$$\hat{z}_{k+1|k} = \hat{z}_{k|k} \tag{5.16}$$

$$P_{k+1|k} = P_{k|k} + \Sigma_z \tag{5.17}$$

$$e_{k+1} = x_{k+1} - C_k \hat{z}_{k+1|k} \tag{5.18}$$

$$S_{k+1} = C_k P_{k+1|k} C_k^\top + \Sigma_w \tag{5.19}$$

$$K_{k+1} = P_{k+1|k} C_k^\top S_{k+1}^{-1} \tag{5.20}$$

$$\hat{z}_{k+1|k+1} = \hat{z}_{k+1|k} + K_{k+1} e_{k+1} \tag{5.21}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} C_k P_{k+1|k} \tag{5.22}$$

with initial values $P_{0|0}$ and $\hat{z}_{0|0}$. In this notation, a subscript $l|k$ indicates the estimate for time step l given the data up to time step k. If Σ_z is chosen *large*, the Kalman filter does weigh more recent samples higher than older ones. The resulting scheme is a recursive estimation method and allows for online learning.

Due to (5.15) and Assumptions 29, the Kalman filter estimate $\hat{z}_{k|k}$ is normally distributed with mean $\mathbb{E}(z_k)$ and error covariance $P_{k|k} = \mathbb{E}((\hat{z}_{k|k} - z_k)(\hat{z}_{k|k} - z_k)^\top)$ (cf. Spall (1984)).

5.4 Parameter Filter Learning Trigger

In this section, we design the learning trigger, which detects significant deviations between models and dynamical systems (cf. Figure 5.1). We derive the distribution of the parameter filter under the assumption that there is no change in dynamics ($\Delta z_k = 0$). Afterward, we propose a statistical test that validates if the data is consistent with the assumption $\Delta z_k = 0$. This is exactly the same hypothesis as (5.2) stated in Section 5.2.3. In the following, we use the notation $\hat{z}$ instead of $\hat{\Theta}$ since we consider the equivalent vectorized formulation for the tests.

5.4.1 Main Idea of the Learning Trigger

First, we explain the main idea of the learning trigger. Essentially, there are three critical parts: i) the current model believe $\hat{z}^*$, which is kept constant and utilized

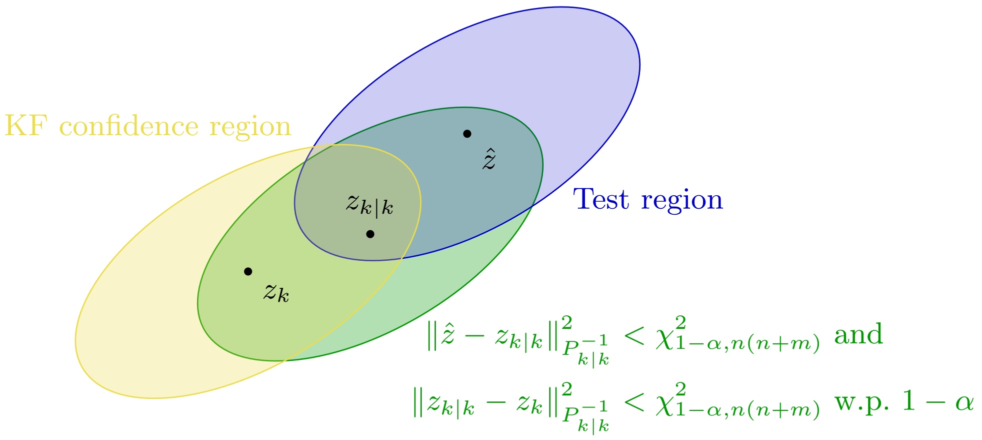

KF confidence region
$\hat{z}$
$z_{k|k}$
z_k
Test region
$\|\hat{z} - z_{k|k}\|^2_{P^{-1}_{k|k}} < \chi^2_{1-\alpha,n(n+m)}$ and
$\|z_{k|k} - z_k\|^2_{P^{-1}_{k|k}} < \chi^2_{1-\alpha,n(n+m)}$ w.p. $1-\alpha$

Figure 5.2: Visualization of confidence regions of the Kalman filter (KF) parameter estimation and model test regions. The estimated parameters $\hat{z}_{k|k}$, the true system parameters z_k, and the model parameters $\hat{z}^*$ are depicted as points in parameter space. The yellow ellipsoid is the confidence region of the Kalman filter estimation in which the error between z_k and $\hat{z}_{k|k}$ is contained with probability $1-\alpha$. If the parameter estimate is inside the test region around the current model, both, the model and the true system parameters, are located inside the green ellipsoid with probability $1-\alpha$.

for potential down-stream tasks, ii) the point estimate of the parameter filter $\hat{z}_{k|k}$, and iii) the covariance ellipsoid $P_{k|k}$. Of course, there is also the ground truth z_k; however, these parameters are unknown.

Therefore, we construct a statistical test around the objects i) - iii) to infer if the model believe $\hat{z}^*$ is significantly different from the ground truth z_k. In particular, we can guarantee that the ground truth z_k is with high probability contained inside an ellipsoid around the current estimate $\hat{z}_{k|k}$. Hence, when the old model belief $\hat{z}^*$ shows a large deviation from $\hat{z}_{k|k}$, we can also infer that the old model is not consistent anymore with the ground truth. Next, we make the main idea of the learning trigger mathematically precise and derive the corresponding distributions and confidence ellipsoids.

5.4.2 Learning Trigger

We start with well-known connections between normal and χ^2-distributions Corder and Foreman (2014). In particular, due to the property $\left(\hat{z}_{k|k} - z_k\right) \sim \mathcal{N}(0, P_{k|k})$ of the Kalman filter estimate, we know that $\left(\hat{z}_{k|k} - z_k\right) P^{-1}_{k|k} \left(\hat{z}_{k|k} - z_k\right)^\top < \chi^2_{1-\alpha,n(n+m)}$ with probability $1-\alpha$ for some $\alpha \in (0,1)$. By $\chi^2_{1-\alpha,n(n+m)}$ we denote the $1-\alpha$ quantile of the chi-square distribution with $n(n+m)$ degrees of freedom. This corresponds to the confidence region in Figure 5.2 depicted in yellow. The estimated parameters

$\hat{z}_{k|k}$ are located inside this ellipsoid around the true system z_k with probability $1-\alpha$. As a short notation for an expression $zP^{-1}z^\top$ we use $\|z\|^2_{P^{-1}}$ in the figure since P as a covariance is positive definite. If the null-hypothesis (5.2) is true, we obtain $\left(\hat{z}_{k|k} - \hat{z}^*\right) P^{-1}_{k|k} \left(\hat{z}_{k|k} - \hat{z}^*\right)^\top < \chi^2_{1-\alpha,n(n+m)}$ with probability $1-\alpha$.

Based on these properties, we propose the learning trigger

$$\gamma_{\text{learn}} = 1 \iff \left(\hat{z}_{k|k} - \hat{z}^*\right) P^{-1}_{k|k} \left(\hat{z}_{k|k} - \hat{z}^*\right)^\top > \chi^2_{1-\alpha,n(n+m)} \tag{5.23}$$

as a statistical test of level α. Thus, a trigger event occurs when the data-based estimate $\hat{z}_{k|k}$ and the fixed model $\hat{z}^*$ are most likely to differ more than what is expected by the uncertainty in the data. Further, by using this learning trigger, the probability of false trigger events, i.e., triggering despite having an accurate model, is bounded as stated in the following theorem.

Theorem 30. *Consider the dynamical system* (5.15) *and let the Assumptions 29 hold. If the learning trigger* (5.23) *is used, then in case of a perfect model, i.e.,* $\hat{z}^* = z_k$*,*

$$\mathbb{P}\left[\gamma_{\text{learn}} = 1\right] \leq \alpha\,. \tag{5.24}$$

Proof. The theorem follows directly from the properties of the Kalman parameter filter. By design, we ensure that the estimates are subject to a normal distribution. Further, through direct access to the variance, we can normalize the distribution and apply a standard χ^2 test. The test then directly induces the confidence region. □

Thus, the risk of false trigger events and at the same time the sensitivity of the trigger can be adjusted by the probability α. If (5.23) is satisfied, the null hypothesis is rejected, and a new model needs to be identified.

The test statistic is also known as squared Mahalanobis distance, which is a generalization of the Euclidean distance to multivariate spaces with different scaling De Maesschalck et al. (2000). If the covariance matrix is non-singular, the Mahalanobis distance fulfills all properties of a metric. It scales the space according to the covariance such that points lying on equal contour lines of the normal distribution have the same distance from the center point. It can also be viewed as a distance of a point to a normal distribution. This is also how our test can be interpreted. The Kalman filter gives a conditional distribution of parameter values with mean $\hat{z}_{k|k}$ and error covariance $P_{k|k}$. Thus, if the assumptions are fulfilled, $\hat{z}_{k|k}$ to z_k has squared Mahalanobis distance $\|\hat{z}_{k|k} - z_k\|^2_{P^{-1}_{k|k}} < \chi^2_{1-\alpha,n(n+m)}$ with probability $1-\alpha$.

If the test yields $\|\hat{z}_{k|k} - \hat{z}^*\|^2_{P^{-1}_{k|k}} < \chi^2_{1-\alpha,n(n+m)}$, we can conclude from the triangle inequality that at least with probability $1-\alpha$ the model and the true system parameters fulfill $\|\hat{z}^* - z_k\|^2_{P^{-1}_{k|k}} < \sqrt{2}\chi^2_{1-\alpha,n(n+m)}$. In Figure 5.2, this can be interpreted as we test if $\hat{z}_{k|k}$ is inside the blue ellipsoid *test region* around the model $\hat{z}^*$. From the Kalman filter, we get the yellow *confidence region*, which contains $\hat{z}_{k|k}$ with probability $1-\alpha$. The boundary of the ellipsoids has squared Mahalanobis distance $\chi^2_{1-\alpha,n(n+m)}$ from the corresponding center points. Since $\hat{z}_{k|k}$ is contained in both regions, the confidence region and the test region must intersect. Hence, the squared distance between z_k and $\hat{z}^*$ fulfills $\|\hat{z}^* - z_k\|^2_{P^{-1}_{k|k}} < \sqrt{2}\chi^2_{1-\alpha,n(n+m)}$ with probability $1-\alpha$.

Remark 31. *If the assumptions on the noise terms are not fulfilled, the normal distribution of the Kalman filter estimate does not follow immediately; however, there are properties that can ensure asymptotic convergence to a normal distribution Spall and Wall (1984). Noteworthy, the parameter estimate is still asymptotically normal distributed without the assumption of Gaussian disturbances if it holds that the disturbances have zero mean, i.e.,* $\mathbb{E}(\Delta z_k) = 0$, $\mathbb{E}(w_k) = 0$, *and the disturbance covariance matrices and the initial error covariance are bounded.*

Remark 32. *In Theorem 30, we assume that the additive process disturbance* w_k *is independent of the state and has constant covariance* Σ_w*. In practice, however, these assumptions could be violated if the controlled system is slightly nonlinear, for example. One approach to increase the robustness of the algorithm against such deviations is to assume worse disturbances in the model than those that are actually observed. Due to the over-approximation of the process noise, the test region of the Kalman filter trigger increases. This leads to a much wider margin between the test statistics and the threshold value compared to tests with ideal noise assumptions. This results in a lower sensitivity to small system changes but also in higher robustness to violated model assumptions.*

5.5 Experiment Design

The error covariance $P_{k|k}$ is an essential object of the proposed learning trigger and should be small for an effective test. At the same time, it is a measure of the accuracy of the corresponding parameter estimate. Thus, after a trigger event occurs, a learning experiment should be carried out to reduce the uncertainty of the estimate. In general, the accuracy of a model estimate highly depends on the data and, in particular, the excitation of the system. For example, consider the least squares estimator (5.9) from Section 5.3. Here, the matrix $C^\top C$ can be written as

$C^\top C = \mathrm{I}_n \otimes \left(\sum_{t=1}^{k-1} d_t d_t^\top\right)$. The collected data must contain at least $(n+m)$ linearly independent vectors d_t such that the inverse of the matrix $C^\top C$ exists. The rank condition on $\sum_{t=1}^{k-1} d_t d_t^\top$ is closely related to persistency of excitation of the data sequence $\{d_t\}$.

5.5.1 Persistency of Excitation and Observability

Persistency of excitation has been introduced in several slightly different notations (cf. Johnstone et al. (1982); Bai and Sastry (1985); Caines and Lafortune (1984); Green and Moore (1986)). In a nutshell, if data has a sufficiently rich information content and gives insight into the system dynamics, then it is possible to guarantee convergence of statistical estimators.

For finite sequences, persistency of excitation over an interval is defined in Green and Moore (1986).

Definition 33 (Persistency of excitation)**.** *A sequence of data* $\{d_t\}_{t=0}^{k-1}$*,* $d_t \in \mathbb{R}^{n+m}$ *is persistently exciting over the time interval* $\{0, \ldots, k-1\}$*, if there exists* $\epsilon > 0$ *such that*

$$\sum_{t=0}^{k-1} d_t d_t^\top \succeq \epsilon \mathrm{I}_{n+m}. \tag{5.25}$$

Here, we make use of the Loewner order for positive semi-definite matrices. For two positive semi-definite matrices M_1 and M_2, we say that $M_1 \succeq M_2$ if $M_1 - M_2$ is positive semi-definite. Thus, if $\{d_t\}_{t=0}^{k-1}$ is persistently exciting, the matrix $C^\top C$ is positive definite and has an inverse.

For $\{d_t\}_{t=0}^{k-1}$ to be persistently exciting, both components $\{x_t\}_{t=0}^{k-1}$ and $\{u_t\}_{t=0}^{k-1}$ must be persistently exciting. Since the input sequence $\{u_t\}_{t=0}^{k-1}$ is a design variable, one purpose of experiment design is to ensure persistency of excitation of $\{x_t\}_{t=0}^{k-1}$. This is contrary to the control objectives during normal operation, where deviations from the setpoint are suppressed.

Persistency of excitation of $\{d_t\}_{t=0}^{k-1}$ induces observability of the lifted parameter system (5.15). A linear discrete-time parameter varying system (5.15) is said to be completely observable if the observability matrix

$$O_0 = \begin{bmatrix} C_0 \; C_1 \; \ldots \; C_{n(n+m)} \end{bmatrix}^\top \tag{5.26}$$

has $\operatorname{rank}(O_0) = n(n+m)$ Witczak et al. (2017). This is the same condition as for persistency of excitation. Thus, without disturbances, the parameter vector z_k could be determined from persistently exciting data.

While persistency of excitation ensures well-defined solutions to the estimation problem, it is not sufficient to control the estimation error. Indeed, this means that the estimate may diverge under permanent updates. Next, we consider an active learning problem to reduce the posterior variance of the estimator through optimized excitation of the system.

5.5.2 Active Learning

The error covariance P of the estimator is a natural measure of the accuracy of the estimate. Here, we aim at minimizing the trace of P, which corresponds to minimizing the sum of the eigenvalues of the error covariance. If trace(P) is minimized by the data of the experiment, the data acquisition is called A-optimal Pronzato (2008). By doing such an A-optimal experiment, the sum of squared lengths of principal semi-axes of the confidence ellipsoid is minimized. For the Kalman filter, the mean and covariance are usually obtained recursively. However, there exist also closed forms as derived in Han (2010), where the influence of the experiment design on the covariance can be investigated.

The excitation during the experiment could be provided by external reference signals such as pseudo-random binary signals or chirp signals. However, then also input and state constraints need to be considered. A different approach was presented in Heirung et al. (2012). There, a dual MPC is proposed, which considers the expected trace of the error covariance matrix of a recursive least squares estimator with exponential forgetting in the optimization problem. Here, this approach is adopted and modified such that the MPC propagates a model of the Kalman filter instead of the recursive least squares estimator. Its error covariance is jointly minimized in the optimization problem. Thus, then we obtain the objective function:

$$\min_{X_{t_j},U_{t_j}} \sum_{k=0}^{N-1} x_{k|t_j}^\top Q x_{k|t_j} + u_{k|t_j}^\top R u_{k|t_j} + x_{N|t_j}^\top Q_N x_{N|t_j} + \nu \operatorname{trace}(\widetilde{P}_{k+1|k+1}) \tag{5.27}$$

$$\begin{aligned} \text{s.t.} \quad x_{0|t_j} &= x_{t_j}, && \\ x_{k+1|t_j} &= \hat{A} x_{k|t_j} + \hat{B} u_{k|t_j}, && k \in \{0,\dots,N-1\} \\ x_{k|t_j} &\in \mathcal{X}, && k \in \{0,\dots,N-1\} \\ u_{k|t_j} &\in \mathcal{U}, && k \in \{0,\dots,N-1\} \\ x_{N|t_j} &\in \mathcal{X}_N, && \\ \widetilde{P}_{0|0} &= P_{t_j|t_j}, && \\ \widetilde{C}_k &= \mathrm{I}_n \otimes \begin{bmatrix} x_{k|t_j}^\top & u_{k|t_j}^\top \end{bmatrix}, && k \in \{0,\dots,N-1\} \\ \widetilde{S}_{k+1} &= \widetilde{C}_k \widetilde{P}_{k+1|k} \widetilde{C}_k^\top + \Sigma_w, && k \in \{0,\dots,N-1\} \\ \widetilde{K}_{k+1} &= \widetilde{P}_{k+1|k} \widetilde{C}_k^\top \widetilde{S}_{k+1}^{-1}, && k \in \{0,\dots,N-1\} \\ \widetilde{P}_{k+1|k+1} &= \widetilde{P}_{k+1|k} - \widetilde{K}_{k+1} \widetilde{C}_k \widetilde{P}_{k+1|k}, && k \in \{0,\dots,N-1\}. \end{aligned}$$

Here, ν is a weighting parameter. Our proposed method is summarized and abstracted in Algorithm 1.

Unfortunately, $\widetilde{P}_{k|k}$ is a nonlinear function of the system's states and inputs. Therefore, the optimization problem becomes nonlinear and nonconvex. To make the computation feasible, the standard MPC problem without the cost of the covariance matrix can be computed and used as an initial guess for the harder optimization problem. Then, even if no global optimum is found, a feasible solution can be provided. In Heirung et al. (2012), no stability guarantees are given about the considered approach. In the case of re-identification after a wrong model is detected, this is a hard problem since the model used for control itself is not trustworthy. In Anderson et al. (2018), a robust re-identification procedure for MPC is given. However, there the considered system changes are bounded such that the controller is stabilizing for all possible system changes.

5.6 Simulation Example

Next, we show the proposed ETL framework in a numerical simulation. In particular, we demonstrate the following properties:

- The ETL framework detects changes in the dynamics;
- we can adapt to change and make the uncertainty ellipsoid of the estimator arbitrarily small during dedicated learning experiments; and
- due to the MPC-based nature of the proposed learning experiments, we can simultaneously shrink the uncertainty and satisfy state and control constraints.

To contrast our approach to adaptive approaches, we compare our method to continuously updating model parameters. We show that continuously updating methods return after the learning experiment back to a steady-state, where the estimator performs a random walk in an ellipsoid that can grow arbitrarily large. Our method does not suffer from this issue.

5.6.1 Setup

The considered system describes the dynamics of a servomechanism consisting of a DC-motor, a gear-box, an elastic shaft, and an uncertain load, which is adopted from Bemporad and Mosca (1998) and Schwenkel et al. (2020). We chose this example as it allows us to illustrate the efficacy of our approach to changes in the dynamics, which will be represented by changes in the load. While there are also other control approaches to deal with load changes, we want to demonstrate that our method is flexible and can readily cope with a problem without any further adaptation.

The continuous-time state-space equations are given by

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_\theta}{J_\mathrm{L}} & -\frac{\beta_\mathrm{L}}{J_\mathrm{L}} & \frac{k_\theta}{\rho J_\mathrm{L}} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_\theta}{\rho J_\mathrm{M}} & 0 & -\frac{k_\theta}{\rho^2 J_\mathrm{M}} & -\frac{\beta_\mathrm{M} R + K_\mathrm{T}^2}{J_\mathrm{M} R} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K_\mathrm{T}}{R J_\mathrm{M}} \end{bmatrix} u\,, \tag{5.28}$$

where the state vector $x \in \mathbb{R}^4$ consists of the load angle, the motor angle and their time-derivatives. The input u corresponds to the input DC voltage in Volt, which is constrained by $|u| \leq 220$. In addition, the state constraint $\left|\begin{bmatrix} k_\theta & 0 & \frac{-k_\theta}{\rho} & 0 \end{bmatrix} x\right| \leq 78.5398$ must be fulfilled. In the example, all model parameters are known (and can be found in Bemporad and Mosca (1998)) except for the load inertia J_L, which has the uncertainty range $10\,J_\mathrm{M} \leq J_\mathrm{L} \leq 30\,J_\mathrm{M}$. The continuous-time model is converted into a discrete-time model with the sampling time $T_\mathrm{s} = 0.1\,\mathrm{s}$ using zero-order hold on the input. For the sake of notational convenience, we omit the units for the discretized system. On the discrete-time system, additive disturbances w_k are introduced, where w_k is drawn from a normal distribution with zero mean and covariance matrix $\Sigma_w = \operatorname{diag}\left(\begin{bmatrix} 0.99 & 0.99 & 0.939 & 0.056 \end{bmatrix} \cdot 10^{-4}\right)$.

Simulations of 3000 time steps are performed. During the first 1000 steps, the nominal system with $J_\mathrm{L} = 20\,J_\mathrm{M}$ is used. At time steps 1000 and 2000, the simulated system changes its parameter to $J_\mathrm{L} = 22\,J_\mathrm{M}$ and $J_\mathrm{L} = 19\,J_\mathrm{M}$, respectively. A standard MPC with zero terminal constraint is introduced based on the nominal model to control the system. The prediction horizon is set to 6 time steps. We use the Kalman filter, as described in Section 5.3.2, for the parameter filter. The assumed covariance of system changes Σ_z is designed using prior knowledge about possible parameter variations. Thus, parameters that are unlikely to change are estimated very precisely, while the estimation of possibly changing parameters is adapting more rapidly.

To compare our ETL approach to nominal control without model adaption and to permanently adapted models, three similar simulations are performed.

5.6.2 Event-triggered Learning

First, the ETL approach is applied to the system. The model and the controller is kept constant if no change is detected. For change detection, the test statistic from Theorem 30 is computed. The graph of the test statistic over time is depicted in Figure 5.3. It is visible that the test statistic rises when the system parameters are changed after time step 1000 and 2000. Eventually, the threshold is exceeded, which means that a significant model error is detected. Then, a learning experiment using

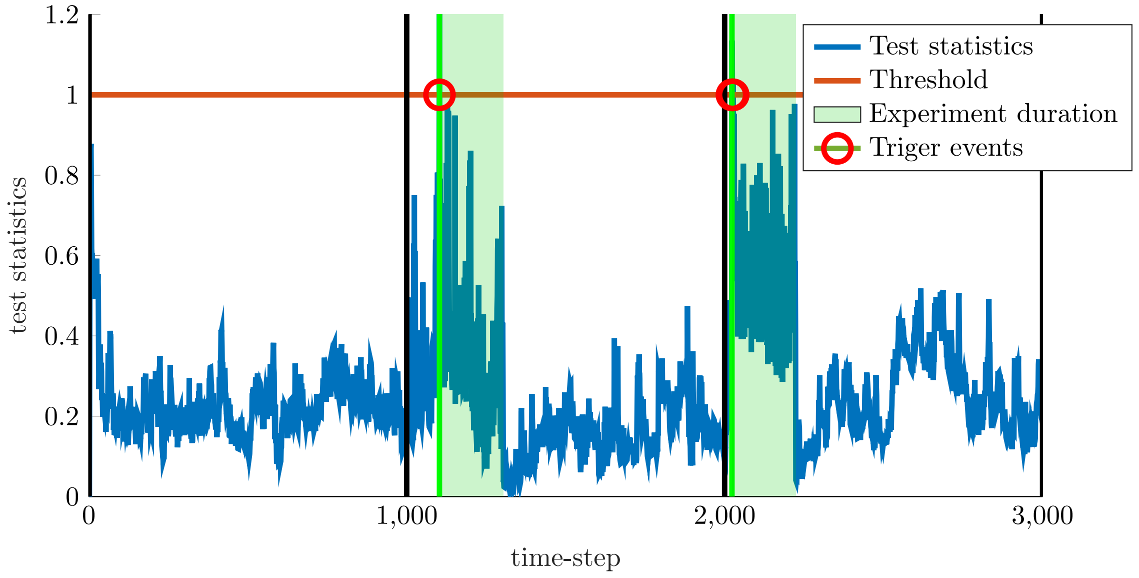


Figure 5.3: Test statistics of the learning trigger. The test condition (5.23) is evaluated in every time-step. In the figure, the test statistics and the threshold are normalized, such that a test statistic larger than one leads to a learning trigger event. This happens shortly after the true system parameters change at time-steps 1000 and 2000, respectively. Thus, changing system parameters are detected reliably without triggering when the parameters stayed constant.

the experiment MPC (5.27) is performed. In this period of 200 time steps, a higher excitation of the system is generated. This fact can also be noticed in Figure 5.7, where the trajectory of angles is shown divided into nominal control and experiment operation. After the experiment, the current parameter estimate is set as the new model and used to update the controller and the test. Model updates only happen directly after the learning experiments. For the remaining time, the model is kept constant.

5.6.3 Permanent Model Updates

In the second simulation, in every time step, the current parameter estimate is used to update the model and the controller. The estimation algorithm is capable of tracking the true system parameters. However, without significant excitation, the uncertainty stays large, and the estimated parameters perform a random walk inside the uncertainty ellipsoid. This behavior is depicted in Figures 5.4 and 5.5, where the estimation error of four exemplary parameters is shown for the ETL and the permanent update approach. In Figure 5.5, the current estimate is always used as the model (permanent updates), which is also used for control. Thus, the model is always varying, which can be seen based on the fluctuating error (red line). In

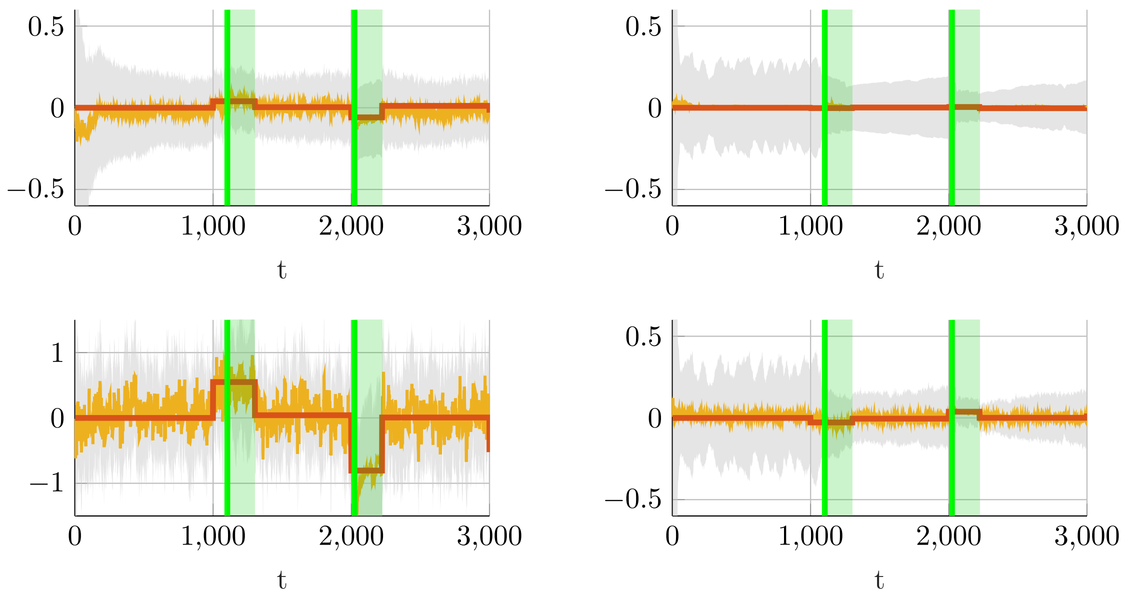


Figure 5.4: Evolution of four exemplary parameters and their estimate using ETL. In red, we show the error between system and model parameters. The yellow line displays the estimated model error using the current parameter point estimate. The gray region depicts a projection of the confidence ellipsoid onto these parameters centered around the estimated error. The model error stays close to zero, while the estimation error varies. When the gray region does not include zero, a learning experiment is triggered that excites the system to force the yellow line toward the red one. Thus, the uncertainty is reduced, and after the experiment, the model is updated with a precise estimate.

contrast, as depicted in Figure 5.4, in the ETL approach, the model is kept constant unless a trigger occurs. Before updating to a new model, the system is excited in the experiment to obtain higher precision. If the estimation error after the update grows, this does not affect the used model and the control algorithm.

5.6.4 No model updates

Third, the same kind of simulation is performed without adapting the model and the controller. This corresponds to purely relying on the robustness of the control system against small parameter deviations. In Figure 5.6, the actual model error for four selected parameters is shown. As no update is carried out, the error will remain large when the system changes its parameters. This can be disadvantageous as this error deteriorates the control performance. In Figure 5.8, it is visible that the state constraints are slightly violated in the experiments using permanent updates and constant nominal models that are never adopted, respectively.

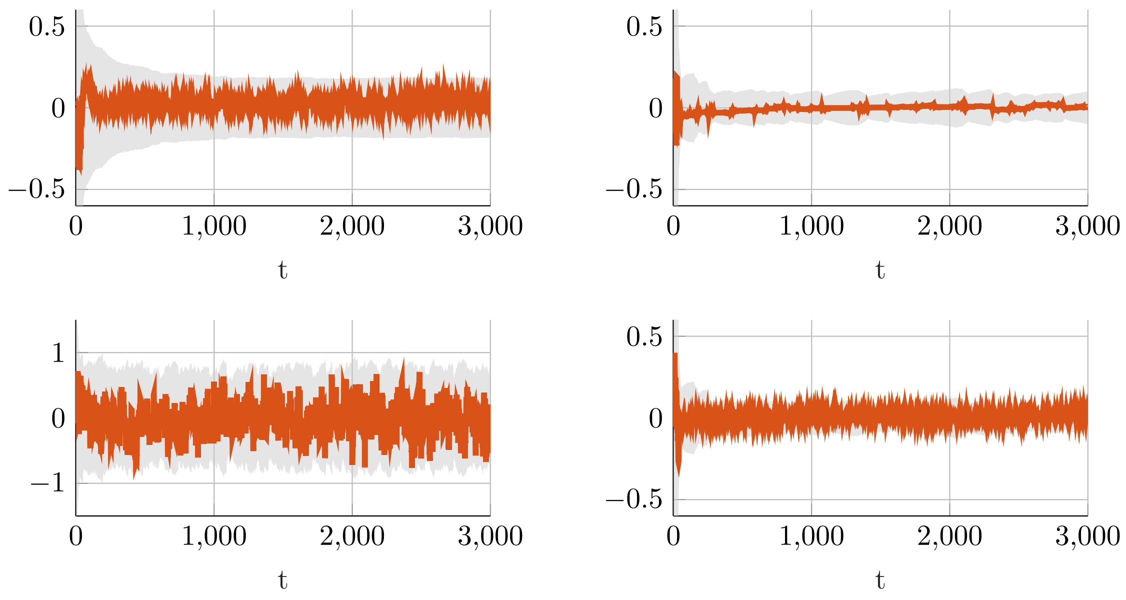


Figure 5.5: Evolution of the same four exemplary parameters and their corresponding estimates using permanent model updates. In contrast to Figure 5.4, the model error oscillates heavily. It is not even possible to detect the introduced changes in the system dynamics (at $t = 1000$ and $t = 2000$) within this random walk.

5.6.5 Results

Updating the model *all the time* or *never* are both problematic. Without any updates (Section 5.6.4), the nominal model will never adapt to changes. Thus, the model error will indefinitely cause problems. On the other hand, updating the model in every time step (Section 5.6.3) with uninformative data also results in poor models even when the true system remains unchanged. Both of these extremes are problematic and lead to issues in terms of violating constraints (cf. Figure 5.8) and performance.

Event-triggered learning addresses these issues and keeps the model constant as long as no significant deviations are detected between the model and the current estimate. Thus, a change of the model is only performed if the data indicates a significant model error. Keeping the model fixed during nominal operation prevents the illustrated divergence issues of permanent updates.

The effect of the learning experiment on the estimation error is shown by the results in Table 5.1, where the average squared model parameter error is given for the different simulations. If we neglect the periods of the simulation in which learning experiments took place ("Excluding excitation" in Table 5.1), the model error is by far the smallest for ETL. This once again highlights that the excitation during the learning experiment yields an accurate model.

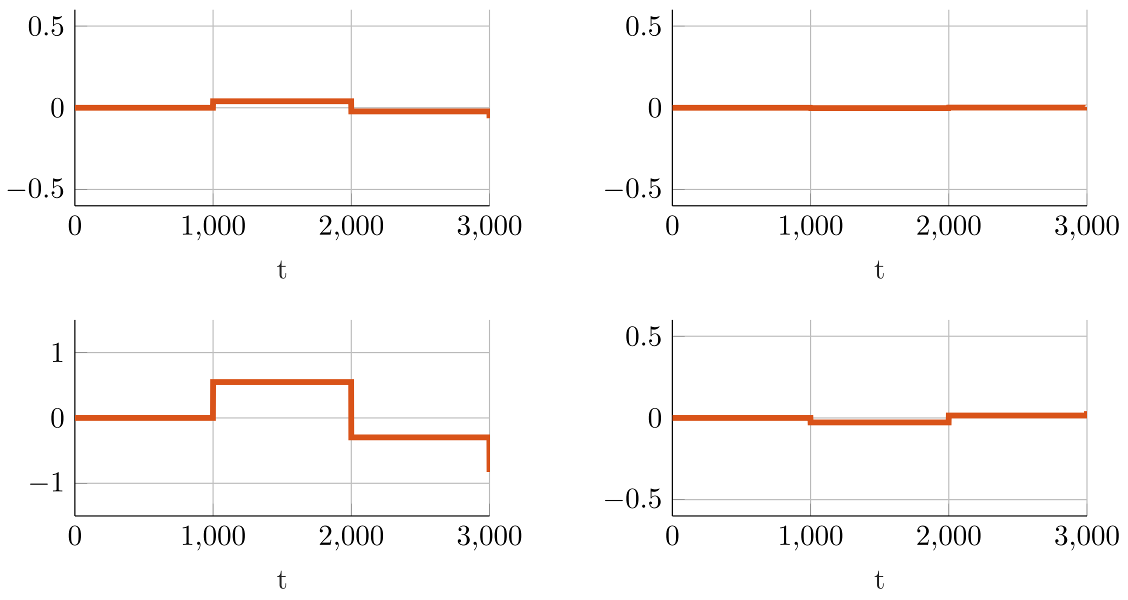


Figure 5.6: Evolution of four exemplary parameters and their estimate using the nominal model for control. Without model updates, the model contains errors when the system changes its parameters.

If the whole duration of the simulation is taken into account, the permanently updated model has a smaller average error than the simulation with ETL and learning experiments. This is a result of the immediate response of the estimate – and hence also of the model and controller – after a system change. In ETL, we still considered the old model belief, which we have detected as inaccurate, during the learning experiment. However, obtaining a new, accurate model takes some time and requires excitation, and thus, learning has its own price. Clearly, the longer the model remains unchanged after learning, the higher the benefit of an accurate model.

5.7 Conclusions

We propose a parameter filter-based learning trigger that generalizes previous work on event-triggered learning. By considering the inherent notion of uncertainty provided by Kalman-type filters, we can combine point estimates with powerful statistical tests to trigger learning experiments on necessity. Further, we are able to provide optimal excitation signals that specifically target parameters with high uncertainty. We provide an ETL framework that joins i) learning trigger, ii) online model learning, and iii) experiment design. The core assumption that makes this possible is the linearity of the dynamics, which is rooted deep inside the parameter filter. Extending this to nonlinear systems requires significant extensions of all aspects i)–iii).

Algorithm 1: Event-triggered learning algorithm

```
Data: Initial model
Initial covariance parameters
Set up parameter filter
Set up nominal controller
Set mode = "control"
while true do
    Measure state
    Update parameter filter estimate and uncertainty
    switch mode do
        case "control" do
            Evaluate trigger condition
            if trigger condition = true then
                Set mode = "experiment"
            end
            Apply nominal input
        end
        case "experiment" do
            Check condition to stop experiment
            if stopping condition = true then
                Set mode = "control"
                Update model and controller
            end
            Apply experiment input
        end
    end
end
```

Table 5.1: Average squared model parameter error $[1 \times 10^{-3}]$.

	ETL	Permanent updates	No updates
Whole simulation	4.359	3.197	7.123
Excluding excitation	1.065	3.322	6.581

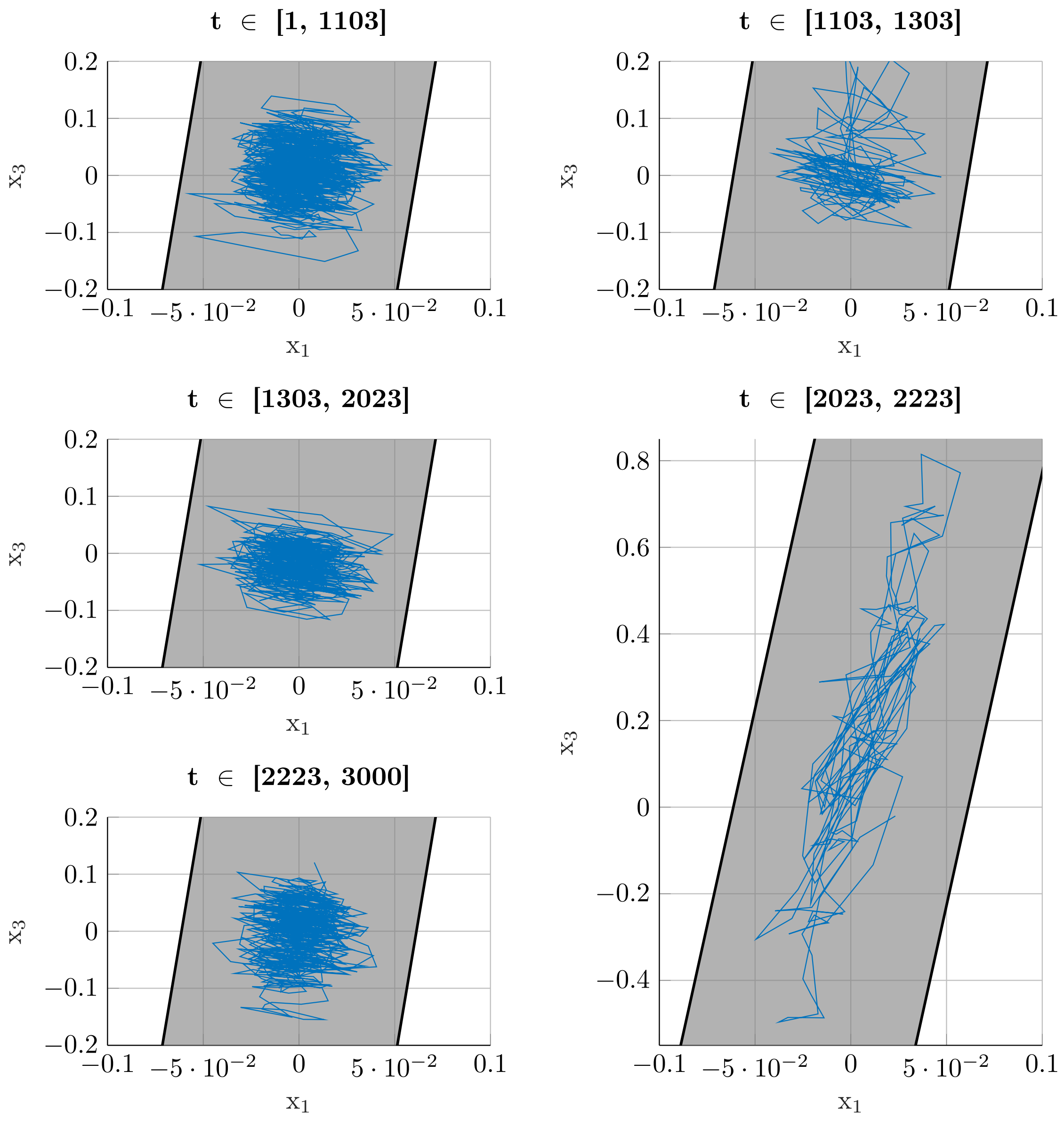
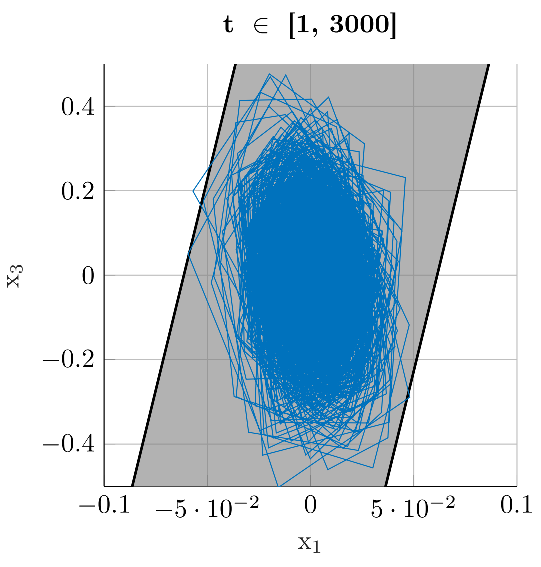
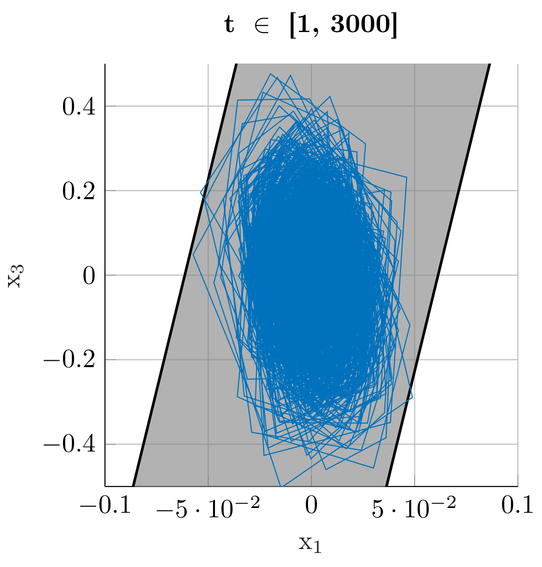


Figure 5.7: Trajectory of the angles x_1 and x_3 partitioned by simulation section. During the experiments (right column), the excitation is clearly larger. The gray area depicts the state constraints.



(a) Permanent model updates



(b) Nominal model

Figure 5.8: Trajectory of the angles x_1 and x_3 using (5.8a) permanent model updates for control and (5.8b) the nominal model for control. The color-coding is the same as in Figure 5.7. In contrast to the ETL approach in Figure 5.7, we can observe state constraint violations here.

Part III

Comparing Dynamical Systems

Chapter 6

Kernel Two-sample Test for Dynamical systems

This chapter is based on the journal paper (Solowjow et al., 2020).

After introducing the general idea of ETL and presenting concrete implementations, we now take a step back. We consider the general underlying core questions of comparing dynamical systems directly from data. We compare the stationary distributions of the states by introducing a new type of mixing. Therefore, we use the notation X_k here to emphasize the interpretation of the states as random variables that are drawn from said stationary distribution.

6.1 Introduction

We consider the two-sample problem of determining whether two distributions are different. In particular, we generalize the well-established kernel two-sample test (Gretton et al., 2012a) to dynamical systems and stochastic processes with certain mixing properties, which we make precise in this chapter.

The kernel two-sample test approximates a metric on the space of probability distributions, the maximum mean discrepancy (MMD), through kernel-based techniques. Due to its powerful theoretical properties and versatile applicability, kernel two-sample testing is a prominent method in the machine learning community (Long et al., 2017; Tolstikhin et al., 2018; Muandet et al., 2017; Schölkopf and Smola, 2001). While we can exploit parts of existing kernel-based results, and especially their theoretical guarantees, the extension to comparing dynamical systems is not straightforward. This is mainly because kernel two-sample testing was initially developed for independent and identically distributed (i.i.d.) random variables (Gretton et al., 2012a). The i.i.d. assumption in the test is critical, but it is violated by the very nature of dynamical systems: through the dynamics, samples are coupled to past samples. To address this issue, we introduce a novel notion of mixing that considers the dependence of data through time with respect to the MMD. Intuitively, mixing reveals how fast autocorrelations decay and, thus, how long we need to wait in-between samples for data to be (approximately) independent. Our new mixing notion can be efficiently estimated from data and is particularly synergistic with kernel two-sample tests since both measure distances of probability distributions with respect to the same metric – the MMD. By estimating the decay of dependency and embedding it into well-established algorithms, we obtain a powerful test for comparing dynamical systems.

Mathematical literature often distinguishes explicitly between deterministic dynamical systems and stochastic processes. In particular, establishing mixing properties for deterministic dynamical systems is an extremely challenging problem and constructing examples that are provably mixing is hard. Further, common mixing properties that are used for stochastic systems are too restrictive and not applicable to deterministic systems (Hang et al., 2017). We propose mixing in MMD, which is applicable to both classes of problems—stochastic and deterministic systems. Further, we show that mixing in MMD is even less restrictive than certain deterministic mixing types ($\mathcal{C}$-mixing Hang et al. (2018)). For suitable choices of kernels and function spaces $\mathcal{C}$, we can show that $\mathcal{C}$-mixing implies MMD-mixing. Based on standard examples with well-established $\mathcal{C}$-mixing properties (the β-map, logistic map, and Gauss map), we demonstrate empirically that they are indeed mixing in MMD. Additionally, we also consider mixing properties of chaotic and stochastic systems and further, also raw sensor data from human walking experiments.

Despite their practical relevance, there is no established data-driven way of comparing dynamical systems. For biomedical systems such as the human cardiovascular system, central nervous system, or musculoskeletal system, implementing a principled comparison of systems based on their output sequences in different time intervals can help to detect diseases or quantify their severity. For example, alterations or unusual patterns in human gait can be indicators for early stages of Parkinson's disease (Pistacchi et al., 2017). An algorithm that automatically detects such alterations by comparing new data to labeled records could help physicians in their decision-making. Current state-of-the-art solutions rely on manually engineered and selected features and thus require expert knowledge (Nguyen et al., 2019). Similarly, feature-based solutions have been proposed for electro- myography-based detection of spasticity (Misgeld et al., 2015; Lueken et al., 2015). But clearly, the success of such approaches critically depends on the expressiveness of these features and on how well the problem is understood.

Modern engineering applications are another prominent and relevant example. They often leverage computer simulations instead of directly interacting with the physical plant since real experiments are more expensive, time-consuming, and cause wear on the hardware. Besides, being able to predict the response of a physical plant based on a mathematical model enables powerful learning algorithms (Hwangbo et al., 2019), model-predictive control (Qin and Badgwell, 2003), and digital twins in future manufacturing (Jeschke et al., 2017). The success of these methods, however, is critically intertwined with the model accuracy. Thus, it is essential to ensure accurate models, for example, by comparing data generated from the simulation model with data collected from the real system.

By combining mixing properties with kernel-based techniques, we obtain a powerful statistical test for comparing dynamical systems. We demonstrate the efficiency and robustness of the proposed test numerically and on experimental data. In particular, we consider human walking experiments and analyze raw data from an inertial measurement unit (IMU) to detect anomalies in the walking pattern. Without the need for human expert knowledge or fitting model parameters, our test outperforms standard baselines in deciding which of the trajectories were generated with an attached knee orthosis, which restricts the movement of the joint.

6.2 Assumptions and Problem Formulation

In the following, we introduce the mathematical objects that we will consider in this chapter. Afterward, we make the problem precise.

6.2.1 Stationary, Ergodic, and Mixing Systems

Let $(\Omega, \mathcal{A}, P)$ be a probability space, $S \subset \mathbb{R}^d$ a compact set, which is the state space of the dynamical system, and $\mathcal{B}$ the corresponding Borel σ-algebra. We define a stochastic dynamical system or stochastic process as a collection of random variables $\{X_k\}$ indexed in discrete time $k \in \mathbb{N}$ and $X_k \colon \Omega \to S$. Next, we introduce some required properties of the process.

Definition 34 (Stationary)**.** *A system is stationary if the joint distribution of its states is time-invariant.*

In addition to stationary behavior, we also require ergodicity. While stationarity ensures time-invariant distributions, ergodicity guarantees that the statistical properties of the system do not differ over multiple realizations. We use a standard definition that goes back to Birkhoff (1931).

Definition 35 (Ergodic)**.** *Assume the system $\{X_k\}$ is stationary with distribution $\mathbb{P}$. We call the system ergodic if for all $f \in L^1_{\mathbb{P}}(S)$ and $\mathbb{P}$-almost all initial states we have*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} f(X_k) = \int_S f(y) \mathrm{d}\mathbb{P}(y) \quad a.s.. \tag{6.1}$$

Equation (6.1) is in some sense a realization of the law of large numbers, and both sides of the equation yield the expected value $\mathbb{E}_{X\sim\mathbb{P}}[f(X)]$. In particular, it allows us to estimate $\mathbb{E}[X_k]$ (the distribution is invariant for all k) from long enough sample paths. Different types of convergence and test functions in Eq. (6.1) yield more sophisticated ergodic theorems. Nonetheless, there can still be severe autocorrelations and if X_k is known, this may have a drastic impact on the distribution of X_{k+1}. Thus, we require additional mixing assumptions.

Classically, mixing is introduced in terms of dependencies between σ-algebras and intuitively, deals with the autocorrelations in the system. Here, we consider a covariance-based approach to mixing, which is more useful and convenient for us since there is a natural connection to Hilbert-Schmidt theory in RKHSs. Both approaches are introduced in Bradley et al. (1987). We begin with a general definition based on (Bradley et al., 1987, Eq. (1.2)) and tailor it to our problem afterward.

Definition 36 (Measure of Dependence)**.** *Assume $\mathcal{F}$ and $\mathcal{G}$ are suitable function spaces. The measure of dependence is defined as*

$$\sup_{f\in\mathcal{F},\, g\in\mathcal{G}} \frac{|\mathbb{E}[fg] - \mathbb{E}[f]\mathbb{E}[g]|}{\|f\|_p \|g\|_q}, \tag{6.2}$$

where p and q are Hölder pairs.

A possible choice for the function spaces is $\mathcal{F} = \mathcal{G} = L^2$, which is referred to as strong mixing and naturally implies ergodicity in L^2 when considering $f = g$. There are various valid choices and many are discussed in (Bradley, 2005; Hang et al., 2017).

Here, we propose to consider unit balls in reproducing kernel Hilbert spaces for $\mathcal{F}$ and $\mathcal{G}$, which has to the best of our knowledge not been done before.

Definition 37 (Mixing)**.** *Assume $\mathcal{F}$ and $\mathcal{G}$ are unit balls in an RKHS. We call a system mixing if for all $f \in \mathcal{F}$ and $g \in \mathcal{G}$*

$$\mathrm{Cov}(f(X_t), g(X_{t+a})) \to 0. \tag{6.3}$$

Later, we will investigate this property in more detail and leverage powerful estimators in form of the Hilbert-Schmidt independence criterion to determine mixing properties. Estimators for the speed of mixing are usually a critical issue when working with mixing arguments. In related work, the speed and type of mixing is almost exclusively postulated. In contrast, we test if a process is mixing and estimate the actual speed.

An important special case that we will investigate in detail are state-space models or Markov chains with continuous state spaces of the type $X_{k+1} = \phi(X_k) + \epsilon_k$, where ϕ is an appropriate dynamics function and ϵ_k the process noise. This system description is highly relevant in systems and control theory and more recently, reinforcement learning. Further, we will also consider chaotic systems, where $\epsilon_k \equiv 0$. These are deterministic and violate common probabilistic mixing assumptions.

6.2.2 Problem Formulation

Consider two stationary and mixing (cf. Def. 36) systems $\{X_k\}$ and $\{Y_k\}$ with stationary distributions $\mathbb{P}_X$ and $\mathbb{P}_Y$. We want to decide whether $\{X_k\}$ and $\{Y_k\}$ are different based on the data streams $X = \{X_0, X_1, \dots, X_n\}$ and $Y = \{Y_0, Y_1, \dots, Y_n\}$. We assume $X_k, Y_k \in S$ and, in general, $X_0 \neq Y_0$. Further, we assume that the dynamical systems have converged to their stationary distribution.

We propose to compare dynamical systems by testing whether their stationary probability measures coincide. Thus, we obtain the null hypothesis

$$H_0 : \mathbb{P}_X = \mathbb{P}_Y, \tag{6.4}$$

which we try to reject with high confidence. For our method, it is not necessary to estimate or construct any intermediate objects such as the dynamics function f, nor the measures $\mathbb{P}_X$ and $\mathbb{P}_Y$.

The main challenge lies in coping with the autocorrelations within the data streams. These autocorrelations are critical and void commonly used concentration results, such as the famous Hoeffding's or McDiarmid's inequalities.

We consider a two-sample setting between two data streams X and Y. However, this can easily be applied to settings where we want to investigate whether a given model coincides with reality. Then, samples obtained through sensor measurements can be compared with samples generated by simulating a given model.

6.3 Technical Preliminaries

The main idea of this chapter can be summarized as generalizing kernel two-sample tests (Gretton et al., 2012a) to dynamical systems through a data-based mixing approach. Essentially, we propose to wait *long enough* between consecutive samples, which is made precise in Sec. 6.4.2 and thus, enforcing negligibly small autocorrelations. We begin by summarizing key results from kernel two-sample tests and kernel mean embeddings.

6.3.1 Kernel Two-sample Test

An elegant and efficient comparison of probability distributions can be achieved with kernel two-sample tests (Gretton et al., 2012a). The distributions are embedded into an RKHS, where it becomes tractable to compute certain metrics on the space of probability distributions such as the MMD. The following definitions and theorems are taken from Gretton et al. (2012a).

Definition 38 (MMD)**.** *Let (S, d) be a metric space and let $\mathbb{P}_X, \mathbb{P}_Y$ be two Borel probability measures defined on S. Further, let $\mathcal{F}$ be the unit ball in an RKHS on S. We define the maximum mean discrepancy by*

$$\mathrm{MMD}^2[\mathbb{P}_X, \mathbb{P}_Y] = \sup_{g \in \mathcal{F}} (\mathbb{E}_{\mathbb{P}_X}[g] - \mathbb{E}_{\mathbb{P}_Y}[g])^2. \tag{6.5}$$

The MMD yields a semi-metric between probability distributions and can be efficiently estimated by embedding the distributions into an RKHS $\mathcal{H}$ with the aid of kernel mean embeddings (Muandet et al., 2017). It is a challenging problem to compute (6.5) directly since $\mathcal{F}$ is usually infinite-dimensional. However, by kernelizing it, we can estimate (6.5) from data.

Theorem 39. *Assume k is a kernel and $\mathcal{F}$ is again the unit ball in the corresponding RKHS $\mathcal{H}$. Further, assume $(X_1, \dots, X_n)$ and $(Y_1, \dots, Y_m)$ are drawn i.i.d. from $\mathbb{P}_X$ and $\mathbb{P}_Y$, respectively. Then, an unbiased estimate of (6.5) is given by*

$$\mathrm{MMD}_b^2[X, Y] = \frac{1}{n^2} \sum_{i,j=1}^{n} k(X_i, X_j) + \frac{1}{m^2} \sum_{i,j=1}^{m} k(Y_i, Y_j) - \frac{2}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} k(X_i, Y_j). \tag{6.6}$$

The additional requirement of a characteristic kernel ensures that the embedding of the probability distribution is injective and, thus, a metric is obtained. The kernel k can, for example, be chosen as a Gaussian kernel since it is well known to be characteristic (Gretton et al., 2012a).

Theorem 40. *Assume k is a characterstic kernel and $\mathcal{F}$ is the unit ball in the corresponding RKHS $\mathcal{H}$. Then $\mathrm{MMD}^2[\mathbb{P}_X, \mathbb{P}_Y] = 0$ if, and only if, $\mathbb{P}_X = \mathbb{P}_Y$.*

Essentially, we do not require any prior knowledge or parameterization of $\mathbb{P}_X$ and $\mathbb{P}_Y$. Access to i.i.d. samples from these distributions is sufficient. In practice, however, we only have access to finitely many data points and, thus, receive an estimate of the MMD from (6.6). This estimate is expected to have some deviation, i.e., even for identical distributions, the test statistic will be larger than zero. Therefore, we need finite sample bounds that quantify the convergence speed of the empirical MMD to obtain confidence bounds. Gretton et al. (2012a) introduce several such bounds of the type

$$\mathbb{P}\left[|\mathrm{MMD}_b[X,Y] - \mathrm{MMD}[\mathbb{P}_X,\mathbb{P}_Y]| \geq \kappa(\alpha,n)\right] \leq \alpha. \tag{6.7}$$

Under the nullhypothesis $\mathbb{P}_X = \mathbb{P}_Y$, we can obtain the rejection region $\mathrm{MMD}_b[X,Y] \geq \kappa(\alpha,n) = \sqrt{2\frac{K}{n}}(1+\sqrt{2\log\alpha^{-1}})$ for a test with level α, where K is the supremum of the kernel (Gretton et al., 2012a, Corollary 9). However, these results rely on the independence assumption and, hence, cannot be used for comparing dynamical systems.

6.3.2 Hilbert-Schmidt Independence Criterion

The Hilbert-Schmidt independence criterion (HSIC) (Gretton et al., 2008) quantifies dependence between random variables. Generally, two random variables X and Y are independent if their joint distribution factorize, i.e., $\mathbb{P}_{X,Y} = \mathbb{P}_X \otimes \mathbb{P}_Y$, where $\otimes$ denotes the tensor product. Estimating the involved objects from data is usually intractable. Instead, the difference in MMD can elegantly be expressed through the HSIC.

Definition 41 (HSIC, Sejdinovic et al. (2013, Def. 11))**.** *Let $X \sim P_X$ and $Y \sim P_Y$ be random variables with joint distribution $P_{X,Y}$. The HSIC is defined as*

$$\mathrm{HSIC}(X,Y) = \|P_X \otimes P_Y - P_{X,Y}\|_{\mathrm{MMD}}. \tag{6.8}$$

Similar to the kernel two-sample test, it is possible to express (6.8) in terms of kernel evaluations. Further, it is also possible to provide high confidence bounds and thus, obtain an efficient statistical test.

As the name suggests, the HSIC is closely related to Hilbert-Schmidt operators. These well-behaved operators are well investigated in functional analysis and in general, are bounded operators between Hilbert spaces. Further, the space of Hilbert-Schmidt operators between two reproducing kernel Hilbert spaces $\mathcal{H}$ and $\mathcal{G}$ forms itself a Hilbert space, which is isomorphic to the product space $\mathcal{H} \otimes \mathcal{G}$ given by the product kernel (Muandet et al., 2017, Page 35).

Here, we want to emphasize the connection between the HSIC and the covariance operator $\mathcal{C}_{XY}$ in terms of the Hilbert-Schmidt norm (Muandet et al., 2017, Eq. 3.37)

$$\|\mathcal{C}_{X,Y}\|_{\mathrm{HS}} = \mathrm{HSIC}(X,Y) \tag{6.9}$$

and the representation of $\mathcal{C}_{X,Y}$ as the unique bounded operator that satisfies the property

$$\langle g, \mathcal{C}_{X,Y} f \rangle_{\mathcal{G}} = \mathrm{Cov}[g(Y), f(X)] \tag{6.10}$$

for all $g \in \mathcal{G}$ and $f \in \mathcal{H}$. Equivalently, the covariance operator can also be defined in terms of tensor spaces (Muandet et al., 2017, Sec. 3.2), however, (6.10) connects nicely to standard mixing expressions (cf. Eq. (6.3)). Further, the HSIC framework provides rich results such as efficient estimators and concentration results.

The general framework is highly flexible and can deal with a variety of objects. For our problem, we can use a simplified setting, where both systems belong to the same space, which is consistent with the setup for the kernel two-sample test. Estimations of (6.8) in terms of kernel evaluations can be found in (Gretton et al., 2008, Equation (4)).

6.3.3 Joint Independence – dHSIC

Pfister et al. (2018) extended the HSIC to d-dimensional random vectors and thus, investigate

$$\mathrm{dHSIC}(X) = \|P_{X_1} \otimes \ldots \otimes P_{X_d} - P_{X_1,\ldots,X_d}\|_{\mathrm{MMD}}. \tag{6.11}$$

Similarly as for the kernel two-sample test and the classical HSIC independence test, it is possible to quantify the convergence speed and thus, obtaining a threshold $\kappa(\alpha, n)$ for statistical testing. Intuitively, this should be the independence notion that we need for the kernel two-sample test. However, we use a slightly different property, which we introduce in Def. 47.

6.4 MMD-Mixing

In this section, we will focus on data from one system $\{X_k\}$ and investigate the temporal dependencies. We assume access to multiple independent trajectories, which we indicate through superscripts $\{X_k^{(i)}\}$. Due to the ergodicity and stationarity assumptions, we obtain a well-defined underlying distribution $\mathbb{P}$ for which we can test. Next, we will introduce the new concept of MMD-mixing that quantifies the decay of autocorrelations with respect to the MMD and connect back to the HSIC.

6.4.1 Time Shifts and MMD-mixing

Let the trajectory $X_0, X_1, \ldots, X_n$ be subject to a given sampling rate. Generally, autocorrelations decay over time, and far apart samples are approximately independent if the underlying system is mixing. Hence, we propose to increase the time between consecutive samples to reduce dependencies. The slower sampling rate is denoted through the time shift $a \in \mathbb{N}$ and yields data $X_0, X_a, X_{2a}, \dots, X_{an}$. Essentially, the question is how to determine and estimate a to ensure approximately independent data points $X_0, X_a, X_{2a}, \dots, X_{an}$. We begin with a simplified setting and assume a sample from the stationary measure $X_0 \sim \mathbb{P}_{X_0} = \mathbb{P}$.

Definition 42 (MMD-mixing)**.** *We call a process* MMD-mixing *if*

$$\|\mathbb{P}_{X_0} \otimes \mathbb{P}_{X_a} - \mathbb{P}_{X_0, X_a}\|_{\text{MMD}} \to 0 \quad \textit{for } a \to \infty. \tag{6.12}$$

This definition only considers the distributions at two points in time. Due to the stationarity of the system, we can move the timeshift through time and also consider different pairs in time. Due to the connection to Hilbert-Schmidt theory and covariance operators, it is also possible to consider expressions similar to (6.10).

Proposition 43. *Let $\{X_k\}$ be an MMD-mixing process. Then,*

$$\text{HSIC}(X_0, X_a) \to 0 \quad \textit{for } a \to \infty. \tag{6.13}$$

We assume that the underlying kernel k is characteristic and refer to it as the base kernel. Further, we assume to have access to m independent trajectories. In this setting, we can readily apply the HSIC (6.8) framework. In particular, we pick two points in time from each trajectory, $X_0^{(i)}$ and $X_a^{(i)}$. Then, we divide the data into $X_0 = \{X_0^{(1)}, X_0^{(2)}, \dots, X_0^{(m)}\}$ and $X_a = \{X_a^{(1)}, X_a^{(2)}, \dots, X_a^{(m)}\}$. The sets are, per construction, i.i.d. within themselves. Next, we can compute $\text{HSIC}(X_0, X_a)$ and iteratively increase a. If we pick a large enough, the HSIC will eventually become arbitrarily small. MMD-mixing ensures that $\text{HSIC}(X_s, X_{s+a}) \to 0$ for $a \to \infty$. For practical algorithms, we will fix a small $\epsilon > 0$ and enforce $\text{HSIC}(X_s, X_{s+a^*}) < \epsilon$. In our experiments, we pick ϵ as the standard test threshold of the HSIC (cf. Fig. 6.3).

6.4.2 Extended MMD-mixing

Next, we extend our arguments to subtrajectories instead of considering two single points. Similarly as before, we use the notation $\mathbb{P}_{X_0,\dots,X_s}$ and $\mathbb{P}_{X_{s+a},\dots,X_{2s+a}}$ for the distributions of the subtrajectories of length s and $\mathbb{P}_{(X_0,\dots,X_s),(X_{s+a},\dots,X_{2s+a})}$ for the joint distribution of the subtrajectories.

Definition 44 (Extended MMD-mixing)**.** *We call a process* extended MMD-mixing *if*

$$\|\mathbb{P}_{X_0,\dots,X_s} \otimes \mathbb{P}_{X_{s+a},\dots,X_{2s+a}} - \mathbb{P}_{(X_0,\dots,X_s),(X_{s+a},\dots,X_{2s+a})}\|_{\text{MMD}} \to 0 \quad \textit{for } a \to \infty. \tag{6.14}$$

Clearly, we require an appropriate kernel to extend the MMD to joint distributions. In particular, tensor products of the base kernel need to be strong enough to distinguish the joint distributions. Szabó and Sriperumbudur (2018) discuss various tensor constructions, which we will leverage here.

Lemma 45 (Choice of Kernel I)**.** *Let k be a characteristic kernel. Then $k^s = \otimes_{i=1}^s k$ is also characteristic.*

Proof. The statement follows directly from Szabó and Sriperumbudur (2018, Theorem 4), which considers a more general problem setting. □

Due to the tensor construction, we naturally obtain MMD-mixing with respect to k as introduced in Definition 42, for a process that is extended MMD-mixing with respect to k^s.

6.4.3 Joint Independence

To apply the mixing results to kernel two-sample testing, we require one more step. We need joint independence between all samples. Intuitively, this coincides with the dHSIC framework (cf. (6.11)) and can be implemented through more sophisticated tensor kernels that embed multiple data points or subtrajectories simultaneously.

Lemma 46 (Choice of Kernel II)**.** *Assume k^s is a characteristic kernel. Then the tensor kernel $k^{s,n} = \otimes_{i=1}^n k^s$ is an $\mathcal{I}$-characteristic kernel, which makes the kernel suitable for joint independence testing.*

Proof. Follows from (Szabó and Sriperumbudur, 2018, Theorem 4). □

To combine mixing with kernel two-sample testing, we require the following technical assumption.

Definition 47 (Approximately ϵ-independent)**.** *Let $\{X_k\}$ be an MMD-mixing process. We call data $X = X_{a^*}, X_{2a^*}, \dots, X_{na^*}$ approximately ϵ-independent if there is a time shift a^* and threshold $\kappa(\epsilon, n)$ that yields*

$$\mathbb{P}[\widehat{\mathrm{MMD}}_b(X, \bar{X}) \geq \kappa] < \epsilon, \tag{6.15}$$

where $\bar{X}$ is data that has been sampled independently from the stationary distribution $\mathbb{P}$.

An important technical detail here is the fact that we consider the MMD with respect to the kernel k and not the tensor kernel $k^{s,n}$. In practice, we apply a level ϵ HSIC test to multiple independent trajectories in order to determine an a^*, which satisfies (6.15).

6.4.4 Connections to Other Mixing Notions

There are various types of mixing that essentially all describe the decay of autocorrelations. An extensive discussion of the relationship between different measures of dependencies can be found in Bradley (2005). The importance of covariance-based expressions for mixing is utilized in Bradley et al. (1987) to investigate how they can dominate each other.

Mixing properties are notoriously difficult or even impossible to estimate, and many types of mixing do not apply to large classes of dynamical systems (Hang et al., 2017). Our proposed type of mixing can be estimated from data and yields advantageous theoretical properties. In McDonald et al. (2011), the β-mixing coefficient is estimated through involved density estimations. While the authors emphasize that they solve a more difficult problem to obtain a solution to a simpler one, this is still one of the few existing approaches to estimate the speed of mixing.

We start with defining the β-mixing coefficient as in (McDonald et al., 2011):

$$\beta(a) = \sup_s \|\mathbb{P}^s_{-\infty} \otimes \mathbb{P}^{\infty}_{s+a} - \mathbb{P}_{s,a}\|_{\mathrm{TV}}, \tag{6.16}$$

where $\mathbb{P}^s_{-\infty}$ is the joint distribution of the states $\{X_t\}^s_{t=-\infty}$ and $\mathbb{P}^{\infty}_{s+a}$ of $\{X_t\}^{\infty}_{t=s+a}$. With $\mathbb{P}_{s,a}$ we denote the joint distribution of the objects around the tensor sign, here
$(\{X_t\}^s_{t=-\infty}, \{X_t\}^{\infty}_{t=s+a})$ and use $\|\cdot\|_{\mathrm{TV}}$ for total variation. The process is β-mixing if $\beta(a) \to 0$ for $a \to \infty$.

MMD-mixing is closely related with (6.16) and yields lower bounds.

Lemma 48. *A β-mixing process is MMD-mixing for any bounded kernel.*

Proof. This property follows by considering Hilbert space embeddings of probability distributions. In particular, Sriperumbudur et al. (2010, Theorem 21 (iii)) shows that

$$\|\mathbb{P} - \mathbb{Q}\|_{\mathrm{MMD}} \leq C\|\mathbb{P} - \mathbb{Q}\|_{\mathrm{TV}}, \tag{6.17}$$

where C is the supremum of the corresponding kernel. □

The other direction does not always hold. For instance, deterministic dynamical systems are, in general, not β-mixing (Hang et al., 2017).

Lemma 49. *A $\mathcal{C}$-mixing process with respect to the underlying function space $\mathcal{C}$ is MMD-mixing with respect to the kernel k, if $\mathcal{H} \subset \mathcal{C}$, where $\mathcal{H}$ is the corresponding RKHS.*

Proof. Following Definition 2 in Hang et al. (2018), $\mathcal{C}$-mixing is essentially defined as (6.2), where $\mathcal{F}$ is chosen as the function space $\mathcal{C}$ and $\mathcal{G}$ as L^1 on the natural filtration of the system. By considering smaller spaces for $\mathcal{F}$ and $\mathcal{G}$, such as $\mathcal{H}$, we directly obtain the result. $\square$

In particular, if k is the squared exponential kernel then the corresponding RKHS is well-investigated Steinwart and Christmann (2008). In particular, the RKHS is contained in common choices for $\mathcal{C}$, such as $\mathrm{BV}(S)$, $\mathrm{Lip}(S)$, and $C^1(S)$.

Further, recent results show that convergence in MMD metrizes weak convergence in the space of probability distributions (Simon-Gabriel et al., 2020). Thus, convergence in MMD is applicable to discrete data and Dirac distributions. This may be particularly relevant when considering mixing properties of deterministic dynamical systems even further.

In practice, mixing is usually exponentially fast in the gap a. In all of our numerical experiments, it was sufficient to estimate a single time shift a^* in the MMD-mixing sense between two data points. The joint dHSIC estimation yields stronger theoretical properties, however, might also induce some conservatism into the estimation.

6.5 Two-sample Test for Dynamical Systems

Next, we utilize mixing to state our main result: a kernel two-sample test for dynamical systems. Due to MMD-mixing, we are able to enforce arbitrarily small dependencies between consecutive samples. In particular, we use our notion of approximately ϵ-independent data (cf. (6.15)) to adjust the test threshold accordingly. For $a \to \infty$, we actually recover the i.i.d. setting from Gretton et al. (2012a).

Proposition 50. *Assume $\{X_k\}$ and $\{Y_k\}$ are stationary, ergodic, and MMD-mixing dynamical systems with distributions $\mathbb{P}_X, \mathbb{P}_Y$. Further, assume data $X_{a^*}, X_{2a^*}, \ldots, X_{na^*}$ and $Y_{a^*}, Y_{2a^*}, \ldots, Y_{na^*}$ are sampled i.i.d. from $\mathbb{P}_X$ and $\mathbb{P}_Y$, respectively. If we obtain for the empirical estimate (6.6) that*

$$\mathrm{MMD}_b^2[\mathbb{P}_X, \mathbb{P}_Y] > \kappa(n, \alpha), \tag{6.18}$$

then we can conclude with probability $1-\alpha$ that $\mathbb{P}_X \neq \mathbb{P}_Y$.

In practice, the autocorrelations will always be greater than zero. Also, it is important that either both systems have the same mixing speed or a^* is chosen with respect to the system with the slower mixing rate.

We state the main result that, in contrast to prior work, foregoes the need for independence assumptions.

Theorem 51. *Assume the same setting as above, however, instead of i.i.d. data, we assume that a^* is a time shift that yields approximately ϵ-independent (cf. 6.15) data $X = X_{a^*}, X_{2a^*}, \dots, X_{na^*}$ and $Y = Y_{a^*}, Y_{2a^*}, \dots, Y_{na^*}$.*

If we obtain for the empirical estimate (6.6) that

$$\mathrm{MMD}_b^2[\mathbb{P}_X, \mathbb{P}_Y] > \kappa(n, \alpha), \tag{6.19}$$

then we can conclude with probability $1 - \alpha'$ that $\mathbb{P}_X \neq \mathbb{P}_Y$, where $\alpha' = \frac{1}{3}(\alpha + 2\epsilon)$

Proof. First, we will decompose the test statistic into the i.i.d. problem and a second term that captures the dependency in the data. Assume $\bar{X}, \bar{Y}$ are i.i.d. data sets (ghost samples) that are drawn from $\mathbb{P}_X$ and $\mathbb{P}_Y$, respectively. A similar argument is frequently used for symmetrization and referred to in Gretton et al. (2012a, P. 736).

$$\begin{aligned}
&|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(X, Y)| && (6.20)\\
=&|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y}) + \mathrm{MMD}_b(\bar{X}, \bar{Y}) - \mathrm{MMD}_b(X, Y)| && (6.21)\\
\leq&|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y})| + |\mathrm{MMD}_b(\bar{X}, \bar{Y}) - \mathrm{MMD}_b(X, Y)| && (6.22)\\
=&||\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y})| + |\|\hat{\mu}_{\bar{X}} - \hat{\mu}_{\bar{Y}}\|_{\mathcal{H}} - \|\hat{\mu}_X - \hat{\mu}_Y\|_{\mathcal{H}}| && (6.23)\\
\leq&|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y})| + \|\hat{\mu}_{\bar{X}} - \hat{\mu}_{\bar{Y}} - \hat{\mu}_X + \hat{\mu}_Y\|_{\mathcal{H}} && (6.24)\\
\leq&|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y})| + \mathrm{MMD}_b(\bar{X}, X) + \mathrm{MMD}_b(\bar{Y}, Y) && (6.25)
\end{aligned}$$

We use the identity $\mathrm{MMD}_b(X, Y) = \|\hat{\mu}_X - \hat{\mu}_Y)\|_{\mathcal{H}}$ (Muandet et al., 2017, Eq. 3.31) and apply the inverse triangle inequality. The first term follows directly from Gretton et al. (2012a) (cf. Eq. 6.7) and can be bounded by κ. By design, the time shift a^* was chosen to induce the concentration

$$\mathbb{P}\left[\mathrm{MMD}_b(X, \bar{X}) > \kappa\right] \leq \epsilon \tag{6.26}$$

and respectively also for $\mathrm{MMD}_b(Y, \bar{Y})$. Thus, we obtain in total

$$\begin{aligned}
&\mathbb{P}\left[|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(X, Y)| > \kappa\right] && (6.27)\\
\leq&\mathbb{P}\left[|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y})| + \mathrm{MMD}_b(\bar{X}, X) + \mathrm{MMD}_b(\bar{Y}, Y) > \kappa\right] && (6.28)\\
\leq&\tfrac{1}{3}\mathbb{P}\left[|\mathrm{MMD}(\mathbb{P}_X, \mathbb{P}_Y) - \mathrm{MMD}_b(\bar{X}, \bar{Y})| > \kappa\right] + \tfrac{1}{3}\mathbb{P}[\mathrm{MMD}_b(\bar{X}, X) > \kappa] + \tfrac{1}{3}\mathbb{P}\left[\mathrm{MMD}_b(\bar{Y}, Y) > \kappa\right] && (6.29)\\
\leq&\frac{1}{3}(\alpha + 2\epsilon). && (6.30)
\end{aligned}$$

□

Remark 52. *By adapting the concentration results inside the kernel two-sample test, i.e., McDiarmid's inequality, we can directly embed significant autocorrelations in the test statistic and potentially be more data-efficient and have tighter bounds. These results, however, would require further technical assumptions (cf. Assumption*

3.1. in (Chérief-Abdellatif and Alquier, 2022)) and are left for future work. Here, we focus on introducing an efficient, sound, and practically relevant statistical test for dynamical systems.

In practice, we usually do not have access to the full state X_k. Instead we receive measurements $X'_k = g(X_k) + \xi_k$, where g is an observation function and $\xi_k \overset{\text{iid}}{\sim} \mathbb{P}_\xi$ measurement noise. Intuitively, the function g could be regarded as sensors that measure some quantity that depends on the underlying system. Thus, we could also infer different systems when, e.g., the measurement noise or the sensors are different. Further, it is not always possible to reconstruct the state, and appropriate observability assumptions would be required for this. However, this is not due to our test but an issue of the problem itself since the true underlying state is unknown. Nonetheless, we are able to apply the proposed test to measurements X'_k by considering the pushforward of the measure $g(\mathbb{P}_X)$ together with $\mathbb{P}_\xi$.

Proposition 53. *Assume the same setting as in Proposition 50, however, with noisy measurements $X'_k = g(X_k) + \xi_k$ and $Y'_k = h(Y_k) + \nu_k$ and independent noise. If $\mathrm{MMD}_b^2[\mathbb{P}_{X'}, \mathbb{P}_{Y'}] > \kappa(n, \alpha)$, then we conclude that $\mathbb{P}_{X'} \neq \mathbb{P}_{Y'}$ with high probability.*

In general, it is not clear what states to choose for an appropriate representation of a dynamical system, e.g., to accurately model human walking. If we obtain rich information through sensor measurements then this can often be sufficient for subsequent downstream tasks (cf. Section 6.7).

6.6 Illustrative Examples

In this section, we illustrate two critical properties of our method: i) respecting the estimated time shift a^* yields samples whose distribution is indistinguishable from the stationary distribution, ii) violating the estimated time shift a^* leads to clustering effects that skew and bias the empirical distributions. More details on all experiments are provided in the appendix.

In practice, it is usually sufficient to consider data from two points in time $\mathbb{P}_s$ and $\mathbb{P}_{s+a}$ – in particular, when kernel two-sample testing is also based on points and not subtrajectories. Thus, we estimate a^* based on MMD-mixing (cf. Def. 42).

6.6.1 Linear Time-invariant System

Linear time-invariant (LTI) systems are prevalent in control and systems theory due to many analytically tractable properties. In particular, we can explicitly determine

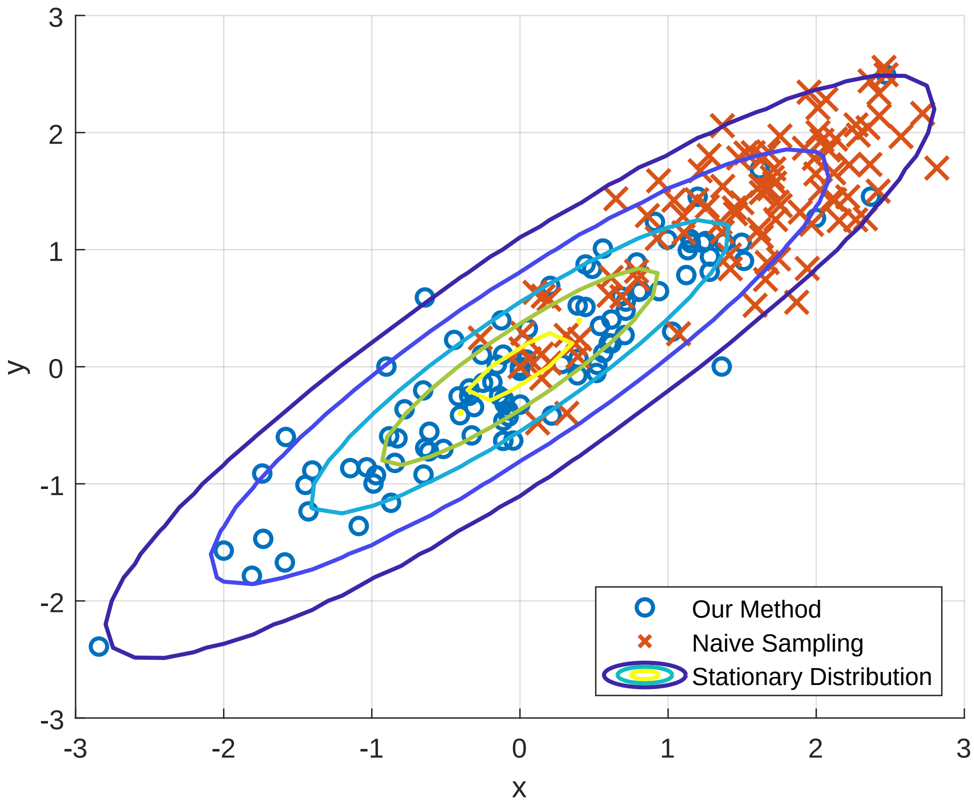


Figure 6.1: Scatter plot of an illustrative 2-dimensional LTI system. The system was designed to yield slow mixing times and initialized at $X_0 = 0$. The red crosses represent the first 100 states $X_0, X_1, \ldots, X_{100}$. The blue circles represent states with an enforced time shift of $a^* = 75$ between samples. The stationary distribution of the system is illustrated as a contour plot.

the stationary distribution (cf. (A.2) in the appendix) and, thus, draw i.i.d. samples. Consider the dynamics

$$X_{k+1} = AX_k + \epsilon_k, \tag{6.31}$$

where $\epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \Sigma)$. Further, assume all eigenvalues of $A \in \mathbb{R}^{d \times d}$ are located within the unit circle and $x_0 = 0$ to avoid potential transient behavior. We can now quantify the speed of mixing directly through the eigenvalues of A and Σ. If A has eigenvalues close to the boundary of the unit sphere, then this results in slow mixing. The same holds for small process noise. On the contrary, small eigenvalues of A and large noise result in rapid mixing. An intuitive corner case is $A = 0$, which yields perfectly independent samples.

In Fig. 6.1, we illustrate the behavior of a two-dimensional slowly mixing system (6.31). In red, we plot the first 100 states of the system $X_1, X_2, \ldots, X_{100}$, and in blue, states with an enforced time shift of $a^* = 75$ between consecutive samples $X_{a^*}, X_{2a^*}, \ldots, X_{100a^*}$. The estimated time shift $a^* = 75$ is obtained in the MMD-

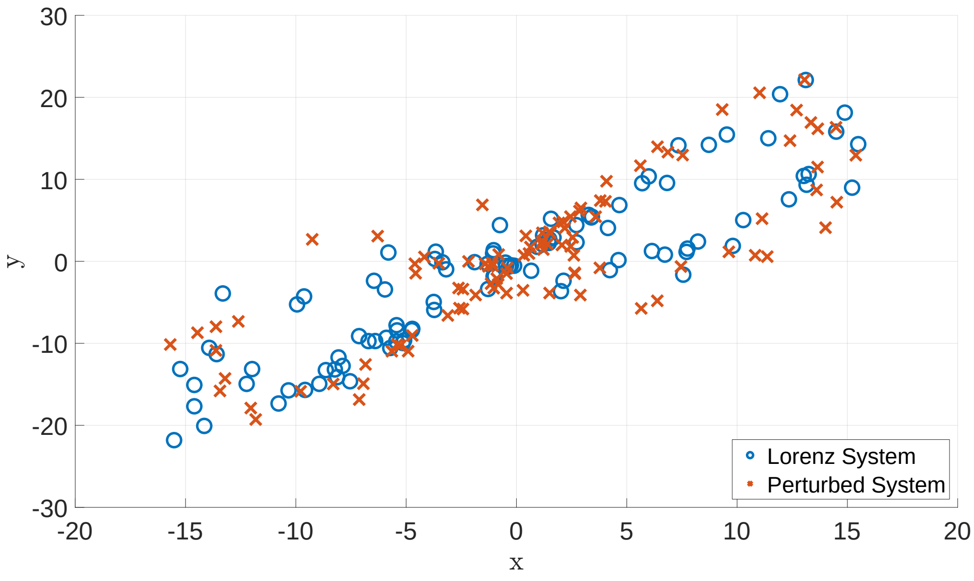


Figure 6.2: Scatter plot of the classical Lorenz system in blue circles (Eq. (6.32) – (6.34)) and samples from a system with slightly perturbed parameters in red crosses. Both systems are randomly initilized and sampled at time instances $t_1 = 20, t_2 = 40, \ldots, t_{100} = 2000$. To the human eye, the distributions look slightly different.

mixing sense as described in Sec. 6.4.2 and ensures that the HSIC is below the test threshold (cf. Fig. A.1 in the appendix).

In Fig. 6.1, we further show a contour plot of the stationary distribution. Samples that were drawn based on our method coincide with the stationary distribution. The first 100 states, on the other hand, cluster in one region of the state space and are subject to heavy auto-correlations. The red crosses are clearly not representative of the stationary distribution. To investigate this further, we applied kernel two-sample tests to distinguish samples that are directly drawn from the stationary distribution and samples drawn based on our method with appropriate time shifts. As expected, this turned out to be impossible, and we cannot distinguish between the two data sets. Details are given in the appendix.

We want to emphasize that mixing can be arbitrarily slow. In particular, it is possible that the system does not mix at all (Simchowitz et al., 2018). With the proposed method, we would notice this since we would not be able to estimate an a^*. Thus, we can decide whether the kernel two-sample test is applicable or not. We have also constructed non-mixing examples and obtained a constant HSIC that does not decrease over time (cf. appendix).

6.6.2 Lorenz Attractor

To illustrate the usefulness of the new mixing notion we present the example of the Lorenz system, which is illustrated in Fig. 6.2 and given by the following equations:

$$\dot{x} = 10(y - x) \tag{6.32}$$

$$\dot{y} = 28x - y - xz \tag{6.33}$$

$$\dot{z} = xy - \frac{8}{3}z. \tag{6.34}$$

The Lorenz attractor is a famous chaotic and deterministic dynamical system that is known to mix in a topological sense (Luzzatto et al., 2005). Other notions, such as β-mixing, are too strong and not suitable here. In the appendix, we provide empirical evidence that the Lorenz system mixes with respect to the herein introduced notion of MMD-mixing. Connecting topological mixing on a rigorous level with MMD-mixing remains for future work.

Further, we show numerically that we can distinguish between two systems with slightly different parameters and obtain the required properties of the kernel two-sample test. For the estimated a^*, the test is well-behaved. When we chose the estimated time shift too small, then the amount of false positives explodes.

6.6.3 $\mathcal{C}$-mixing Systems

We also consider the three examples that are discussed in Hang et al. (2018) and are provably $\mathcal{C}$-mixing. Due to the Gaussian kernel that we use and the choices for $\mathcal{C}$ (Lip(S) and BV(S), cf. Hang et al. (2018) for details), $\mathcal{C}$-mixing directly implies MMD-mixing. The empirical results confirm the MMD-mixing property.

We considered the following systems:

β-map: For $\beta > 1$ and $x_0 \in (0, 1)$, the dynamical system is defined by

$$x_{k+1} = \beta x_k \mod 1. \tag{6.35}$$

Logistic map: For $x_0 \in (0, 1)$, the logistic map is defined by

$$x_{k+1} = 4x_k(1 - x_k). \tag{6.36}$$

Gauss map: For $x_0 \in (0, 1)$, the Gauss map is defined by

$$x_{k+1} = \frac{1}{x_k} \mod 1. \tag{6.37}$$

For all examples, the speed of mixing is extremely fast and after $a^* = 10$, the data is close to independent. We initialized x_0 uniformly on the interval $(0, 1)$ and used $\beta = e$.

Interestingly, the Lebesgue densities of the stationary distributions are also known and stated in Hang et al. (2018). This would allow for kernel two-sample testing, exactly as done for the OU-process in App. A.1.1. Since the mixing is extremely fast here, we expect the same result—time shifted data that respects the speed of mixing is indistinguishable from data that has been drawn directly from the stationary distribution.

6.7 Experimental Example – Human Walking

We apply the developed kernel two-sample test to real-world experimental data that will be made available. We consider gait data of human subjects walking on a treadmill. Detecting characteristics and alterations in human gait is a highly relevant problem in disease prediction, diagnosis and progress monitoring as well as in biometrics (Nguyen et al., 2019; Gaßner et al., 2020; Muro-De-La-Herran et al., 2014). An example data set with a known ground truth label is obtained by letting subjects walk with and without a knee orthosis. Our goal is to classify each measured trajectory correctly with the labels *orthosis* and *no orthosis*.

6.7.1 Data Collection

The inertial measurement unit (IMU) data of foot motion were collected from 38 healthy subjects without any restrictions in gait or illnesses that affect their walking ability. The data collection was conducted in the movement analysis laboratory of one of the authors' universities on a Mercury Med treadmill. The IMU sensors were attached to the test subjects' shoes using velcro straps. The measurements were taken for 90 seconds each trial under the following conditions: walking at very slow ($1.5\,\mathrm{km\,h^{-1}}$), slow ($3\,\mathrm{km\,h^{-1}}$), slow with simulated gait pathology, and normal walking speed ($5\,\mathrm{km\,h^{-1}}$). For the simulated gait pathology, the mobility of the left knee joint was restricted using a knee orthosis, which was fixed in a neutral position to disable further extension or flexion of the joint. The subjects were asked to stand still with both feet next to each other for 3 seconds at the beginning and the end of each trial, Before the trials, the subjects were able to practice walking on the treadmill. They were allowed to use the handrail of the treadmill if necessary. For three subjects, the orthosis experiment could not be carried out. An approval from the local ethics committee was obtained.

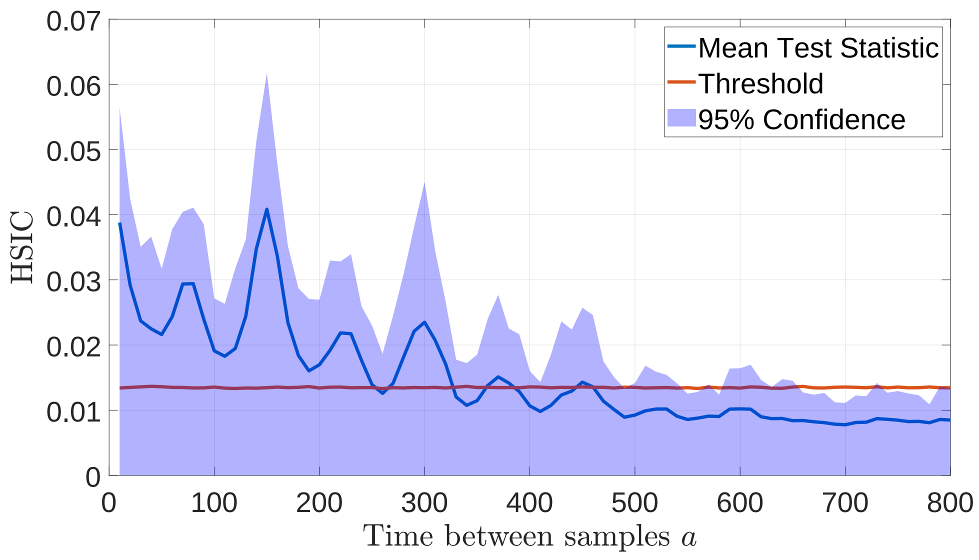


Figure 6.3: Mixing properties of gait data. On the x-axis, we depict the time shift a between consecutive samples. One time step corresponds to 0.01 seconds. The y-axis shows the dependence between data points with respect to the corresponding time shift. The initial point is randomized, and the estimation is repeated 50 times. Depicted is the mean of the test statistic and the 95% upper confidence bound. We also show the threshold κ of the independence test. When the blue line is below the red line, it is not possible to infer statistical dependence between the data points.

6.7.2 Description of the Statistical Test

We consider the raw gyroscopic data of the left foot for 35 subjects. The gyroscopic data is three-dimensional and consists of roughly 14 000 data points per trajectory.

Mixing Properties

First, we quantify the mixing properties of human walking. We estimate MMD-mixing (cf. Sec. 6.4.2) by applying the HSIC to the 35 subjects. We consider the trials with and without the orthosis simultaneously, which yields 70 independent trajectories. We draw an initial point X_k from a uniform distribution between $k = 2000$ and $k = 4000$ and fix that point for all trajectories. Afterward, we compute $\mathrm{HSIC}(X_k, X_{k+a})$ for various values of a. In Fig. 6.3, the results are illustrated and we can see the decrease of dependencies. To exclude numerical artifacts, we repeat the estimation of the mixing properties 50 times with randomly chosen initial points.

Classification

We compare MMD-based classification against standard baselines for the 70 trajectories.

MMD-based classification: We choose one trajectory of interest, for which we forget the correct label, and separate it from all other trajectories, for which the correct label is known. We pick a random initial point X_0 uniformly distributed between $k = 2000$ and $k = 3000$. After time shifting the data with respect to a^*, we estimate the MMD (6.6) between the trajectory of interest and all other trajectories. Then, we use the label of the trajectory with the smallest MMD to label the unlabeled trajectory. Intuitively, unrestricted trajectories look more similar among themselves than trajectories with a restricted knee, and vice versa.

Baseline: We compare the proposed approach to common baselines for classification of biomedical data (Bidabadi et al., 2019; Misgeld et al., 2015; Tien et al., 2010). We consider the following features:

- Maximum and minimum value of each dimension;
- The 4 largest frequencies based on a Fourier transform;
- The 2-norm over time and the state dimensions.

In total, this results in 19 features for each trajectory that are used to train linear classifiers – support vector machines (SVM) and logistic regression (LR). Further, we use a 3-fold cross-validation technique. We repeat the training also 1000 times and report the average accuracy and standard deviation in Table 6.1.

6.7.3 Results

Our empirical analysis reveals that human walking mixes with respect to MMD-mixing (cf. Sec. 6.4.2). Further, as illustrated in Fig. 6.3, we can effectively estimate the speed of mixing. After roughly five footsteps, the dependence of data to its past is mostly gone, and we can treat data as independent.

For MMD-based classification, we use a time shift of $a^* = 400$. This results in 25 points per trajectory. A larger choice of a^* around 600 would be closer to our theoretical results. However, due to the limited amount of data, this would reduce the number of available samples even further.

We run all classification algorithms 1000 times and report the average accuracy and standard deviation in Table 6.1. Our method achieves the best accuracy, and we are able to classify 99.99% of the subjects with *no orthosis* correctly. Some very few subjects are repeatedly misclassified when walking with the orthosis, which might be explained using futher insights and data analysis. For the other methods, in contrast, there is no apparent structure in the errors.

Table 6.1: Classification accuracy for labeling the trajectories correctly into the labels *orthosis* and *no orthosis*. Mean accuracy with standard deviation over 1000 repetitions.

Our method	SVM	LR
95.7% $\pm$ 2.4%	86.9% $\pm$ 4.4%	92.5% $\pm$ 3.2%

6.7.4 Discussion

The above results show that the proposed method works well on a practically relevant non-trivial problem and outperforms common baselines. We spent a reasonable amount of time on designing good features in the comparison. While the accuracy of the linear classifiers could potentially be improved by adding additional features, designing such features requires more insight into the problem and system properties, which is unavailable in many applications. In order to improve the accuracy of our method, it would suffice to add more data (i.e., consider longer trajectories). Further, it can directly be applied to a range of similar problems.

Classification and clustering algorithms based on the MMD can be applied in more general settings (Jegelka et al., 2009). Thus, our proposed nearest-neighbor approach for dynamical systems should generalize to more sophisticated clustering algorithms, which could yield unprecedented insights into the behavior of complex dynamical systems.

6.8 Conclusion

We propose a kernel two-sample test for dynamical systems with deep connections to a new type of mixing in MMD. The proposed method is straightforward to use, has only a few parameters, and is model-free. In particular, we are able to estimate the speed of mixing from data in a relevant norm, which was previously not possible. The flexibility and relevance of the proposed method are demonstrated numerically and experimentally on raw motion sensor data. The presented results show the potential for biomedical and engineering applications, which we plan to explore in future work.

Chapter 7

Conclusions

In this thesis, we present event-triggered learning as a new paradigm for applying learning-based techniques to learning-control systems. In particular, we address the question of how to cope with potentially time varying dynamical systems. We show that the very nature of learning-control systems makes the problem different from common static machine learning and control problems. Therefore, we need to be careful when developing methods that can deal with change to ensure meaningful learning outcomes.

7.1 Summary

We introduce ETL on an abstract level as a new paradigm that addresses the issue of applying learning-based techniques to potentially changing learning-control systems. Statistical tests and learning triggers are at the heart of ETL and monitor suitable signals to quantify the probability of change. Due to the challenging properties of learning-control systems we had to develop new statistical tests. Learning models of learning-control systems is also a difficult problem and we show in detail, how the accuracy of the estimator critically depends on the data properties.

To demonstrate the relevance and potential of ETL, we consider three instantiations of ETL algorithms, where we make all design choices precise and derive theoretical guarantees for the algorithms. Further, the algorithms are applied to experimental data and among others, show substantial communication savings for sensor networks.

In detail, we show how ETL can be used to save communication in networked control systems. Analyzing carefully chosen features such as inter-communication times is one possible approach that we develop in Part I. In addition to theoretical results, the algorithms are also applied to real-world data, where they outperform the considered baselines and reduce the communication load up to 70%.

The applications to optimal control tasks Part II show a similar potential. Here, we leverage in-depth knowledge of the corresponding distributions and structure, such as for the LQR cost signal. The theory is similarly rich and there are many potential real-world applications. In particular, in combination with reinforcement learning and other suitable optimal control strategies. As a first step, we have shown the power of the algorithms for balancing an inverted pendulum.

Most of the above ETL results are developed for linear systems. To generalize them to nonlinear dynamical systems, we introduce non-parametric testing techniques and explicitly deal with the correlations in the data. In Part III, we consider the core questions of comparing dynamical systems from state measurements. To this end, we introduce a new notion of mixing, which synergizes well with kernel two-sample tests and results in a statistical test for dynamical systems. In particular, the mixing properties can be estimated from data, which is a huge benefit over other methods and notions of mixing. We expect that these results will pave the way towards a general nonlinear event-triggered learning framework.

7.2 Future Work

We have developed the paradigm of event-triggered learning and derived concrete algorithms that are tailored to certain problems and show improvements in practice. Through this work, the fundamental question of "how to compare dynamical system" came up, which has far reaching consequences and points to interesting directions for future work.

As an answer to the fundamental question, we have developed a kernel two-sample test for dynamical systems. However, we did not address the question of relearning the dynamics. Combining the statistical information that the test yields with techniques from active learning appears to be a fruitful direction for future research. Particularly well suited approaches are deep learning and Gaussian processes, for example, adapting the work from Buisson-Fenet et al. (2020) to MMD-based arguments and integrating the kernel two-sample test into the learning framework. Causality-based approaches, such as proposed in Baumann et al. (2022), could also be fruitful.

It still remains to apply our kernel two-sample test to large scale problems and validate how well it performs on sophisticated engineering systems and if it generalizes to industrial applications. Detecting change for these systems is an important task with a huge potential impact in industry.

Simultaneously, the theoretical mixing results that emerged from the publication Solowjow et al. (2020) show promising connections to well-established theory. Extending these results and establishing more sophisticated concentration results is also a worthwhile endeavour. Especially, since our new notion of mixing can be estimated from data, which is a huge advantage over other methods.

In reinforcement learning, the exploration-exploitation trade-off plays a crucial role. This trade-off is also at the heart of ETL. Further, there is also the question of utilizing models or relying on model-free methods. Addressing these questions rigorously might lead to a similar principled decision making as we have explored in event-triggered learning.

While we have demonstrated that all components of the linear ETL framework work well in practice and theory, it would be exciting to push them to their limits and apply them to more sophisticated real-world systems with challenging baselines.

Quantifying the speed of change and embedding this into the ETL algorithms also remains an open question that should be addressed in the future. Most likely, there will be a substantial increase in performance.

7.3 Closing the Loop

Control theory has developed powerful tools over the last decades to deal with systems that have time dependencies. Feedback loops are at the heart of most of these algorithms and essentially, outputs are used as future inputs. When applying learning techniques to potentially changing environments similar techniques and ways of thinking can be applied.

Event-triggered learning is one way of closing the loop with the aid of statistical tests. We infer the accuracy of models and apply control inputs in form of learning only when necessary. In control theory, there is a strong emphasis on analyzing the whole system in addition to carefully designing the individual parts. This way, it is possible to derive strong theoretical guarantees, e.g., show types of stability. Similar concepts might be applicable for ETL systems in form of guaranteed adaptations to certain changes and stable or respectively successful learning outcomes.

Bibliography

A. Anderson, A. H. González, A. Ferramosca, A. D'Jorge, and E. Kofman. Robust MPC suitable for closed-loop re-identification, based on probabilistic invariant sets. *Systems & Control Letters*, 118:84–93, 2018.

B. D. Anderson. Failures of adaptive control theory and their resolution. *Communications in Information & Systems*, 05(1):1–20, 2005.

B. D. Anderson and J. B. Moore. *Optimal control: Linear quadratic methods.* Courier Corporation, 2007.

B. D. Anderson and J. B. Moore. *Optimal filtering.* Courier Corporation, 2012.

K. J. Åström. *Introduction to stochastic control theory.* Courier Corporation, 2012.

K. J. Åström and B. Wittenmark. *Adaptive control.* Courier Corporation, 2013.

G. J. Babu and C. R. Rao. Goodness-of-fit tests when parameters are estimated. *Sankhyā: The Indian Journal of Statistics*, 66(1):63–74, Feb. 2004.

E.-W. Bai and S. S. Sastry. Persistency of excitation, sufficient richness and parameter convergence in discrete time adaptive control. *Systems & Control Cetters*, 6 (3):153–163, 1985.

G. Battistelli, L. Chisci, and D. Selvi. A distributed kalman filter with event-triggered communication and guaranteed stability. *Automatica*, 93:75–82, 2018.

M. Bauer, A. Horch, L. Xie, M. Jelali, and N. Thornhill. The current state of control loop performance monitoring–a survey of application in industry. *Journal of Process Control*, 38:1–10, 2016.

D. Baumann, J.-J. Zhu, G. Martius, and S. Trimpe. Deep reinforcement learning for event-triggered control. In *Proceedings of the 57th IEEE International Conference on Decision and Control*, pages 943–950, Dec. 2018.

D. Baumann, F. Solowjow, K. H. Johansson, and S. Trimpe. Event-triggered pulse control with model learning (if necessary). In *Proceedings of the American Control Conference*, pages 792–797, July 2019.

D. Baumann, F. Solowjow, K. H. Johansson, and S. Trimpe. Identifying causal structure in dynamical systems. *Transactions on Machine Learning Research*, 2022.

A. Bemporad and E. Mosca. Fulfilling hard constraints in uncertain linear systems by reference managing. *Automatica*, 34(4):451–461, Apr. 1998.

J. Beuchert, F. Solowjow, J. Raisch, S. Trimpe, and T. Seel. Hierarchical event-triggered learning for cyclically excited systems with application to wireless sensor networks. *IEEE Control Systems Letters*, 4(1):103–108, Jan. 2020a.

J. Beuchert, F. Solowjow, S. Trimpe, and T. Seel. Overcoming bandwidth limitations in wireless sensor networks by exploitation of cyclic signal patterns: An event-triggered learning approach. *Sensors*, 20(1), Jan. 2020b. article number: 260.

S. S. Bidabadi, I. Murray, G. Y. F. Lee, S. Morris, and T. Tan. Classification of foot drop gait characteristic due to lumbar radiculopathy using machine learning algorithms. *Gait & posture*, 71:234–240, 2019.

H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen. Mean and variance of the LQG cost function. *Automatica*, 67:216 – 223, May 2016.

G. D. Birkhoff. Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660, 1931.

X. Bombois, M. Gevers, R. Hildebrand, and G. Solari. Optimal experiment design for open and closed-loop system identification. *Communications in Information and Systems*, 11(3):197–224, 2011.

F. Borrelli and T. Keviczky. Distributed LQR design for identical dynamically decoupled systems. *IEEE Transactions on Automatic Control*, 53(8):1901–1912, Sept. 2008.

R. C. Bradley. Basic properties of strong mixing conditions. a survey and some open questions. *Probability surveys*, 2:107–144, 2005.

R. C. Bradley, W. Bryc, and S. Janson. On dominations between measures of dependence. *Journal of multivariate analysis*, 23(2):312–329, 1987.

S. J. Bradtke, B. E. Ydstie, and A. G. Barto. Adaptive linear quadratic control using policy iteration. In *Proceedings of the American Control Conference*, volume 3, pages 3475–3479, 1994.

S. L. Brunton, B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz. Chaos as an intermittently forced linear system. *Nature communications*, 8(1):1–9, 2017.

M. Buisson-Fenet, F. Solowjow, and S. Trimpe. Actively learning gaussian process dynamics. In *Learning for dynamics and control*, pages 5–15. PMLR, 2020.

M. V. Butz. Towards strong ai. *KI-Künstliche Intelligenz*, 35(1):91–101, 2021.

M. V. Butz, O. Sigaud, and P. Gérard. Anticipatory behavior: Exploiting knowledge about the future to improve current behavior. In *Anticipatory behavior in adaptive learning systems*, pages 1–10. Springer, 2003a.

M. V. Butz, O. Sigaud, and P. Gérard. Internal models and anticipations in adaptive learning systems. In *Anticipatory behavior in adaptive learning systems*, pages 86–109. Springer, 2003b.

P. Caines and S. Lafortune. Adaptive control with recursive identification for stochastic linear systems. *IEEE Transactions on Automatic Control*, 29(4):312–321, 1984.

S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari. Approximating explicit model predictive control using constrained neural networks. In *Proceedings of the American Control Conference*, pages 1520–1527, 2018.

B.-E. Chérief-Abdellatif and P. Alquier. Finite sample properties of parametric mmd estimation: robustness to misspecification and dependence. *Bernoulli*, 28 (1):181–213, 2022.

H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23(4):493–507, Dec. 1952.

K. Chwialkowski and A. Gretton. A kernel independence test for random processes. In *International Conference on Machine Learning*, pages 1422–1430, 2014.

K. P. Chwialkowski, D. Sejdinovic, and A. Gretton. A wild bootstrap for degenerate kernel tests. In *Advances in Neural Information Processing Systems*, pages 3608–3616, 2014.

G. W. Corder and D. I. Foreman. *Nonparametric statistics: A step-by-step approach.* John Wiley & Sons, 2014.

R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart. The Mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, pages 1–47, 2019.

Y. Dong and S. J. Qin. A novel dynamic pca algorithm for dynamic data modeling and process monitoring. *Journal of Process Control*, 67:1–11, 2018.

G. Doran, K. Muandet, K. Zhang, and B. Schölkopf. A permutation-based kernel conditional independence test. In *UAI*, pages 132–141, 2014.

M. Eisen, K. Gatsis, G. J. Pappas, and A. Ribeiro. Learning in non-stationary wireless control systems via Newton's method. In *Proceedings of the American Control Conference*, pages 1410–1417, 2018.

N. M. Filatov and H. Unbehauen. *Adaptive dual control: Theory and applications*, volume 302. Springer Science & Business Media, 2004.

N. Funk, D. Baumann, V. Berenz, and S. Trimpe. Learning event-triggered control from data through joint optimization. *IFAC Journal of Systems and Control*, 16: 100144, 2021.

J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

E. Garcia and P. J. Antsaklis. Adaptive stabilization of uncertain systems with model-based control and event-triggered feedback updates. In *Control of Complex Systems*, pages 67–92. Elsevier, 2016.

H. Gaßner, D. Jensen, F. Marxreiter, A. Kletsch, S. Bohlen, R. Schubert, L. M. Muratori, B. Eskofier, J. Klucken, J. Winkler, et al. Gait variability as digital biomarker of disease severity in huntington's disease. *Journal of neurology*, pages 1–8, 2020.

K. Gatsis and G. J. Pappas. Sample complexity of networked control systems over unknown channels. In *Proceedings of IEEE Conference on Decision and Control*, pages 6067–6072. IEEE, 2018.

G. C. Goodwin and R. L. Payne. Chapter 7 recursive algorithms. In *Dynamic System Identification*, volume 136 of *Mathematics in Science and Engineering*, pages 175 – 208. Elsevier, 1977.

M. Green and J. B. Moore. Persistence of excitation in linear systems. *Systems & Control Letters*, 7(5):351 – 360, 1986.

A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. A kernel statistical test of independence. In *Advances in Neural Information Processing Systems*, pages 585–592, 2008.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012a.

A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems*, pages 1205–1213, 2012b.

A. Gut. *Probability: a graduate course*, volume 75. Springer, 2013.

D. Han, Y. Mo, J. Wu, S. Weerakkody, B. Sinopoli, and L. Shi. Stochastic event-triggered sensor schedule for remote state estimation. *IEEE Transactions on Automatic Control*, 60(10):2661–2675, 2015.

S. Han. A closed-form solution to the discrete-time Kalman filter and its applications. *Systems & Control Letters*, 59(12):799–805, dec 2010.

H. Hang, I. Steinwart, et al. A bernstein-type inequality for some mixing processes and dynamical systems with an application to learning. *The Annals of Statistics*, 45(2):708–743, 2017.

H. Hang, I. Steinwart, Y. Feng, and J. A. Suykens. Kernel density estimation for dynamical systems. *The Journal of Machine Learning Research*, 19(1):1260–1308, 2018.

T. J. Harris. Assessment of control loop performance. *The Canadian Journal of Chemical Engineering*, 67(5):856–861, 1989.

M. Heemels, K. H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *Proceedings of the 51st IEEE International Conference on Decision and Control*, pages 3270–3285. IEEE, 2012.

T. A. N. Heirung, B. E. Ydstie, and B. Foss. Towards dual MPC. *IFAC Proceedings Volumes*, 45(17):502–507, 2012. 4th IFAC Conference on Nonlinear Model Predictive Control.

J. P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proc. of IEEE* Special Issue on Technology of Networked Control Systems, 95(1):138–162, Jan. 2007.

J. Hodges. The significance probability of the smirnov two-sample test. *Arkiv för Matematik*, 3:469–486, 1958.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar. 1963.

Z.-S. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, June 2013.

B. Huang, S. Shah, and E. Kwok. Good, bad or optimal? Performance assessment of multivariable processes. *Automatica*, 33(6):1175–1183, 1997.

J. Huang, D. Shi, and T. Chen. Energy-based event-triggered state estimation for hidden markov models. *Automatica*, 79:256–264, 2017.

J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

P. A. Ioannou and J. Sun. *Robust adaptive control*. Courier Corporation, 2012.

R. Isermann. Process fault detection based on modeling and estimation methods—a survey. *Automatica*, 20(4):387–404, 1984.

R. Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2006.

I. Ishikawa, K. Fujii, M. Ikeda, Y. Hashimoto, and Y. Kawahara. Metric on nonlinear dynamical systems with Perron-Frobenius operators. In *Advances in Neural Information Processing Systems*, pages 2856–2866, 2018.

I. Ishikawa, A. Tanaka, M. Ikeda, and Y. Kawahara. Metric on random dynamical systems with vector-valued reproducing kernel Hilbert spaces. *arXiv preprint arXiv:1906.06957*, 2019.

P. M. J. Apkarian, M. Levis. *QUBE-Servo 2 Experiment for Matlab/Simulink Users: Instructor Workbook*. Quanser Inc, 2016. URL https://www.quanser.com/products/qube-servo-2/.

S. Jegelka, A. Gretton, B. Schölkopf, B. K. Sriperumbudur, and U. Von Luxburg. Generalized clustering via kernel embeddings. In *Annual Conference on Artificial Intelligence*, pages 144–152. Springer, 2009.

M. Jelali. An overview of control performance assessment technology and industrial applications. *Control engineering practice*, 14(5):441–466, 2006.

S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert. Industrial Internet of Things and cyber manufacturing systems. In *Industrial Internet of Things*, pages 3–19. Springer, 2017.

R. M. Johnstone, C. Richard Johnson, R. R. Bitmead, and B. D. Anderson. Exponential convergence of recursive least squares with exponential forgetting factor. *Systems & Control Letters*, 2(2):77–82, 1982.

R. H. Julien, M. W. Foley, and W. R. Cluett. Performance assessment using a model predictive control benchmark. *Journal of Process Control*, 14(4):441–456, 2004.

M. Kac, J. Kiefer, and J. Wolfowitz. On tests of normality and other tests of goodness of fit based on distance methods. *The Annals of Mathematical Statistics*, pages 189–211, June 1955.

R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *International Conference on Machine Learning*, pages 487–494, 2000.

P. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2011.

S. Klus, I. Schuster, and K. Muandet. Eigendecompositions of transfer operators in reproducing kernel Hilbert spaces. *Journal of Nonlinear Science*, 30(1):283–315, 2020.

K. Krauth, S. Tu, and B. Recht. Finite-time analysis of approximate policy iteration for the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pages 8512–8522, 2019.

P. R. Kumar and P. Varaiya. *Stochastic systems: Estimation, identification and adaptive control*. Prentice Hall, 1986.

B. Kuvaritakis and M. Cannon. *Model Preditive Control: Classical, Robust and Stochastic*. Springer, 2014.

D. Lee and J. Hu. Primal-dual Q-learning framework for LQR design. *IEEE Transactions on Automatic Control*, 64(9):3756–3763, 2018.

J. Lee, J.-S. Kim, and H. Shim. Disc margins of the discrete-time LQR and its application to consensus problem. *International Journal of Systems Science*, 43 (10):1891–1900, Oct. 2012.

M. Lemmon. Event-triggered feedback in control, estimation, and optimization. *Networked Control Systems*, 406:293–358, 2010.

L. Ljung. *System Identification: Theory for the user.* Information and System Sciences Series. Prentice Hall, 2 edition, 2009.

L. Ljung and T. Söderström. *Theory and practice of recursive identification.* MIT press, 1983.

J. R. Lloyd and Z. Ghahramani. Statistical model criticism using kernel two sample tests. In *Advances in Neural Information Processing Systems*, pages 829–837, 2015.

M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2208–2217. JMLR. org, 2017.

D. Looze, J. Weiss, J. Eterno, and N. Barrett. An automatic redesign approach for restructurable control systems. *IEEE Control systems magazine*, 5(2):16–22, May 1985.

J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363, 2018.

M. J. Lueken, B. J. Misgeld, and S. Leonhardt. Classification of spasticity affected emg-signals. In *International Conference on Wearable and Implantable Body Sensor Networks*, pages 1–6. IEEE, 2015.

S. Luzzatto, I. Melbourne, and F. Paccaut. The Lorenz attractor is mixing. *Communications in Mathematical Physics*, 260(2):393–401, 2005.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics.* Wiley, 1999.

H. Mania, S. Tu, and B. Recht. Certainty equivalence is efficient for linear quadratic control. In *Advances in Neural Information Processing Systems*, pages 10154–10164, 2019.

A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. Automatic LQR tuning based on Gaussian process global optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 270–277, 2016.

S. Mason, L. Righetti, and S. Schaal. Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 374–379, 2014.

P. Massart. The tight constant in the dvoretzky-kiefer-wolfowitz inequality. *The annals of Probability*, pages 1269–1283, 1990.

P. Matisko and V. Havlena. Optimality tests and adaptive Kalman filter. *IFAC Proceedings Volumes*, 45(16):1523–1528, jul 2012.

D. McDonald, C. Shalizi, and M. Schervish. Estimating beta-mixing coefficients. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 516–524, 2011.

I. Mezic. On comparison of dynamics of dissipative and finite-time systems using Koopman operator methods. *IFAC-PapersOnLine*, 49(18):454–461, 2016.

B. J. Misgeld, M. Lüken, D. Heitzmann, S. I. Wolf, and S. Leonhardt. Body-sensor-network-based spasticity detection. *IEEE journal of biomedical and health informatics*, 20(3):748–755, 2015.

M. Miskowicz. Send-on-delta concept: an event-based data reporting strategy. *Sensors*, 6(1):49–63, 2006.

M. Miskowicz. *Event-based control and signal processing*. CRC Press, 2015.

K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

A. Muro-De-La-Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla. Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. *Sensors*, 14(2):3362–3394, 2014.

V. Narayanan and S. Jagannathan. Event-triggered distributed control of nonlinear interconnected systems using online reinforcement learning with exploration. *IEEE transactions on cybernetics*, 48:2510–2519, 2017.

A. Nguyen, N. Roth, N. Ghassemi, J. Hannink, T. Seel, J. Klucken, H. Gaßner, and B. Eskofier. Development and clinical validation of inertial sensor-based gait-clustering methods in Parkinson's disease. *Journal of NeuroEngineering and Rehabilitation*, 16(77):1–14, 2019.

J. Nubert, J. Koehler, V. Berenz, F. Allgower, and S. Trimpe. Safe and fast tracking on a robot manipulator: Robust MPC and neural network control. *IEEE Robotics and Automation Letters*, 2020.

B. Øksendal. Stochastic differential equations. In *Stochastic differential equations*. Springer, 2003.

R. Oldenburger. Infinite powers of matrices and characteristic roots. *Duke Mathematical Journal*, 6(2):357–361, Feb. 1940.

P. Patie and C. Winter. First exit time probability for multidimensional diffusions: a pde-based approach. *Journal of computational and applied mathematics*, 222 (1):42–53, 2008.

G. Pavliotis. *Stochastic Processes and Applications Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer, 2014.

N. Pfister, P. Bühlmann, B. Schölkopf, and J. Peters. Kernel-based tests for joint independence. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):5–31, 2018.

M. Pistacchi, M. Gioulis, F. Sanson, E. De Giovannini, G. Filippi, F. Rossetto, and S. Z. Marsala. Gait analysis and clinical correlations in early Parkinson's disease. *Functional neurology*, 32(1):28, 2017.

L. Pronzato. Optimal experimental design and some related control problems. *Automatica*, 44(2):303–325, 2008.

S. J. Qin. Control performance monitoring – a review and assessment. *Computers & Chemical Engineering*, 23(2):173–186, 1998.

S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

M. Rabi, K. H. Johansson, and M. Johansson. Optimal stopping for event-triggered sensing and actuation. In *2008 47th IEEE Conference on Decision and Control*, pages 3607–3612, 2008.

B. Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:253–279, 2019.

S. Schlor, F. Solowjow, and S. Trimpe. Parameter filter-based event-triggered learning. *under review*, 2022.

H. Schluter, F. Solowjow, and S. Trimpe. Event-triggered learning for linear quadratic control. *IEEE Transactions on Automatic Control*, 2020.

B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

L. Schwenkel, M. Gharbi, S. Trimpe, and C. Ebenbauer. Online learning with stability guarantees: A memory-based warm starting for real-time mpc. *Automatica*, 122:109247, 2020.

D. Sejdinovic, B. Sriperumbudur, A. Gretton, and K. Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, pages 2263–2291, 2013.

D. Shi, T. Chen, and L. Shi. Event-triggered maximum likelihood state estimation. *Automatica*, 50(1):247–254, 2014.

D. Shi, L. Shi, and T. Chen. *Event-Based State Estimation: A Stochastic Perspective*. Springer, 2015.

J. Sijs and M. Lazar. Event based state estimation with time synchronous updates. *IEEE Transactions on Automatic Control*, 57(10):2650–2655, 2012.

J. Sijs, L. Kester, and B. Noack. A study on event triggering criteria for estimation. In *17th International Conference on Information Fusion*, pages 1–8, July 2014.

M. Simchowitz, H. Mania, S. Tu, M. I. Jordan, and B. Recht. Learning without mixing: Towards a sharp analysis of linear system identification. *Conference on Learning Theory*, pages 439–473, 2018.

M. Simchowitz, R. Boczar, and B. Recht. Learning linear dynamical systems with semi-parametric least squares. In *Conference on Learning Theory*, pages 2714–2802. PMLR, 2019.

C.-J. Simon-Gabriel, A. Barp, and L. Mackey. Metrizing weak convergence with maximum mean discrepancies. *arXiv preprint arXiv:2006.09268*, 2020.

F. Solowjow and S. Trimpe. Event-triggered learning. *Automatica*, 117:109009, 2020.

F. Solowjow, D. Baumann, J. Garcke, and S. Trimpe. Event-triggered learning for resource-efficient networked control. In *2018 Annual American Control Conference (ACC)*, pages 6506–6512. IEEE, 2018.

F. Solowjow, D. Baumann, C. Fiedler, A. Jocham, T. Seel, and S. Trimpe. A kernel two-sample test for dynamical systems. *arXiv preprint arXiv:2004.11098*, 2020.

M. J. Sorocky, S. Zhou, and A. P. Schoellig. Experience selection using dynamics similarity for efficient multi-source transfer learning between robots. In *Proc. of the IEEE International Conference on Robotics and Automation* , 2020.

J. C. Spall. Validation of state space models in non-gaussian systems. In *1984 American Control Conference*, jul 1984.

J. C. Spall and K. D. Wall. Asymptotic distribution theory for the Kalman filter state estimator. *Communications in Statistics - Theory and Methods*, 13(16): 1981–2003, jan 1984.

B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:1517–1561, 2010.

I. Steinwart and A. Christmann. *Support vector machines.* Springer Science & Business Media, 2008.

Y. S. Suh. Send-on-delta sensor data transmission with a linear predictor. *Sensors*, 7(4):537–547, 2007.

A. P. Swanda and D. E. Seborg. Controller performance assessment based on set-point response data. In *Proceedings of the American Control Conference*, volume 6, pages 3863–3867, 1999.

Z. Szabó and B. Sriperumbudur. Characteristic and universal tensor product kernels. *Journal of Machine Learning Research*, 18:233, 2018.

I. Tien, S. D. Glaser, and M. J. Aminoff. Characterization of gait abnormalities in Parkinson's disease using a wireless inertial sensor system. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 3353–3356. IEEE, 2010.

E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the American Control Conference*, pages 300–306, 2005.

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schölkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations (ICLR 2018)*. OpenReview. net, 2018.

S. Trimpe. Event-based state estimation: An emulation-based approach. *IET Control Theory & Applications*, 11(11):1684–1693, July 2017.

S. Trimpe and D. Baumann. Resource-aware IoT control: Saving communication through predictive triggering. *IEEE Internet of Things Journal*, 2019.

S. Trimpe and M. C. Campi. On the choice of the event trigger in event-based estimation. In *Int. Conf. on Event-based Control, Communication, and Signal Processing*, pages 1–8, 2015.

S. Trimpe and R. D'Andrea. An experimental demonstration of a distributed and event-based state estimation algorithm. *IFAC Proceedings Volumes*, 44(1):8811–8818, 2011.

S. Trimpe and R. D'Andrea. The balancing cube: A dynamic sculpture as test bed for distributed estimation and control. *IEEE Control Systems Magazine*, 32(6): 48–75, Dec. 2012.

S. Trimpe and R. D'Andrea. Event-based state estimation with variance-based triggering. *IEEE Transactions on Automatic Control*, 59(12):3266–3281, 2014.

J. Umlauft and S. Hirche. Feedback linearization based on Gaussian processes with event-triggered online learning. *IEEE Transactions on Automatic Control*, 2019.

H. Unbehauen. Adaptive dual control systems: a survey. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 171–180. IEEE, 2000.

K. G. Vamvoudakis and H. Ferraz. Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. *Automatica*, 87: 412–420, 2018.

K. G. Vamvoudakis, F. R. Pour Safaei, and J. P. Hespanha. Robust event-triggered output feedback learning algorithm for voltage source inverters with unknown load and parameter variations. *International Journal of Robust and Nonlinear Control*, 29(11):3502–3517, 2019.

S. Vishwanathan, A. J. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73(1):95–119, 2007.

M. Witczak, V. Puig, D. Rotondo, and P. Witczak. A necessary and sufficient condition for total observability of discrete-time linear time-varying systems. *IFAC-PapersOnLine*, 50(1):729–734, 2017.

B. Wittenmark. An active suboptimal dual controller for systems with stochastic parameters. *Automatic Control Theory and Application*, 3:13–19, 1975.

J. Wu, Q.-S. Jia, K. H. Johansson, and L. Shi. Event-based sensor data scheduling: Trade-off between communication rate and estimation quality. *IEEE Transactions on Automatic Control*, 58(4):1041–1046, 2013.

Y. Xu and J. P. Hespanha. Communication logics for networked control systems. In *Proceedings of the American Control Conference*, pages 572–577, 2004.

J. Yoo and K. H. Johansson. Event-triggered model predictive control with a statistical learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–11, 2019.

W. Zaremba, A. Gretton, and M. Blaschko. B-test: A non-parametric, low variance kernel two-sample test. In *Advances in Neural Information Processing Systems*, pages 755–763, 2013.

Y. Zhan and J. Jiang. An interacting multiple-model based fault detection, diagnosis and fault-tolerant control approach. In *Proceedings of IEEE Conference on Decision and Control*, volume 4, pages 3593–3598, 1999.

K. Zhou and J. C. Doyle. *Essentials of robust control.* Prentice hall, 1998.

Appendix A

Details on the MMD-Experiments

In this appendix, we discuss further details about the experiments presented in Part III.

A.1 LTI Systems

We consider the dynamics

$$X_{k+1} = AX_k + \epsilon_k, \tag{A.1}$$

where $\epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \Sigma)$. Further, assume all eigenvalues of $A \in \mathbb{R}^{d \times d}$ are located within the unit circle.

Stationary Distribution: The stationary distribution of an LTI system is Gaussian with expected value zero. The Gaussian distribution follows from the Gaussian noise and linear structure of the system. The expected value can be computed by leveraging that all eigenvalues of A are located within the unit circle. Obtaining the variance is more involved. It can be expressed as the solution to the following Lyapunov equation in Z (Schluter et al., 2020, Eq. 7):

$$AZA^{\intercal} - Z + \Sigma = 0, \tag{A.2}$$

where A is the system matrix and Σ the covariance matrix of the process noise.

A.1.1 Comparison to Stationary

We investigate if we can, based on a kernel two-sample test, distinguish between time shifted samples with respect to a^* and i.i.d. samples from the stationary distribution.

Setup: We create 500 randomly generated LTI systems with a random dimensionality between 1 and 100. For each system we create $m = 250$ independent trajectories and sample $n = 20\,000$ points for each trajectory. All systems are initialized in $X_0 = 0$ to avoid transient effects. The decay of dependence is quantified

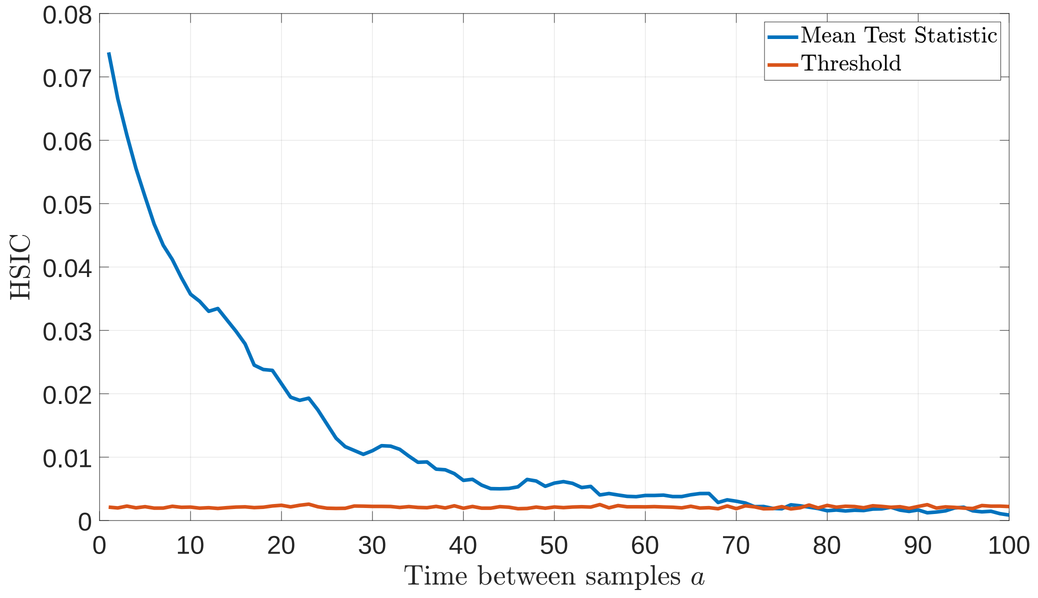


Figure A.1: Mixing properties of the LTI system that is used to create Fig. 1 in the main paper. On the x-axis, we depict the time shift a between consecutive samples. The y-axis shows the dependence between data points with respect to the corresponding time shift. At $a^* = 75$ the test statistic is below the threshold.

in the MMD-sense for one gap (cf. Sec. 5 of main paper). We use data from the end of the trajectory to avoid numerical artifacts due to the identical initial values.

Next, we describe how we generate the system matrices.

Sampling Σ: The entries for the covariance matrix are drawn from a standard multivariate normal distribution. Since the matrix is supposed to yield a covariance matrix, we require symmetry and positive definiteness. Thus, we denote Σ' as the matrix drawn from the normal distribution and define $\Sigma = 0.5(\Sigma' + \Sigma'^{\intercal})^2$. To control the magnitude of noise, we scale the matrix with the largest eigenvalue of Σ.

Sampling A: The system matrix A is required to have eigenvalues within the unit sphere. To achieve this, we draw the entries of A from a uniform distribution and extend the system with a control input

$$X_{k+1} = AX_k + Bu_k + \epsilon_k. \tag{A.3}$$

The control matrix B is set to the identity matrix and the control input as a standard linear quadratic feedback controller $u_k = -Kx_k$. This yields the closed loop dynamics

$$X_{k+1} = (A - BK)X_k + \epsilon_k. \tag{A.4}$$

The feedback gain K can be computed to minimize a linear quadratic cost function. By adjusting the weights of the cost function, we can indirectly adjust the eigenvalues of the closed loop system matrix $(A - BK)$. We set the weight matrix for the state cost Q to the identity matrix and the control cost to $R = 10^7$. This makes it very expensive to apply large control inputs and magnitude of the eigenvalues of $(A - BK)$ stays close to 1. This implies slow mixing and further, by considering $R \to \infty$, we can make this arbitrarily slow.

Results: First, we use the $m = 250$ trajectories to estimate the mixing speed a^*. We choose a^* as the first time instance at which the test statistic is below the test threshold.

For the kernel two-sample test, we draw 100 points from the first trajectory that respect the time shift a^*. We also draw 100 points directly from the stationary distribution ($\mathcal{N}(0, Z)$, cf. (A.2)).

From the 500 systems we considered overall, we only obtained 4 false positives, which shows the high precision of our proposed test. Due to the probabilistic nature of these experiments, we could obtain systems with arbitrarily slow mixing times and, subsequently, very long a^*. Thus, we decided to fix a maximum a^* as $a_{\max} = 200$, and ignore all systems with larger a^*. We obtained 81 systems that mix too slowly, i.e., $a^* > a_{\max}$.

A.1.2 Details for Fig. 6.1

In Fig. A.1, we show the mixing properties of the system that yields $a^* = 75$. We used the same setup as in Sec. A.1.1 with some modifications. We chose $R = 10^{10}$ and divided Σ by $10\lambda_{\max}^2$, where $\lambda_{\max}$ is the largest eigenvalue of Σ. Further, to be able to better visualize the samples and the stationary distributions, we fixed the dimension to two.

The randomly generated system matrices are

$$A = \begin{pmatrix} 0.2345 & 0.8609 \\ 0.7298 & 0.1316 \end{pmatrix}, \Sigma = \begin{pmatrix} 0.0378 & 0.0135 \\ 0.0135 & 0.0971 \end{pmatrix}. \tag{A.5}$$

A.2 Lorenz System

To perform kernel two-sample testing, we slightly change the parameters in the Lorenz system by decreasing the coefficient in (6.32) from 10 to 6 to obtain a second slightly different system. The mixing analysis is done for both systems. The attractors of both systems look optically very similar. The attractor can be interpreted as the stationary probability distribution of the state in some sense.

A.2.1 Mixing properties

We estimate the mixing properties of the Lorenz system in the MMD-mixing sense for one time shift a (cf. Sec.5).

Initial points: We sample from an uniform distribution $\mathcal{U}([-0.5, 0.5] \times [-0.5, 0.5] \times [20, 21])$ to initialize the starting point X_0.

Data: We use a standard ODE solver[1] to obtain a solution to the Lorenz system. Due to variable step sizes within the solver, we interpolate the solution to obtain samples with a fixed discretization in time. We consider the time horizon $t \in [0, 200]$ and create 2001 samples (with a fixed time step of 0.1).

Repetitions: We create $M = 100$ independent trajectories to estimate the mixing properties. The experiment is repeated $N = 100$ times to investigate deviations in the decay of the dependence.

Estimating mixing: To avoid numerical artifacts due to the initial points and potential transients, we consider data from the end of the trajectory. Thus, we sample at $t_{\text{end}} = 200$ and at $t - a$ for various values of $a = 0.1, 1.1, 2.1, \dots, 99.1$ with respect to the continuous time index t.

Results: We depict the decay of dependence in Fig. A.2. After waiting for $a^* = 20$, the dependence in the data is not detectable anymore. Since the decay is not necessarily monotonic, we consider significantly higher time shifts up to $a = 99.1$. The dependence does not increase again, which indicates that the system is mostly mixing in the MMD-sense. Of course, this does not prove that the Lorenz system mixes and it remains to be shown rigorously. Nonetheless, these results are promising and provide empirical evidence.

A.2.2 Kernel Two-sample Test

We try to distinguish between the Lorenz system given in (6.32) – (6.34) and a slightly disturbed system where we change the parameter in (6.32) from 10 to 6. Based on the previous mixing analysis (cf. Fig. A.2) we set the time shift $a^* = 20$. This yields approximately independent samples for both systems.

We create two trajectories of length $t_{\max}$ and pick n points that respect the time shift a^* as illustrated in Fig. 6.2. We repeat all experiments 100 times. We start the sampling after $t = 20$, which gives the system enough time to converge to the stationary distribution.

Accuracy: We use $t_{\max} = 6000$ and pick $n = 300$ points from both system. We achieve 95% accuracy in detecting different systems.

False positives: We consider two trajectories that were generated by the classical Lorenz system ((6.32) – (6.34). The initial points for both trajectories were random and different. This setup yields 2.67% false positives, which is less than the α-level of 5% that we used.

[1]ode45 in Matlab

Next, we investigate what happens if we violate a^*. We choose $n = 100$ and $t_{\max} = 30$. Thus, we sample 100 points in the time interval $t \in [20, 30]$. This clearly violates the estimated a^* and indeed, we obtain 51% false positives. Essentially, this makes the test useless when a^* is severely violated and thus, we want to emphasize again that it is critical to estimate a^*. Further, through an appropriate choice of a^* we inherit all the rich theoretical properties of kernel two-sample testing.

A.3 Non-mixing System

We construct a system that does not mix in the MMD sense and is also not expected to mix. However, the system is well known to be ergodic and stationary. In particular, we consider a dynamical system that moves on a circle with a radius of one and steps of length $\frac{\pi}{10}$. We create $m = 100$ randomly initialized points θ_0 and iterate them for $n = 100$ timesteps with the dynamics following

$$\theta_{k+1} = \theta_k + \frac{\pi}{10}, \tag{A.6}$$

and

$$X_{k+1} = \begin{pmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{pmatrix} \tag{A.7}$$

We show the mixing properties in Fig. A.3. The dependence between data points stays constant and does not decrease and we detect this. Thus, we correctly identify systems that are not mixing in the MMD sense.

We have also tried different increments instead of $\frac{\pi}{10}$, such as $\frac{e}{10}$ and also $\frac{1}{10}$, which all resulted in the same outcome.

A.4 Implementations

Since our method is leveraging results from standard kernel two-sample testing and the HSIC, we directly used existing implementations without modifying them.

Kernel Two-sample Test Implementation: We used the Matlab implementation: http://www.gatsby.ucl.ac.uk/~gretton/mmd/mmd.htm and the standard hyperparameters without any tuning. We used the significance level $\alpha = 0.05$ for all experiments.

HSIC Implementation: We used the Matlab implementation: http://people.kyb.tuebingen.mpg.de/arthur/indep.htm with standard hyperparameters and $\alpha = 0.05$ for all experiments.

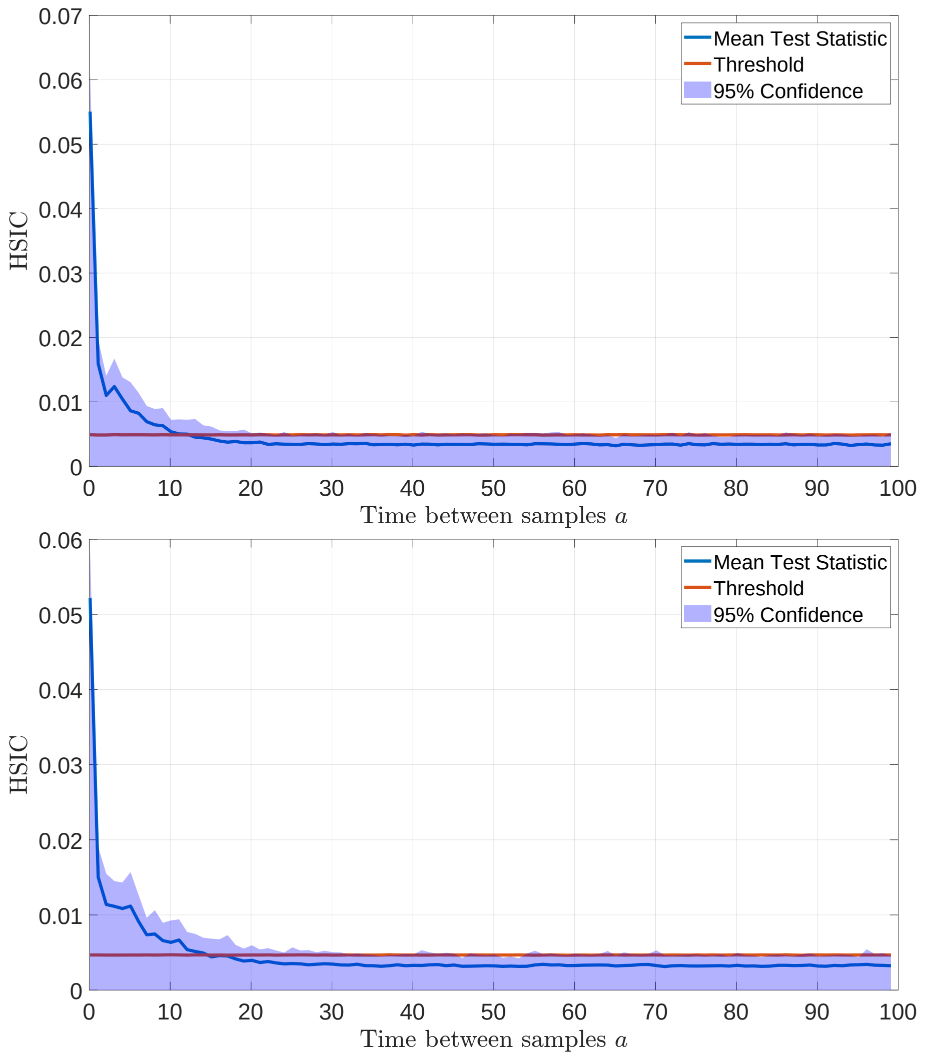


Figure A.2: Mixing properties of the Lorenz system. In the top plot with parameters as in (6.32)–(6.34) and in the bottom, we adapted the parameter in (6.32) to 6. On the x-axis, we depict the time shift a between consecutive samples. The y-axis shows the dependence between data points with respect to the corresponding time shift. The initial point is randomized, and the estimation is repeated 100 times. Depicted is the mean of the test statistic and the 95% upper confidence bound. We also show the threshold κ of the independence test. When the blue line is below the red line, it is not possible to infer statistical dependence between the data points.

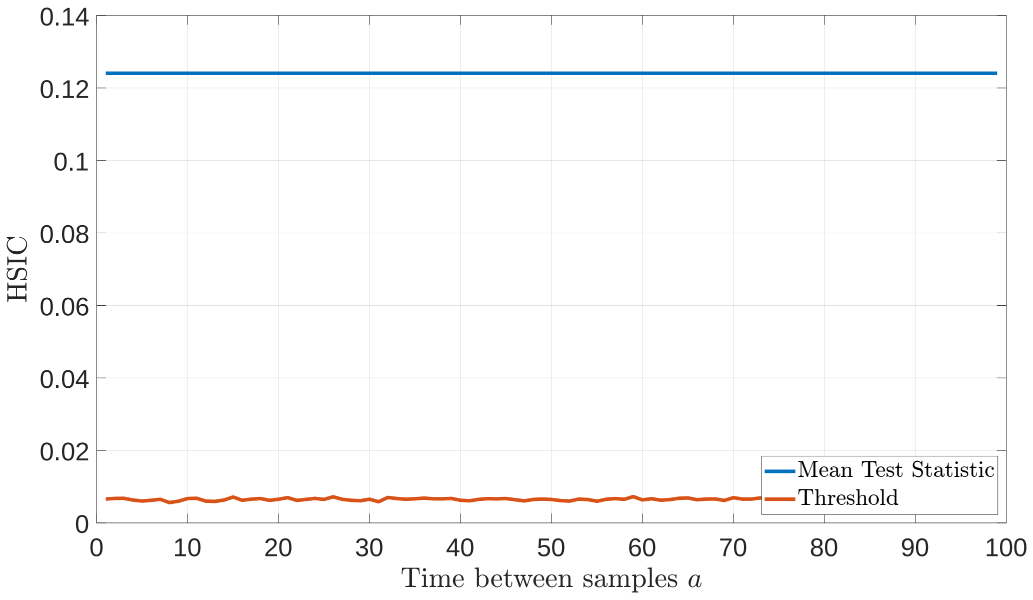


Figure A.3: Mixing properties of a dynamical system that moves on a circle. The dependency between data points does not decrease and stays above the threshold.