Event-triggered Learning for Linear Quadratic Control

Henning Schlüter\textsuperscript{1,2,*}, Friedrich Solowjow\textsuperscript{1,*}, Sebastian Trimpe\textsuperscript{1}

Abstract—When models are inaccurate, performance of model-based control will degrade. For linear quadratic control, an event-triggered learning framework is proposed that automatically detects inaccurate models and triggers learning of a new process model when needed. This is achieved by analyzing the probability distribution of the linear quadratic cost and designing a learning trigger that leverages Chernoff bounds. In particular, whenever empirically observed cost signals are located outside the derived confidence intervals, we can provably guarantee that this is with high probability due to a model mismatch. With the aid of numerical and hardware experiments, we demonstrate that the proposed bounds are tight and that the event-triggered learning algorithm effectively distinguishes between inaccurate models and probabilistic effects such as process noise. Thus, a structured approach is obtained that decides when model learning is beneficial.

Index Terms—Event-triggered Learning; Optimal Control; Statistical Learning; Stochastic Systems.

I. INTRODUCTION

LINEAR quadratic regulator (LQR) problems are well understood in literature and yield tractable and well-behaved solutions (see for example [1], [2] 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 [3]. While LQR has favorable robustness properties [4], performance of the controller naturally depends on the accuracy of the underlying model. Thus, just as any other model-based design, LQR will generally benefit from a precise model, both in terms of performance and robustness.

As the accuracy of the utilized model is decisive for control performance, we propose to improve the model during operation from data when needed. Clearly, the idea of data driven model updates is not new [5], 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 [6]. 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 behavior of the system and investigate the question of when to propose to separate the process of learning from the nominal from divergence issues when the system is standing still and can be wasteful from a resource point of view and may suffer on when to learn is a novel approach. Learning permanently updates is not new [5], however, principled decision making from data when needed, was first proposed in [7], [8] 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.

and optimal control approaches by performing learning in a controlled environment and afterward, applying the rich optimal control framework to learned models. However, the key 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. 1.

Related Work

The idea of event-triggered learning, i.e., triggering model learning when needed, was first proposed in [7], [8] to achieve communication savings in networked control systems, where model-based predictions are used to anticipate other agents’ behavior and thus, avoiding continuous communication. Recently, the idea of event-triggered learning has been applied to event-triggered pulse control [9] and to track human running gaits in sensor networks [10], where it yields significant communication savings. In all of the above articles, the times between two communication instances are analyzed and treated as random variables. By deriving a model-induced probability distribution and simultaneously testing how likely it is that observed inter-communication times are actually drawn from this probability distribution, it is possible to construct learning triggers. Further, statistical guarantees can be deduced based on concentration inequalities such as Hoeffding’s inequality. In this article, we build on the main idea of ETL, but we develop learning triggers for a very different setting: models...
will be used for control design (instead of predictions), and triggers will be based on control performance (rather than communication). This is the first work to develop the idea of ETL for control. Concretely and to the best of our knowledge, this is the first approach that online monitors the performance of an LQR and triggers model learning when needed to improve performance.

The LQR framework is a popular control method with many recent applications (see for instance [11]–[14]). Usually, the expected value of the cost function is minimized, however, there are also approaches that consider variance such as minimum variance control (see [1] for details). Closed form solutions for the variance of the cost were derived very recently in [15].

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 cost.

Adaptive control (see [6], [16], [17] 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, convergence of such algorithms is usually tightly connected to persistently excited signal vectors [17], which is not necessarily satisfied. Further, it is well known in statistical literature that simultaneous parameter estimation and testing might lead to distorted test statistics and different asymptotic distributions [18] of statistical tests. There exist statistical tests that explicitly take the distribution of the estimator into account [19], 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.

Robust control (cf. [20] 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 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.

The key contribution of this article lies in designing dedicated learning triggers that compare empirical costs to a model-induced distribution. Change detecting and fault detection [21]–[24] have already been addressed in literature and there are many methods that can be applied to the consider problem. However, the herein 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 generation function of the cost.

Contributions

To summarize, this article makes the following contributions:

- Introducing the concept of control performance based event-triggered learning for linear quadratic control – 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 (cf. Thm. 3);
- Design of two distinct learning triggers based on the first moment (Sec. III) and the full distribution (Sec. IV) of the LQR cost with additional theoretical guarantees, which are derived with the aid of concentration inequalities;
- Validation and comparison of the derived triggers in simulation and hardware experiments, in which we demonstrate fast, reliable, and robust detection.

When developing the learning triggers, we obtain mean and variance of the quadratic costs as side results (Lemmata 1 and 4). These are the discrete-time analogues to the continuous-time results in [15], and thus potentially relevant also beyond their use for ETL.

Notation

The maximum eigenvalue of a matrix \( Q \) is denoted \( \lambda_{\max}(Q) \). The Kronecker delta \( \delta_{ij} \) is equal to one if \( i = j \) and zero otherwise. We use \( \mathbb{P}[s \in S], \mathbb{P}[s \in \mathbb{S}, t \in T] \), and the notation \( \mathbb{P}[s \in S|t \in T] \) for the probability of an event \( s \in S \), the joint probability of the events \( s \in \mathbb{S} \) and \( t \in T \), and the conditional probability of event \( s \in \mathbb{S} \) given \( t \in T \). Further, we denote the respective probability density functions of the random variables \( s \) and \( t \) with \( \mathbb{P}[s], \mathbb{P}[s, t], \) and \( \mathbb{P}[s|t] \). The expected value of a random variable \( s \) is \( \mathbb{E}[s] \).

II. 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 of this article.

A. Problem setup

We assume the linear dynamics

\[
x_{k+1} = A_n x_k + B u_k + v_k,
\]

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 \( A_n \in \mathbb{R}^{n \times n} \), input matrix \( B \in \mathbb{R}^{n \times q} \).
and independent identically distributed (i.i.d.) Gaussian noise \( v_k \sim \mathcal{N}(0, V) \) with \( \mathbb{E}[v_i v_j^\top] = V \delta_{ij} \). Further, the system is assumed to be \((A_o, B)\)-stabilizable. Hence, stable closed-loop dynamics can be achieved through state feedback

\[
u_k = -F x_k + u_{\text{ref}}(k),
\]

where \( F \in \mathbb{R}^{q \times n} \) is the feedback gain and \( u_{\text{ref}}(k) \) 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 [2]. In particular, we can use Riccati equations to find analytical solutions to the optimal control problem with the quadratic cost function

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

where \( Q_{\text{LQR}} \) and \( R_{\text{LQR}} \) 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

\[
\hat{J}_N(k) = \sum_{j=k-N+1}^{k} x_j^\top Q_{\text{LQR}} x_j + u_j^\top R_{\text{LQR}} u_j.
\]

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

\[
x_{k+1} = A x_k + v_k,
\]

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

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

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

It is well-known that the states of a stable system (such as (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., [25, Lemma 2.1])

\[
A X^V A^\top - X^V + V = 0.
\]

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 1: The closed-loop model (5) is stable in the sense that \( |\lambda_{\text{max}}(A)| < 1 \).

This assumption is not very restrictive, as we only require the feedback law (2) to stabilize the open-loop model (1).

Assumption 2: 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 1, it follows directly that the system converges exponentially to a steady-state Gaussian distribution [26, 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 \( J_N(k) \), 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 2. 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 \( k \).

B. Problem and Main Idea

In this work, we systematically analyze the question of when to learn a new model of the dynamical system (1), which is later on utilized to synthesis 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 in 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. 1. The core piece of the proposed method is the binary event trigger \( \gamma_{\text{learn}} \) for learning a new model and the corresponding test statistic \( \psi \) 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 \( \eta \), we are able to compute critical thresholds \( \kappa \) 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 \( \psi \). Identifying linear systems is not the focus of this article and has been extensively discussed in previous work (see [27] 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 in this article.

Problem 1: Detect, when the model has changed, by comparing the deviation of model-induced cost properties to empirical costs, thus, yielding the learning trigger

\[
\psi(\hat{J}_N, J_N) > \kappa \Leftrightarrow \gamma_{\text{learn}} = 1,
\]

where \( \psi \) is an appropriate test statistic, \( \kappa \) is the computed critical threshold and \( \gamma_{\text{learn}} \) 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 \( \kappa \) such that

\[
\mathbb{P}[\psi(\hat{J}_N, J_N) > \kappa] < \eta,
\]

i.e., the probability of the trigger misfiring is less then desired confidence level \( \eta \).
In a first approach, we will develop learning triggers based on the expected value of the cost (Sec. III), which allows for a straightforward implementation to detect system changes. Then we show how this approach can be improved by incorporating the entire distribution in form of a moment generating function in Sec. IV. This yields a trigger with superior theoretical properties as well as better empirical performance and reliability.

III. MEAN-BASED LEARNING TRIGGER

In this section, we will derive a learning trigger, which is based on the moving average of the cost function. The idea is to derive a threshold $\kappa$ on the deviation from the expected value $\mathbb{E}[J_N]$, leading to

$$\sum_{j \in \mathcal{L}(k)} \hat{J}_N(j) - L \mathbb{E}[J_N] \geq \kappa \Leftrightarrow \gamma_{\text{learn}} = 1,$$  \hspace{1cm} (10)

where $\mathcal{L}(k)$ is a summation set of cardinality $L$ that achieves approximately uncorrelated samples and will be discussed later (cf. (14)). We will first provide a derivation of the expected value of the cost $\mathbb{E}[J_N]$, leading to

$$\mathbb{E}[J_N] = \mathbb{E} \left[ \sum_{k=0}^{N-1} x_k^T Q x_k \right] = \text{trace} \left( \sum_{k=0}^{N-1} S_k Q \right) = \text{trace} \left( N V^T X^Q \right).$$  \hspace{1cm} (11)

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

Proof: We first note that $\mathbb{E}[J_N] = \mathbb{E} \left[ \sum_{k=0}^{N-1} x_k^T Q x_k \right] = \mathbb{E} \left[ \sum_{k=0}^{N-1} \text{trace} (x_k x_k^T Q) \right] = \text{trace} \left( \sum_{k=0}^{N-1} \mathbb{E}[x_k x_k^T] Q \right) = \text{trace} \left( \sum_{k=0}^{N-1} S_k Q \right)$.

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

$$S_N - S_0 = \sum_{k=0}^{N-1} (S_{k+1} - S_k) = \sum_{k=0}^{N-1} (A S_k A^T - S_k + V)$$

$$= A \sum_{k=0}^{N-1} S_k A^T - \sum_{k=0}^{N-1} S_k + V$$

$$= A Y(N) A^T - Y(N) + NV.$$  \hspace{1cm} (12)

One can show by substituting the Lyapunov equations, that

$$\mathbb{E}[J_N] = \text{trace}(Y(N) Q) = \text{trace}( (S_0 - S_N + NV) X^Q)$$

with $Y(N)$ and $X^Q$ being the solution to $0 = A^T Y(N) A^T - Y(N) + S_0 - S_N + NV$ and $0 = A^T X^Q A - X^Q + Q$. With Assumption 2 the covariance is time-invariant, thus the result simplifies to $\mathbb{E}[J_N] = \text{trace} \left( N V^T X^Q \right)$. \hspace{1cm} \blacksquare

Next, we will consider how to design the threshold $\kappa$.

B. Statistical Guarantees

The trigger (10) 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 $\kappa$ can be regarded as a confidence bound, i.e., it is chosen such that with confidence level $\eta$, the deviation term does not exceed $\kappa$. Therefore, observing deviations larger than $\kappa$ can not be sufficiently explained by noise and random fluctuations. Thus, we trigger model learning whenever this happens.

Theorem 1 (Hoeffding's Inequality [28, 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 $\kappa > 0$

$$P \left[ \sum_{i=1}^{n} X_i - n \mu \geq \kappa \right] \leq \exp \left( \frac{-2 \kappa^2}{\sum_{i=1}^{n} (b_i - a_i)^2} \right).$$  \hspace{1cm} (12)

Comparing with (10), the cost samples $\hat{J}_N$ corresponds to the random variables $X_i$ and $\mathbb{E}[J_N]$ to the mean $\mu$. 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, so we cannot apply Hoeffding directly. 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 2: Assume the states are constrained by $\|W^{-1} x_k\| < \alpha$ for all $k$, where $W \in \mathbb{R}^{n \times n}$ is invertible. Then, the cost function $J_N$ is bounded by

$$0 \leq J_N \leq \max_{\|W^{-1} x_k\| < \alpha} J_N = \alpha^2 N \lambda_{\text{max}} \left( W^T Q W \right).$$  \hspace{1cm} (13)

Proof: The lower bound follows immediately from the positive definiteness of $Q$, as $x^T Q x \geq 0$ 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,

$$J_N \leq \max_{\|W^{-1} x_k\| = \alpha} \left[ \sum_{k=0}^{N-1} x_k^T Q x_k \right] \leq N \max_{\|W^{-1} x_k\| = \alpha} \left[ \sum_{k=0}^{N-1} x_k^T Q x_k \right] \leq \alpha^2 N \lambda_{\text{max}} \left( W^T Q W \right).$$  \hspace{1cm} \blacksquare

Remark 1: Even for naturally unconstrained system, considering Assumptions 1 and 2, 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.

**Lemma 3:** By Assumption 2 we have $x_0 \sim N(0, X^V)$. Then, the joint distribution of a sequence of states $(x_0, x_1, \ldots, x_N)$ is a multivariate Gaussian distribution with mean $\mu = 0$ and symmetric block-Toeplitz covariance matrix

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

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

**Lemma 4:** Under Assumptions 1 and 2, and with arbitrarily small $\varepsilon > 0$, there exist an $r_0$ such that $\left|\left[A^r x_k\right]_{i,j}\right| < \varepsilon$ for all matrix-entries $(i,j)$. For any large enough $r > r_0$, the state $x_k$ is approximately independent from the state $x_{k-r}$.

**Proof:** Using Lemma 3, we obtain $\mathbb{E}[x_k x_{k-r}^T] = A^{r_0} X^V$ as the cross-covariance for the jointly multivariate normal distributed states. For multivariate normal distributions, we have that zero cross-covariance is equivalent to independent. Since, as $A$ is Schur-stable by Assumption 1, i.e., $\lambda_{\text{max}}(A) < 1$, the term $A^{r_0}$ approaches zeros as $r_0 \to \infty$ [29]. Hence, by definition of the limit, there exists an $r_0$ such that the absolute value of the cross-covariance is elementwise smaller than $\varepsilon$. 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. 2). 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

$$
\mathcal{L}(k) := \{k - (N + r)i | i \in 0, \ldots, L - 1\}
$$

for the sample average in (10), which by construction has cardinality $L$. Considering the definition of the cost (6), the indices in $\mathcal{L}(k)$ mark the end of each red interval in Fig. 2. 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. 1). Therefore for the time being, we assume the approximation is exact and then, apply Hoeffding to obtain the thresholds. The later introduced Chernoff trigger (Sec. IV) solves this issue in an elegant and clean way, however, cannot be applied here.

**Theorem 2 (Hoeffding Trigger):** Let Assumptions 1 and 2 hold, assume the conditions of Lemma 2 are satisfied, and the samples $J_N(k - (N + r)i)$, $i = 0, \ldots, L - 1$ are mutually independent. Further, let $\eta$ denote the desired confidence level and $\kappa$ is chosen as

$$
\kappa = \sup_j \{J_N(j)\} \sqrt{-\frac{L}{2} \ln \frac{\eta}{2}}
$$

or

$$
= \alpha^2 N \lambda_{\text{max}}(W^T Q W) \sqrt{-\frac{L}{2} \ln \frac{\eta}{2}}.
$$

Then, the probability of triggering with (10), while the model coincides with the ground truth, is bounded by

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

**Proof:** By construction, we can apply Hoeffding’s inequality (Thm. 1) to $J_N$ at the sampling instances $\mathcal{L}(k)$. The bound is given by Lemma 2 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

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

We set $\eta$ to coincide with the upper bound and rearrange for $\kappa$. Thus, we obtain

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

Then, the result is obtained by taking the square root and inserting the value for $\sup J_N$ from the Lemma 2.
C. Numerical Simulation

Next, we will numerically study the trigger architecture, as shown in Fig. 1. We will illustrate the triggering behavior and, in particular, investigate how well model change is detected with the Hoeffding trigger.

1) Setup: Initially, a 5-dimensional system \((A_o, B, V)\) is randomly generated, by sampling the matrices \(A_o \in \mathbb{R}^{5 \times 5}\), \(B \in \mathbb{R}^{5 \times 1}\), and \(V \in \mathbb{R}^{5 \times 5}\) 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 2.

Next, we introduce the model \((\hat{A}_o, \hat{B}, \hat{V})\), 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 \((A_o, B, V)\) 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 Hoeffding trigger is computed from the model as described in (10) and Lemma 1 with \(\eta = 25\%\). For the moving average sampling we use \(N = r = 60\) and \(L = 20\) (cf. Fig. 2). The required state bound \(W\) is set to the covariance of the state and \(\alpha = 18\). These bounds are constant throughout the simulation. Thus, the threshold \(\kappa\) remains constant as well, while the mean \(\mathbb{E}[J_N]\) is the only part of the trigger that changes with model updates.

The system is simulated for 50,000 time steps. At each time step the cost is computed and the trigger value \(\psi = \sum J_N(\cdot) - \mathbb{E}[J_N]\) is derived as discuss before. If the trigger \(|\psi| \geq \kappa\) detects a system change, then the model is set to the true parameters, i.e., \((\hat{A}_o, \hat{B}, \hat{V}) \leftarrow (A_o, B, V)\). 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. V.

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 \((A'_o, B', V')\) 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

\[
\Delta = \beta \frac{(A'_o, B', V') - (A_o, B, V)}{\|(A'_o, B', V') - (A_o, B, V)\|_2},
\]

where \(\beta \in (-0.1, 0.1)\) is also sampled from an uniform distribution. Thus, the new system is obtained by adding \(\Delta\) 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 \(\kappa\) will be reached eventually and thus, triggering is trivial when the system is unstable.

2) Results: Next, we look at the numerical performance. In Fig. 3, a roll-out, which displays many interesting effects, is shown. In the upper plot, we see the normalized cost \(J_N\) and in the lower one the trigger statistic \(\psi\). The system is distorted every 10,000 time steps (green line). A detection (red line) occurs when the trigger value \(\psi\) (in blue) hits either threshold \(\kappa\), which can be seen in the lower plot.

At \(k = 10,000\) the first system change occurs, which is detected after 2480 steps. 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. 2) 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 1383 steps.

The change at \(k = 20,000\) is not detected, 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, would also increase the delays even further. Smaller upper bounds are not possible since these were designed for the initial system. In principle, it would be possible to change the bounds during triggering, however, we
will introduce a cleaner approach in Sec. IV.

At \( k = 30,000 \) there is a fast increase of the cost and a lot of deviation within the signal. This causes a fast detection after only 435 steps. After updating the model, the new controller stabilizes the system nicely within the confidence bounds.

In this run neither instability, significant violation of the upper bound on the cost, nor the issues that arise from the inactive trigger after a detection are shown. Their effects are obvious and observed in other roll-outs.

3) Discussion: First of all, the introduced 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 2 is an issue. In order to apply Hoeffding’s inequality rigorously, we need the bound to hold everywhere. However, this has significant impact on the detection rate since the possibility of large 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.

D. Alternative Triggers

Following the same principles, it is possible to derive alternative triggers that consider different error terms.

One approach would be to consider a relative error, which yields the following trigger

\[
1 - \sum_{j \in \mathcal{E}(k)} \mathcal{J}_N(j) \geq \kappa \Leftrightarrow \gamma_{\text{learn}} = 1, \quad (18a)
\]

\[
\kappa = \frac{\alpha^2 N}{E[J_N]} \lambda_{\text{max}} (W^\top Q W) \sqrt{-\frac{L}{2} \ln \frac{\eta}{2}}. \quad (18b)
\]

However, this trigger is nothing more than an equivalence transform of the previously introduced design (10).

An alternative design could be based on higher moments. For example, under the same assumptions and conditions as in Thm. 2, we can, obtain the trigger

\[
\sum_{j \in \mathcal{E}(k)} \mathcal{J}_N^2(j) - L E[J_N^2] \geq \kappa \Leftrightarrow \gamma_{\text{learn}} = 1 \quad (19a)
\]

\[
\kappa = \alpha^2 N^2 \lambda_{\text{max}}^2 (W^\top Q W) \sqrt{-\frac{L}{2} \ln \frac{\eta}{2}}, \quad (19b)
\]

which is based on the second moment. The value of the second moment is given by the following lemma.

\textit{Lemma 5:} Under Assumption 1 and 2, the expected value for the cost \( J_N \) with respect to the system (5) is given by

\[
E[J_N^2] = E[J_N]^2 - 2 \left( \mu_0^\top F \mu_0 \right)^2 + 4 \text{trace} (EX^V Q X^V) \quad (20)
\]

\[
\mathcal{E} := N \left( F - \frac{Q}{2} \right) + F - \sum_{k=0}^{N-1} (A^k)\top GA^k
\]

\[
\mathcal{F} := \sum_{k=0}^{\infty} (A^k)\top QA^k \quad \mathcal{G} := \sum_{k=0}^{\infty} (A^k)\top QA^k.
\]

The Lemma is the discrete-time analog of [15, Thm. 4] and as such a useful result in its own right. The proof is similar to [15] and thus, omitted here. The summations in \( \mathcal{E}, \mathcal{F}, \) and \( \mathcal{G} \) can be found using appropriate Lyapunov equations, similarly to Lemma 1 and [15].

Further, it is possible to obtain similar results to Thm. 2. However, our numerical investigations – analog to Sec. III-C – showed that it brings no significant advantage over the mean trigger (10). The same holds true for the centered second moment \( \mathbb{V}[J_N] \). Thus, we decided to not further discuss these results in this article. Even more so in regard to the Chernoff trigger that leverages the whole distribution, yields superior theoretical properties, and better empirical performance.

IV. DISTRIBUTION-BASED TRIGGER

The Hoeffding trigger is a straightforward and intuitive trigger based on a comparison between averaging and the analytically derived expected value. The previous example demonstrated that it is generally effective, albeit it may miss smaller changes, detection may take rather long, and there are some theoretical shortcomings (i.e., independent samples and sharp upper bounds). Next, we will present an improved trigger design that leverages not just the mean but the complete statistical information and thus, does not suffer from any of the above mentioned limitations.

A. Moment Generating Functions

The moment-generating function (MGF) \( M_X(\xi) := E[e^{\xi X}] \) of a random variable \( X \) if it exists is a powerful tool to characterize the distribution (see, e.g., [30, 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 \( E[X^n] \) can be obtained by computing \( \frac{d}{d\xi} M_X(\xi)|_{\xi=0} \).

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

\textit{Theorem 3 (Moment Generating Function of the Cost):} Assuming the state sequence \( z = (x_0, x_1, \ldots, x_{N-1})^\top \) is a jointly normally distributed random variable with mean \( \mu \) and covariance matrix \( \Sigma \). 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

\[
M_{J_N}(\xi; \mu, \Sigma, \Omega) = \exp \left( \frac{1}{2} \mu^\top \left( (1-2\xi \Omega \Sigma)^{-1} - 1 \right) \Sigma^{-1} \mu \right) \frac{1}{\sqrt{\det (1-2\xi \Omega \Sigma)}}
\]

(21)

where \( \xi \in [-\infty, \frac{1}{2\lambda_{\text{max}}(\Omega \Sigma)}] \) and \( \Omega = \text{diag}(Q, \ldots, Q) \) with weight matrix \( Q \).
Proof: It is a known fact that there exits an $m \times m$ matrix $T$ such that \( \det T \neq 0 \), $T^\top T = I$, $T^\top Q T = \Lambda$, and $\Sigma \Omega = T^{-\top} \Lambda T^\top$, where $\Lambda$ has the eigenvalues $\lambda_i$ of $\Sigma \Omega$ on the diagonal, given that $\Sigma$ and $\Omega$ are symmetric and $\Sigma$ is positive definite. As both $\Sigma$ and $\Omega$ 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

$$M_{J_N}(\xi) = \mathbb{E}[e^{\xi X} \Omega z] = \int_{\mathbb{R}^N} \exp(\xi x^\top \Omega x) dF_z(x)$$

\[= \det(2\pi^\top)^{-\frac{1}{2}} \int_{\mathbb{R}^N} \exp \left( \xi x^\top \Omega x - \frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right) dx,\]

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

$$M_{J_N}(\xi) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp \left( \xi \lambda_i (y_i + c_i)^2 - \frac{1}{2} y_i^2 \right) dy_i$$

$$= \left[ \prod_{i=1}^{m} \frac{1}{\sqrt{1 - 2\xi \lambda_i}} \right] \left[ \exp \sum_{i=1}^{m} \frac{1}{2} \xi^2 \lambda_i \right]$$

$$= \exp \left( \frac{1}{2} \mu^\top (I - 2\xi \Omega \Sigma)^{-1} - 1 \right) \Sigma^{-1} \mu \right) \sqrt{\det(I - 2\xi \Omega \Sigma)}.$$

By Assumption 1 and 2, we have $\mu = 0$ and $\Sigma = \text{const}$. This yields the simplified form

$$M_{J_N}(\xi; \Sigma, \Omega) = \det(I - 2\xi \Omega \Sigma)^{-\frac{1}{2}}, \quad (22)$$

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.

To compare this with the result from the previous chapter, let us compute the moments given by the MGF.

**Lemma 6:** 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

$$\mathbb{E}[J_N] = \left. \frac{d}{d\xi} M_{J_N}(\xi) \right|_{\xi=0} = \text{trace} \Sigma \Omega \quad (23a)$$

$$\mathbb{E}[J_N^2] = \left. \frac{d^2}{d\xi^2} M_{J_N}(\xi) \right|_{\xi=0} = 2 \text{trace}(\Sigma \Omega \Sigma) + \text{trace}^2 \Sigma \Omega \quad (23b)$$

**Proof:** The result follows from using Jacobi's formula

$$\frac{d}{dt} \det A(t) = \det A(t) \text{trace} \left( A^{-1} \frac{d}{dt} A(t) \right) \quad [31, \text{Sec. 8.3}]$$

to compute the derivatives of the moment-generating function. This result is equivalent to the previous results from Lemmata 1 and 5 for the expected value and second moment. However, here we need to explicitly compute $\Sigma$, which is less computationally efficient for individual moments, than the previous direct solution, where $\Sigma$ is implicit.

**B. Chernoff Trigger**

In order to obtain an effective trigger with theoretical guarantees, we need a different concentration inequality result. Further, we want to make use of the whole distribution instead of just the expected value as with Hoeffding’s inequality.

Next, we introduce the Chernoff bound and utilize it to derive the trigger threshold $\kappa$.

**Theorem 4 (Chernoff Bound, [32, Thm. 1]):** Given the moment-generating function $\mathbb{E}[e^{\xi X}]$ of the random variable $X$, for any real number $\xi > 0$, it holds that

$$\mathbb{P}[X \geq \kappa] \leq \frac{M_X(\xi)}{e^{\xi \kappa}} \quad \mathbb{P}[X \leq \kappa] \leq \frac{M_X(-\xi)}{e^{-\xi \kappa}}. \quad (24)$$

In particular, it holds that

$$\mathbb{P}[X \geq \kappa] \leq \inf_{\xi > 0} \frac{M_X(\xi)}{e^{\xi \kappa}} \quad \mathbb{P}[X \leq \kappa] \leq \inf_{\xi < 0} \frac{M_X(\xi)}{e^{-\xi \kappa}}. \quad (25)$$

**Proof:** Follows from Markov’s inequality applied to $e^{\xi X}$.

Thus, we can state the main theorem of this article, which is the full distributional analog to Thm. 2.

**Theorem 5 (Chernoff Trigger):** Let the parameter $N \in \mathbb{N}$ be given and Assumptions 1 and 2 hold. Then, we can obtain for any time-index $k$ an upper bound $\eta \in (0, 1)$ for the probability

$$\mathbb{P}[J_N(k) \notin (\kappa^-, \kappa^+)] \leq \eta, \quad (26)$$

where the thresholds are chosen in the following

$$\kappa^+ = \sup_{\xi \in (0, \pi_{\text{max}})} \chi(\xi) \quad \kappa^- = \inf_{\xi \in (-\infty, 0)} \chi(\xi) \quad (27)$$

$$\chi(\xi) = -\frac{1}{\xi} \ln \frac{\eta}{2} - \frac{1}{2\xi} \ln \det(I - 2\xi \Omega \Sigma) \quad (28a)$$

$$= -\frac{1}{\eta} \frac{\eta}{2} - \frac{1}{2\xi} \sum_{j=0}^{N_n} \ln(1 - 2\xi \lambda_j). \quad (28b)$$

Further, $\lambda_j$ are the eigenvalues of $\Omega \Sigma$, the state covariance matrix is denoted as $\Sigma$ (as introduced in Lemma 3), and the weight matrix $\Omega = \text{diag}(Q, \ldots, Q)$.

**Proof:** We distribute the tail probability $\eta$ symmetrically to both sides of the interval. Thus,

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

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

$$\inf_{\xi > 0} \frac{M_X(\xi)}{e^{\xi \kappa^+}} \leftrightarrow 0 = \inf_{\xi > 0} \ln M_X(\xi) - \xi \kappa^+ - \ln \frac{\eta}{2} \bigg|_{\xi > 0}$$

$$\leftrightarrow \kappa^+ = \inf_{\xi > 0} \frac{1}{\xi} \ln M_X(\xi) - \frac{1}{\eta} \ln \frac{\eta}{2}. \quad (29)$$

We can proceed similarly for $\kappa^-$, just that the infimum turns into a supremum, when we divide by $\xi$ as $\xi < 0$. By inserting the simplified MGF from (22) 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:

$$\hat{J}_N(j) \notin (\kappa^-, \kappa^+) \iff \gamma_{\text{learn}} = 1, \quad (29)$$

with $\kappa^\pm$ as introduced in Thm. 5.

The proposed trigger considers only one sample of the cost function, however, with a longer horizon $N$ (cf. Fig. 2). This is substantially different to the Hoeffding trigger (10) and thus, we avoid any independence requirement between samples. The
longer horizon is equivalent to considering multiple consecutive sample, as they can be joined into one longer sample, e.g., for two samples we have \( J_N(k - N) + J_N(k) = J_{2N} \).

Further, we do not assume any bounds on \( J_N \). This is a significant improvement over the Hoeffding trigger (10) since we effectively obtain tight bounds. Therefore, detection speed and data efficiency should be dramatically improved, which we will later investigate further in numerical simulations.

Intuitively, the Chernoff trigger is tailored tightly around the distribution of the cost and analysis how likely it is that observed samples are drawn from said distribution. The previous trigger was oblivious to any statistical information besides the expected value. It exclusively analyzed the deviation between empirical mean and expected value.

In order to obtain the thresholds \( \kappa^\pm \) we need to solve the two optimization problems (27). However, this is easily tractable online due to the following properties of the objective function \( \chi \). Intuitively this can also been seen in Fig. 4, where the general shape of the objective function is illustrated.

**Theorem 6:** The function \( \chi(\xi) \) is strictly convex in the range for \( \kappa^+ \), and thus has only one minimum on the interval.

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

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

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. □

**Remark 2:** 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.

C. Numerical Simulation

Next, we analyze the empirical performance of trigger (29).

1) Setup: We use the same setup as before in Sec. III-C, where we investigated the properties of the Hoeffding trigger (10). Here, we replace the trigger with the new design (29), a horizon of \( N = 200 \) and \( \eta = 1\% \). Further, we use the same random seed as before and the result can be seen in Fig. 5. Thus, we can directly compare to Fig. 3.

2) Results: The trigger detects every system changes, including the one at \( k = 20000 \), which the Hoeffding trigger failed to detect.

Secondly, 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 indented. 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 \( \eta = 1\% \).

Due to the superior design, the proposed trigger is way faster than the Hoeffding trigger. 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.

3) Discussion: Since the trigger thresholds are tailored to the actual distribution, we can see superior performance. In particular, the adaptivity of the thresholds to different magnitudes of process noise can be clearly seen in Fig. 5. For instance at \( k = 40000 \) and afterward, there is little deviation in the cost and this is also captured in the bounds. However, between \( k = 0 \) and \( k = 10000 \) 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

![Fig. 4. 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, \( \kappa^- \) and \( \kappa^+ \), are found through straightforward maximization (over \( \xi < 0 \)) and minimization (\( \xi > 0 \)), respectively.](image)

![Fig. 5. Numerical simulation of the Chernoff trigger on a 5-dimensional system with random \( A \) 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.](image)
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. V, where we consider a hardware experiment.

D. Detection Delay

In order 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 3 976 366 system changes.

1) 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 $G$ with input $w$ and output $y$ is defined as

$$
\|G\|_{H_2} = \lim_{k \to \infty} \frac{\text{trace} \left( E[y_k y_k^\top]\right)}{\sqrt{\text{trace} \left( E[w_k w_k^\top]\right)}}.
$$

Further, we decided to measure the system change as

$$
\delta_{\text{sys}} := \sqrt{A_{\text{new}} - A_{\text{old}}},
$$

with the old and new closed-loop system matrix $A$ and the square root of process noise covariance $V$. The $\sqrt{\cdot}$-notation represents the system and is commonly used in the robust control community [20, Sec. 3]. We use the square root of $V$ as input since this will transform the white noise of the norm into the actual Gaussian noise.

2) Estimating Probability Density $P(T_D|\delta_{\text{sys}})$: Additionally, we need to clarify, how we measure the detection time $T_D$. 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. V).

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

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

3) Results: In Fig. 6, the obtained density function for $P(T_D|\delta_{\text{sys}})$ is shown. For the visualization the graph renormalized such that $\forall \theta : \max_x \{P(T_D = \tau | \delta_{\text{sys}} = \theta)\} = 1$. 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. 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. 6, in particular, around $\delta_{\text{sys}} = 1$.

V. 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. We focus on the CHERNOFF trigger, which proved to be superior in the numerical experiments and has better theoretical properties.
We modify the trigger slightly. We introduce the additional constraint 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 on the control input. Choosing a signal that is both sufficiently exciting [34] for possibly changed dynamics of the pendulum and avoids the hardware constraints in $\theta$ 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 toolbox1-3, 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. 5 and Equation (27)), which completes the model update.

### C. Results

In Fig. 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 an slightly 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

\[ J_N(k) = \left[ \theta, \dot{\theta}, e \right] \]

Fig. 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.

Fig. 8. Experimental run of the Chernoff trigger on the rotary pendulum (Fig. 7). The black lines indicate $\kappa^+$ and $\kappa^-$, respectively. Additionally, the model-derived expected value (cf. Lemma 6) 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 line, the physical system is changed by adding a weight.
faulty model. The main goal is to show that we are able to detect change systematically and thus, effectively reduce the cost by updating the controller.

As we can see in the very beginning of Fig. 8, the measured cost does not lie within the interval \((k^- , k^+)\). 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 downcrossing caused by random effects. Hence, the change is only detect 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 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. 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.

### D. 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 \(k^\pm\) 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 \(\alpha\) and the base introduces a time-variant nonlinearity into the setup. As this wire twists randomly during operation and swing-up, the equilibrium state without input changes in \(\theta\). Additionally, the wire applies a force towards some \(\theta\), 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 system might be possible, but requires some adjustment in our approach.

### VI. Conclusion

In this article, we propose event-triggered learning for control as a novel concept to trigger model learning when needed. 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 change and reduces the incurred control cost and steady-state variance.

### ACKNOWLEDGEMENT

The authors thank Jonathan Fiene and Ruben Werbke for their support with the experimental setup.

### REFERENCES


Henning Schlüter received the B. Sc. and M. Sc. degrees in engineering cybernetics from the University of Stuttgart in Germany, in 2017 and 2019, respectively. He has since been a doctoral student at the Institute for Systems Theory and Automatic Control of the University of Stuttgart under the supervision of Prof. Frank Allgöwer and a member of the International Max Planck Research School for Intelligent Systems. His research interests are in the area of model predictive control, and stochastic control.

Friedrich Solowjow received B.Sc. degrees in Mathematics and Economics from the University of Bonn in 2014 and 2015, respectively, and a M.Sc. degree in Mathematics also from the University of Bonn in 2017. He is currently a PhD student in the Intelligent Control Systems Group at the Max Planck Institute for Intelligent Systems, Stuttgart, Germany and a member of the International Max Planck Research School for Intelligent Systems. His main research interests are in systems and control theory and machine learning.

Sebastian Trimpe (M’12) received the B. Sc. degree in general engineering science and the M.Sc. degree (Dipl.-Ing.) in electrical engineering from Hamburg University of Technology, Hamburg, Germany, in 2005 and 2007, respectively, and the Ph. D. degree (Dr. sc.) in mechanical engineering from ETH Zurich, Zurich, Switzerland, in 2013. He is currently a Research Group Leader at the Max Planck Institute for Intelligent Systems, Stuttgart, Germany, where he leads the independent Max Planck Research Group on Intelligent Control Systems. His main research interests are in systems and control theory, machine learning, networked and autonomous systems. Dr. Trimpe is recipient of several awards, among others, the triennial IFAC World Congress Interactive Paper Prize (2011), the Klaus Tschira Award for achievements in public understanding of science (2014), the Best Demo Award of the International Conference on Information Processing in Sensor Networks (2019), and the Best Paper Award of the International Conference on Cyber-Physical Systems (2019).