Optimization-Based Iterative Learning Control for Trajectory Tracking

Angela Schöllig and Raffaello D'Andrea

***Abstract*— In this paper, an optimization-based iterative learning control approach is presented. Given a desired trajectory to be followed, the proposed learning algorithm improves the system performance from trial to trial by exploiting the experience gained from previous repetitions. Taking advantage of the a-priori knowledge about the systems dominating dynamics, a data-based update rule is derived which adapts the feedforward input signal after each trial. By combining traditional model-based optimal filtering methods with state-of-the-art optimization techniques such as convex programming, an effective and computationally highly efficient learning strategy is obtained. Moreover, the derived formalism allows for the direct treatment of input and state constraints. Different (nonlinear) performance objectives can be specified defining the overall learning behavior. Finally, the proposed algorithm is successfully applied to the benchmark problem of swinging up a pendulum using open-loop control only.**

***Index Terms*— Iterative learning control, state and input constraints, Kalman filtering, convex programming, inverted pendulum, swing-up.**

I. Introduction

Controlled and automated systems have an unbelievably wide use in many different areas of application ranging from manufacturing systems, transportation, and space engineering to biological engineering and robotics. The operation of increasingly complex systems requires sophisticated and intelligent control methods guaranteeing high performance even under the presence of model uncertainties, disturbances, and measurement noise. These challenges are met by control strategies which adapt to unknown or changing environments and use previous experiences to learn and improve performance. Moreover, repetition is an inherent feature of almost every automated process. The learning algorithm proposed herein takes advantage of this repetitive operating scheme. The goal is to precisely follow a desired trajectory. Data from previous trials is exploited and the gained experience is used to improve the performance of the following execution by updating the feedforward input signal. With this, the algorithm compensates for unmodeled system dynamics, repetitive noise, or parametric uncertainties.

The presented approach can be characterized as an iterative learning control (ILC) technique. ILC became a popular research topic beginning with [1], and since then has proven to be a very powerful method for high performance reference tracking. A recent overview of ILC with an extensive bibliography is available in [2] and [3]. Yet methods from optimal control theory have only recently been applied to the design of ILC laws. Based on a so-called 'lifted' domain representation, see [4]–[6], LQG-type solutions were proposed [3], [7]–[10] estimating the tracking error and minimizing a quadratic cost function. The approach presented in this paper clearly divides both steps, estimation and control, illustrated in Figure 1. In the first step, a time-varying Kalman filter is designed, which estimates the model error along the trajectory. The estimated error not only serves as the input to the following control step, but also provides additional insight into the real dynamics along the trajectory and allows a later interpretation in terms of the real system's behavior. In the second step, the control objective is formulated as a convex optimization problem [11]. Here, in contrast to least-squares approaches or LQG design [3], [7]–[10], input and state constraints can be explicitly incorporated.

A. Schöllig and R. D'Andrea are with the Institute for Dynamic Systems and Control, Swiss Federal Institute of Technology (ETH Zurich), 8092 Zürich, Switzerland. {aschoellig,rdandrea}@ethz.ch

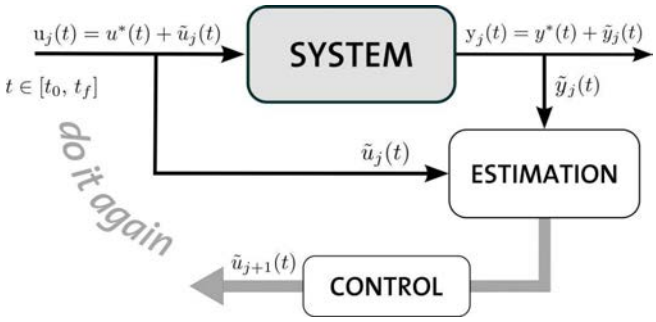


Fig. 1: The general ILC framework considered in this paper: A complete trial $u_j(t),\ t \in [t_0,\ t_f]$ is performed. Based on the output error $\tilde{y}_j(t)$, a new input $u_{j+1}(t)$ is calculated and applied during the next trial.

The algorithm proceeds as follows: A simple, possibly nonlinear, dynamic model is derived, which captures the essential dynamics of the real system. Based on this model, a nominal input trajectory is calculated, which yields the desired reference trajectory. By linearizing the system about the nominal trajectory, a time-varying linear state space model is obtained which approximates the system dynamics along the reference trajectory, cf. [7]. With a time-discretized version of this model, cf. [2], [3], [12], the system's lifted domain representation is derived, which characterizes the actual system dynamics by a static map whereas the learning dynamics are described by a difference equation. Based on this notation, the estimation and control steps are defined. When performing the first trial, the experimental results are stored and compared to the desired trajectory. The resulting error vector is fed into the Kalman filter providing a new estimate of the modeling error. With this information, the following control step determines a more adequate input trajectory by solving a constrained optimization problem. When starting the next trial, the updated input is applied.

The novelty of the presented approach lies in the combination of optimal filtering methods with convex optimization techniques. Different performance objectives can be defined for the learning process by choosing appropriate vector norms and adequate scaling in the control step. Moreover, the estimation equations explicitly take noise characteristics into account. Input and state constraints are directly incorporated in the calculation of the input update. Finally, the effectiveness of our approach is shown by successfully learning the swing-up of an inverted pendulum in a very small number of trials while only applying feedforward control.

II. Dynamics and System Representation

The starting point of our approach is a time-varying nonlinear model of the form

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t) \\ y(t) &= g(x(t), u(t), t), \end{aligned} \tag{1}$$

978-3-9524173-9-3

which captures the key dynamics of a real physical system with control input $\mathrm{u}(t) \in \mathbb{R}^{n_\mathrm{u}}$, state $\mathrm{x}(t) \in \mathbb{R}^{n_\mathrm{x}}$, and output $\mathrm{y}(t) \in \mathbb{R}^{n_\mathrm{y}}$. The vector fields f and g are assumed to be continuously differentiable in x and u. Constraints on the input $\mathrm{u}(t)$ and the state $\mathrm{x}(t)$ are defined componentwise,

$$\begin{aligned} \mathrm{u}_{\min} \leq\ & \mathrm{u}(t)\ \leq \mathrm{u}_{\max} \\ \mathrm{x}_{\min} \leq\ & \mathrm{x}(t)\ \leq \mathrm{x}_{\max}\,, \quad \forall\, t \geq 0\,, \end{aligned} \tag{2}$$

where $\mathrm{u}_{\min,\,\max} \in \mathbb{R}^{n_\mathrm{u}}$ and $\mathrm{x}_{\min,\,\max} \in \mathbb{R}^{n_\mathrm{x}}$ represent the lower and upper bounds on $\mathrm{u}(t)$ and $\mathrm{x}(t)$.

The goal of the presented learning algorithm is to track an *a priori* determined output trajectory $y^*(t)$ over a finite time interval $t \in \mathcal{T} = [t_0,\, t_f]\,,\ t_f < \infty$. The desired trajectory $y^*(t),\ t \in \mathcal{T}$ is assumed to be feasible with respect to the nominal model (1), (2). That is, there exists a triple

$$(u^*(t),\, x^*(t),\, y^*(t))\,, \quad t \in \mathcal{T}\,, \tag{3}$$

satisfying Equations (1) and (2). For some applications, the desired output trajectory $y^*(t)$ may be known ahead of time. However, it may also be the result of an optimal control problem solved based on the simplified system dynamics (1).

We assume that the motion of the system stays close to the generated reference trajectory (3) during the learning process. Only considering small deviations $(\tilde{u}(t),\, \tilde{x}(t),\, \tilde{y}(t))$ from the desired trajectory (3),

$$\tilde{u}(t) = \mathrm{u}(t) - u^*(t),\ \tilde{x}(t) = \mathrm{x}(t) - x^*(t),\ \tilde{y}(t) = \mathrm{y}(t) - y^*(t), \tag{4}$$

the system's behavior (1) can be approximated by a first-order Taylor series expansion about the reference trajectory (3) resulting in the following linear, time-varying system

$$\begin{aligned} \dot{\tilde{x}}(t) &= A(t)\tilde{x}(t) + B(t)\tilde{u}(t) \\ \tilde{y}(t) &= C(t)\tilde{x}(t) + D(t)\tilde{u}(t)\,, \quad t \in \mathcal{T}\,, \end{aligned} \tag{5}$$

where the time-dependent matrices $A(t)$, $B(t)$, $C(t)$, $D(t)$ are the corresponding Jacobian matrices of the nonlinear functions f and g with respect to x and u.

The developed learning algorithm subsequently makes use of data collected during previous iterations. With the goal of improving the performance of the system during the next iteration, a model-based update rule is used to calculate a new, more adequate input trajectory, cf. Figure 1. Since measurements of previous trials are only available at fixed time intervals, a discrete-time representation of the plant dynamics (5) is introduced, see [2], [3], [12], and references therein. Converting (5) to a discrete-time system results in the following linear, time-varying difference equations,

$$\begin{aligned} \tilde{x}(k+1) &= A_D(k)\tilde{x}(k) + B_D(k)\tilde{u}(k) \\ \tilde{y}(k) &= C_D(k)\tilde{x}(k) + D_D(k)\tilde{u}(k)\,, \end{aligned} \tag{6}$$

where $k \in \mathcal{K} = \{0,\, 1,\, \ldots,\, N\}\,,\ N < \infty$ represents the discrete-time index. To simplify notation, we use $\tilde{x}(k)$ to denote $\tilde{x}(k\Delta t)$, where Δt is the sampling time. The desired trajectory (3) is represented by a $(N+1)$-sample sequence

$$(u^*(k),\, x^*(k),\, y^*(k))\,, \quad k \in \mathcal{K}\,. \tag{7}$$

Other associated signals, see e.g. (4), are discretized analogously. The input and state constraints (2) read now as

$$\begin{aligned} u_{\min}(k) \leq\ & \tilde{u}(k)\ \leq u_{\max}(k) \\ x_{\min}(k) \leq\ & \tilde{x}(k)\ \leq x_{\max}(k)\,, \end{aligned} \tag{8}$$

where $u_{\min,\,\max}(k) \in \mathbb{R}^{n_\mathrm{u}}$ and $x_{\min,\,\max}(k) \in \mathbb{R}^{n_\mathrm{x}}$. The values of these vectors depend on the discretization method and are, for example, given by an equation of the form

$$u_{\min}(k) = \mathrm{u}_{\min} - \min_{\substack{t \in \\ [k\Delta t,\,(k+1)\Delta t]}} u^*(t)\,, \tag{9}$$

when using a zero-order hold with sample time Δt.

Performing a new trial, the deviations $(\tilde{u}(k),\, \tilde{x}(k),\, \tilde{y}(k)),\ k \in \mathcal{K}$, with respect to the desired trajectory (7) are given by the following lifted vector representation, cf. [13]:

$$\begin{aligned} u &= [\,\tilde{u}(0),\, \tilde{u}(1),\, \ldots,\, \tilde{u}(N)\,]^T \ \in \mathbb{R}^{(N+1)n_\mathrm{u}} \\ x &= [\,\tilde{x}(0),\, \tilde{x}(1),\, \ldots,\, \tilde{x}(N)\,]^T \ \in \mathbb{R}^{(N+1)n_\mathrm{x}} \\ y &= [\,\tilde{y}(0),\, \tilde{y}(1),\, \ldots,\, \tilde{y}(N)\,]^T \ \in \mathbb{R}^{(N+1)n_\mathrm{y}}. \end{aligned} \tag{10}$$

This notation allows us to capture the dynamic relation (6) between input, state, and output trajectories (10) by a simple static mapping

$$\begin{aligned} x &= Fu + d^0 \\ y &= Gx + Hu\,, \end{aligned} \tag{11}$$

Here, the lifted matrix $F \in \mathbb{R}^{(N+1)n_\mathrm{x} \times (N+1)n_\mathrm{u}}$ is composed of the matrices $F_{(l,m)} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}},\ \ 0 \leq l, m \leq N$,

$$F = \begin{bmatrix} F_{(0,0)} & \cdots & F_{(0,N)} \\ \vdots & \ddots & \vdots \\ F_{(N,0)} & \cdots & F_{(N,N)} \end{bmatrix}\,, \tag{12}$$

where

$$F_{(l,m)} = \begin{cases} A_D(l-1)\ldots A_D(m+1)B_D(m) & \text{if } m < l-1 \\ B_D(m) & \text{if } m = l-1 \\ 0 & \text{if } m > l-1\,. \end{cases}$$

Matrices G and H are block-diagonal and analogously defined by

$$G_{(l,m)} = \begin{cases} C_D(l) & \text{if } l = m \\ 0 & \text{otherwise} \end{cases}$$

and

$$H_{(l,m)} = \begin{cases} D_D(l) & \text{if } l = m \\ 0 & \text{otherwise}\,, \end{cases}$$

respectively, where, for $0 \leq l, m \leq N$, $G_{(l,m)} \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{x}}$ and $H_{(l,m)} \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{u}}$. Vector d^0 contains the free response of the system (6) to the initial deviation $\tilde{x}(0) = \tilde{x}_0 \in \mathbb{R}^{n_\mathrm{x}}$,

$$d^0 = \left[\tilde{x}_0,\, A_D(0)\tilde{x}_0,\, A_D(1)A_D(0)\tilde{x}_0,\, \ldots,\, \prod_{q=0}^{N-1} A_D(q)\,\tilde{x}_0\right]^T.$$

The introduced lifting technique is extremely well suited for the analysis and synthesis of iterative learning control schemes, where the system is assumed to operate in a repetitive mode, cf. [4]–[6], [9]. The static linear system (11) captures the complete *time-domain* dynamics of a single trial by mapping the finite input time series $\tilde{u}(k),\ k \in \mathcal{K}$ into the corresponding output time series $\tilde{y}(k),\ k \in \mathcal{K}$. At the end of each execution, the state is reset to a specified repetition-independent initial condition and a new trial is started with an updated control. The goal of ILC is to update the feedforward signal u, see Equation (10), based on the data gathered during the previous executions aiming at improving the system's performance over *iteration time*. The *dynamics of the learning*, i.e., the dynamic behavior of a sequence of consecutive trials, can be characterized in the lifted domain by introducing a subscript j indicating the jth execution of the desired task, $j \in \{1,\, 2,\, \ldots\}$. Using this notation, the dynamics associated with the learning algorithm are described by difference equations in the *iteration-time domain* j.

The remainder of this section is devoted to an adequate incorpora-

tion of noise in the system description (11) and to an explanation of how the system evolves over several iterations. In order to explicitly take different noise sources into account, the equations in (11) are kept separate and disturbances are included as follows

$$\begin{aligned} x_j = &\quad Fu_j + d_j \quad + N_\xi \xi_j \\ y_j = &\quad Gx_j + Hu_j \quad + N_\upsilon \upsilon_j \,. \end{aligned} \tag{13}$$

Here, j denotes the jth trial, $\xi_j \in \mathbb{R}^{n_\xi}$ can be interpreted as process disturbance, and $\upsilon_j \in \mathbb{R}^{n_\upsilon}$ is considered as measurement or sensor noise. Moreover, the random variable ξ_j also captures the zero-mean noise component of the initial condition. Both random variables, ξ_j and υ_j, are assumed to be trial-uncorrelated sequences of zero-mean Gaussian white noise with covariance Ξ_j and Υ_j, respectively; that is, $\xi_j \sim \mathcal{N}(0, \Xi_j)$ and $\upsilon_j \sim \mathcal{N}(0, \Upsilon_j)$. The disturbance joint covariance is given by

$$E\left[\begin{bmatrix}\xi_j \\ \upsilon_j\end{bmatrix}\begin{bmatrix}\xi_j^T & \upsilon_j^T\end{bmatrix}\right] = \begin{bmatrix}\Xi_j & \Delta_j \\ \Delta_j^T & \Upsilon_j\end{bmatrix} \quad \text{with } \Delta_j = E\left[\xi_j \upsilon_j^T\right]. \tag{14}$$

$E[\cdot]$ denotes the expected value, and Δ_j is zero, if ξ_j and υ_j are uncorrelated. The properties of the introduced disturbances ξ_j, υ_j, which play a major role in the estimation step presented in Section III, may be obtained by carrying through given noise characteristics of sensors and known process disturbances of the real system from the original model description (1) to the lifted domain representation (13). The vector d_j represents the model error along the reference trajectory which shows only slight changes from iteration to iteration. Repeating disturbances [14] as well as repeated nonzero initial conditions [15], which are previously modeled by d^0, Equation (11), are captured in d_j. The iteration-domain dynamics of d_j is described by the subsequent difference equation

$$d_j = d_{j-1} + \omega_{j-1}\,, \tag{15}$$

with ω_j being another trial-uncorrelated sequence of zero-mean Gaussian white noise characterized by $\omega_j \sim \mathcal{N}(0, \Omega_j)$.

Subsequently, some significant characteristics of the iteration-domain systems dynamics (13), (15) are summarized which are indispensable for understanding the overall operation of the proposed learning algorithm developed in Sections III and IV. The state deviation x_j along the reference trajectory x^*, which is analogously defined by

$$x^* = [\,x^*(0),\, x^*(1),\, \ldots,\, x^*(N)\,]^T \;\in \mathbb{R}^{(N+1)n_\mathrm{x}}\,, \tag{16}$$

is affected by two different noise sources, a trial-uncorrelated zero-mean component ξ_j and a 'random walk' component d_j. This versatile noise model includes the stochasticity of the process noise ξ_j and the repetitive nature of the modeling errors d_j, which can vary between trials trough the influence of ω_j, see also [7], [10], [16]. In particular, the modeling error d_j captures all non zero-mean noise effects along the desired trajectory x^*. Since the above derivations are based on a fairly simple model of the real system, Equation (1), not taking complex and possibly high-frequency dynamics into account, the disturbance d_j may also be interpreted as a vector representation of all unmodeled dynamics along the desired trajectory x^*. As a result, d_j might depend on the applied input $\mathrm{u}(t) = u^*(t) + \tilde{u}(t)$, $t \in \mathcal{T}$. The ultimate goal of the subsequent derivations is to estimate and optimally compensate for the errors d_j by updating the input trajectory appropriately. Assuming a convergent input $(u^* + u_j)$ for an increasing number j of trials, the sequence d_j convergences, too. Hence, one possible definition of the covariance Ω_j reflecting previous considerations is

$$\Omega_j = \epsilon_j I\,, \quad \text{with} \quad \epsilon_j < \epsilon_{j-1}\,, \tag{17}$$

where $I \in \mathbb{R}^{(N+1)n_\mathrm{x} \times (N+1)n_\mathrm{x}}$ denotes the identity matrix and $\epsilon_j > 0\ \forall j \in \mathbb{N}\backslash\{0\}$. Characterizing the noise ω_j by a time-varying covariance Ω_j, as for example defined by (17), supports a fast convergence of the proposed learning algorithm.

To complete the lifted representation (13), (15), the constraints (8) are transformed appropriately. Stacking the bounds $u_{(\min,\,\max)}(k)$ and $x_{(\min,\,\max)}(k)$ in vectors as $u_\mathrm{min} = [u_\mathrm{min}(0),\, u_\mathrm{min}(1),\, \cdots,\, u_\mathrm{min}(N)]^T \in \mathbb{R}^{(N+1)n_\mathrm{u}}$, constraints (8) read as

$$\begin{aligned} u_\mathrm{min} \;&\leq\; u \;\leq u_\mathrm{max} \\ x_\mathrm{min} \;&\leq\; x \;\leq x_\mathrm{max}\,. \end{aligned} \tag{18}$$

To conclude, it is particularly important to highlight the flexibility and universality of the lifted representation, summarized by Equations (13), (15), and (18) which, subsequently, allows for the derivation and the execution of operations in the trial-time domain.

III. Estimation

The proposed learning algorithm is considered as a two-step update law, see Figure 1. In a first step, the modeling error d_j along the desired trajectory is *estimated* using optimal filtering techniques [17]. The subsequent *control* step, then, provides a new control input $u_{j+1} \in \mathbb{R}^{(N+1)n_\mathrm{u}}$ optimally compensating for the estimated vector $\widehat{d}_{j\,|\,j}$.

An iteration-domain Kalman filter is proposed retaining all available information from previous trials, namely the output signals $y_0,\, y_1,\, \ldots,\, y_j$, in order to estimate the current error d_j. Combining Equations (13) and (15), and recalling corresponding noise characteristics, a discrete-time system is obtained fitting into the standard Kalman filter approach, cf. [18]:

$$\begin{aligned} d_j = &\quad d_{j-1} + \omega_{j-1} \\ y_j = &\quad G\,d_j + (GF + H)\,u_j + \mu_j\,, \end{aligned} \tag{19}$$

where

$$\mu_j = \begin{bmatrix} GN_\xi & N_\upsilon \end{bmatrix}\begin{bmatrix}\xi_j \\ \upsilon_j\end{bmatrix}. \tag{20}$$

With the previous noise definitions, cf. Equation (14), the stochastic variable μ_j, $j \in \{1,\, 2,\, \ldots\}$, is characterized by $\mu_j \sim \mathcal{N}(0, M_j)$, where

$$M_j = \begin{bmatrix} GN_\xi & N_\upsilon \end{bmatrix}\begin{bmatrix}\Xi_j & \Delta_j \\ \Delta_j^T & \Upsilon_j\end{bmatrix}\begin{bmatrix}(GN_\xi)^T \\ N_\upsilon^T\end{bmatrix}. \tag{21}$$

Recall the noise characteristics of ω_j are given by $\omega_j \sim \mathcal{N}(0, \Omega_j)$. Both stochastic inputs, ω_j and μ_j, are trial-uncorrelated and assumed to be independent; that is, for $i, j \in \{0,\, 1,\, 2,\, \ldots\}$,

$$E\left[\omega_i \omega_j^T\right] = E\left[\mu_i \mu_j^T\right] = 0 \quad \text{if } i \neq j \tag{22}$$

and

$$E\left[\omega_i \mu_j^T\right] = 0 \quad \forall\, i, j\,. \tag{23}$$

Given the above framework, Equations (19)-(23), a standard Kalman filter process provides for each time step j an estimation $\widehat{d}_{j\,|\,j}$ of the modeling error d_j minimizing the error variance

$$P_{j\,|\,j} = E[(d_j - \widehat{d}_{j\,|\,j})\,(d_j - \widehat{d}_{j\,|\,j})^T]\,. \tag{24}$$

Here, $\widehat{d}_{j\,|\,i}$ denotes the estimate of the modeling error d_j at iteration j taking measurements $y_0,\, y_1,\, \ldots,\, y_i$ up to and including iteration $i,\ i \leq j$ into account. The matrix $P_{j\,|\,i} \in \mathbb{R}^{(N+1)n_\mathrm{x} \times (N+1)n_\mathrm{x}}$

represents the error variance at iteration j given observations up to and including time i, $i \leq j$.

For our specific problem, the Kalman filter equations read as:

$$\left\{\begin{array}{lcl} P_{0\,|\,0} & = & P_0 \\ P_{j\,|\,j-1} & = & P_{j-1\,|\,j-1} \,+\, \Omega_{j-1} \\ \Theta_j & = & G\, P_{j\,|\,j-1}\, G^T \,+\, M_j \\ K_j & = & P_{j\,|\,j-1}\, G^T\, \Theta_j^{-1} \\ P_{j\,|\,j} & = & (I - K_j G)\; P_{j\,|\,j-1}\,, \end{array}\right. \tag{25}$$

where $I \in \mathbb{R}^{(N+1)n_{\mathrm{x}} \times (N+1)n_{\mathrm{x}}}$ represents the identity matrix.

Finally, with the optimal Kalman gain K_j, the modeling error estimate $\widehat{d}_{j\,|\,j}$ is calculated by

$$\left\{\begin{array}{ll} \widehat{d}_{0\,|\,0} = & \widehat{d}_0 \\ \widehat{d}_{j\,|\,j} = & \widehat{d}_{j-1\,|\,j-1} \;+ \\ & K_j \left(y_j - G\widehat{d}_{j-1\,|\,j-1} - (GF+H)\, u_j\right). \end{array}\right. \tag{26}$$

Note that, given the Kalman gain K_j, the estimate of the model error at time j, $\widehat{d}_{j\,|\,j}$, is obtained by updating the previous estimate $\widehat{d}_{j-1\,|\,j-1}$, taking the actual measurement y_j into account. Moreover, especially noteworthy is that besides the previously introduced noise characteristics, $\mu_j \sim \mathcal{N}(0, M_j)$ and $\omega_j \sim \mathcal{N}(0, \Omega_j)$, two additional design parameter are inherent in the Kalman filter process, Equations (25) and (26). First, an initial value $\widehat{d}_{0\,|\,0}$ has to be defined. Most of the time, $\widehat{d}_{0\,|\,0} = \widehat{d}_0 = 0$ is a reasonable first guess. Second, with the starting value $P_{0\,|\,0} = E[(d_0 - \widehat{d}_0)\,(d_0 - \widehat{d}_0)^T]$, the initial error variance is specified. Choosing P_0 to be a diagonal matrix with large positive elements on the diagonal, results in larger changes of $\widehat{d}_{j\,|\,j}$ at the beginning of the learning.

The obtained error estimate $\widehat{d}_{j\,|\,j}$ enables us to improve the system's performance by designing an appropriate controller compensating for this error.

IV. Control

The proposed learning algorithm is completed by the subsequent control step. Making use of the information provided by the estimator, see Section III, a nonlinear model-based update rule is derived. A new input series $u_{j+1} \in \mathbb{R}^{(N+1)n_{\mathrm{u}}}$ is calculated in response to the estimated modeling error $\widehat{d}_{j\,|\,j}$.

The objective of the control step is to find an input u_{j+1}, which *optimally* compensates for the modeling error $\widehat{d}_{j\,|\,j}$. In the context of the dynamics (13), that means, minimizing

$$x_{j+1} \approx F u_{j+1} + \widehat{d}_{j\,|\,j}\,, \tag{27}$$

over all feasible u_{j+1}, $u_{\min} - u^* \leq u_{j+1} \leq u_{\max} - u^*$, cf. (18). In Equation (27), the disturbance ξ_{j+1}, see Equation (13), is neglected due to the fact that the stochastic variable ξ_{j+1} is not known a priori and has zero mean. Moreover, in Equation (27), the modeling error d_{j+1} is approximated by $\widehat{d}_{j\,|\,j}$. Recalling previous comments on the evolution and convergence of d_j, $j = 1, 2, \ldots$, see Section II, this approximation is a reasonable assumption. Taking the constraints on the system's state x_j and input u_j (18) explicitly into account, the update rule can be expressed by the following optimization problem:

$$\min_{u_{j+1}} \left\| F u_{j+1} + \widehat{d}_{j\,|\,j} \right\|_\ell \tag{28}$$

subject to

$$\begin{array}{rcl} u_{\min} & \leq & u_{j+1} \;\leq\; u_{\max} \\ x_{\min} & \leq & x_{j+1} \;\leq\; x_{\max}\,, \end{array} \tag{29}$$

where the future state x_{j+1} is approximated by (27).

The vector norm ℓ, $\ell \in \{1, 2, \infty\}$, of the minimization (28) affects the result and convergence of the learning algorithm and should be chosen in accordance with the performance objectives. For a vector $p = \left[p^{(1)}, p^{(2)}, \ldots, p^{(n_{\mathrm{p}})}\right]^T \in \mathbb{R}^{n_{\mathrm{p}}}$ the one norm, $\ell = 1$, the Euclidean norm, $\ell = 2$, and the maximum norm, $\ell = \infty$, are defined as

$$\|p\|_1 = \sum_{i=1}^{n_{\mathrm{p}}} |p^{(i)}|, \quad \|p\|_2 = \sqrt{p^T p}, \quad \|p\|_\infty = \max_{i \in \{1,\,2,\,\ldots,\,n_{\mathrm{p}}\}} |p^{(i)}|. \tag{30}$$

The update law defined in (28) and (29) can be expressed as a standard convex optimization problem of the following form, cf. [11]:

$$\min_z \left(\frac{1}{2} z^T V z + v^T z\right) \tag{31}$$

subject to

$$W z \leq w \quad \text{and} \quad \eta_1 \leq z \leq \eta_2\,, \tag{32}$$

where $z \in \mathbb{R}^{n_{\mathrm{z}}}$ represents the vector of decision variables. Vectors v, w and matrices V, W have appropriate dimensions.

In case of norms $\|\cdot\|_\ell$, $\ell \in \{1, \infty\}$, which are inherently nonlinear, non-quadratic functions, Equation (28) is re-formulated by extending the original vector of decision variables u_{j+1} and adding additional inequality constraints. Thus, in case of the *one norm*, Equation (28) is replaced by

$$\min_{u_{j+1},\,e} \; \mathbb{I}^T e \quad \text{subject to} \quad -e \leq F u_{j+1} + \widehat{d}_{j\,|\,j} \leq e\,, \tag{33}$$

where $e \in \mathbb{R}^{(N+1)n_{\mathrm{x}}}$ and $\mathbb{I}$ represent a vector of ones, $\mathbb{I} = [1,\, 1,\, 1,\, \ldots]^T \in \mathbb{R}^{(N+1)n_{\mathrm{x}}}$. Similarly, for the *maximum norm*, the extended equation reads as

$$\min_{u_{j+1},\,e} \; e \quad \text{subject to} \quad -e\,\mathbb{I} \leq F u_{j+1} + \widehat{d}_{j\,|\,j} \leq e\,\mathbb{I} \tag{34}$$

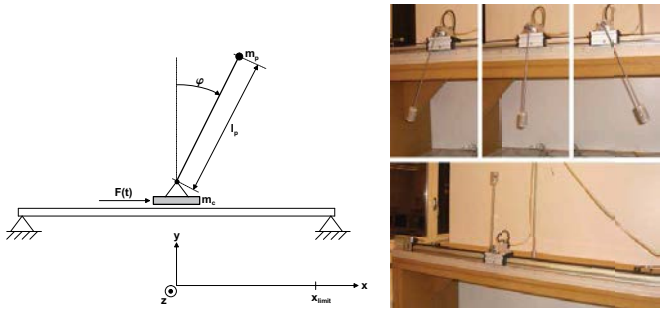
with $e \in \mathbb{R}$. In both cases, the constraints (29) still have to be satisfied. The *Euclidean norm* is minimized with the following equation

$$\min_{u_{j+1}} \left(F u_{j+1} + \widehat{d}_{j\,|\,j}\right)^T \left(F u_{j+1} + \widehat{d}_{j\,|\,j}\right). \tag{35}$$

Convex optimization problems of the form (31), (32) can be solved very efficiently and have the nice property that if the optimization problem is feasible, i.e., if there exist points $u_{j+1} \in \mathbb{R}^{(N+1)n_{\mathrm{u}}}$ satisfying the constraints (32), then there exists a local minimum, which is also globally optimal. In this paper, the commercial software CPLEX [19] is used which provides fast solutions for most large problems. Furthermore, formulating the update law as a convex optimization problem allows an explicit incorporation of input and state constraints. Such constraints are present in any existing real system and have a notable influence on the dynamic behavior of the system. The experimental setup chosen in Section V shows the particular importance of taking input and state constraints directly into account.

In view of the optimization (28), a scaling of the original signals $\mathrm{u}(t)$, $\mathrm{x}(t)$, $\mathrm{y}(t)$ in Equation (1) is indispensable to guarantee reasonable results. Aiming at equalizing the magnitude of the different physical quantities in the lifted domain, the scaling, exemplarily shown on the system's state $\mathrm{x}(t)$, reads as

$$x = S_x x^s\,, \quad S_x \in \mathbb{R}^{(N+1)n_{\mathrm{x}} \times (N+1)n_{\mathrm{x}}} \tag{36}$$



(a) Schematic drawing. (b) Experimental setup.

Fig. 2: The cart-pendulum system.

with x^s representing the scaled version of a lifted state vector x and S_x being the corresponding scaling matrix, usually represented by a diagonal matrix. In line with the previous notes in Section II, the introduced lifted domain representation allows us to modify the scaling along the trajectory. This can be effectively used in order to *weight* some parts of trajectory more than the other.

Besides choosing an appropriate norm for the minimization (28), a specific learning behavior can be particularly enforced by introducing an additional weighting of the entries in x_{j+1}, Equation (27). If, for example, a point-to-point motion is to be performed and the goal is to achieve the final point with high precision, the final part of the state vector x_{j+1} might be given significantly more weight. This is done in the experimental setup presented in Section V, where the objective of the learning algorithm is to swing-up the pendulum and reach the upright position.

V. Experiment: Swinging Up a Pendulum

The performance of the proposed technique is evaluated using the often cited problem of swinging up a pendulum which represents a benchmark problem involving the maintenance of balance, such as moving a robot's finger tip, stabilizing a walking robot, or controlling a rocket thruster. The huge number of publication on the inverted pendulum [20]–[24] show the importance and wide interest in this highly nonlinear, underactuated system. Applying a *purely open-loop* input, our proposed algorithm successfully learns the desired swing-up motion within only a few trials.

A. System, Model, and Constraints

A cart-pendulum system as schematically depicted in Figure 2(a) is chosen to prove the effectiveness of the proposed learning algorithm. In this system, the cart is moving along a rail of total length one meter driven by a DC motor through a belt which applies a force $F(t)$ in the x-direction. A pendulum is attached to the side of the cart by means of a pivot that allows the pendulum to freely swing in the (xy)-plane. The equations of motion of cart and pendulum are given in the following without a detailed derivation, cf. [25]:

$$\begin{aligned} \ddot{x} &= \frac{\frac{F}{m_p} - g\sin\varphi\cos\varphi + l_p\,\dot{\varphi}^2\sin\varphi}{\frac{m_c}{m_p} + \sin^2\varphi} \\ \ddot{\varphi} &= \frac{-\cos\varphi\frac{F}{m_p} + \left(\frac{m_c+m_p}{m_p}g - l_p\,\dot{\varphi}^2\cos\varphi\right)\sin\varphi}{l_p\left(\frac{m_c}{m_p} + \sin^2\varphi\right)}, \end{aligned} \tag{37}$$

where x is the position of the cart, φ is the pendulum angle, measured from the upright position, and F is the force applied to the cart. The definitions of the parameters and their values are given in Table I. In our experimental system, the force F is produced by the DC motor belt drive and can be modeled by

$$F = \alpha_1 u + \alpha_2 \dot{x}, \tag{38}$$

where u represents the input signal which is proportional to the voltage supplied to the motor. The values of the constants α_1 and α_2 of the motor used in our experimental apparatus are also given in Table I. Equations (37) and (38) can be reformulated as a system of first-order differential equations of the form (1) with $\mathrm{x}(t) = [x(t)\,\dot{x}(t)\,\varphi(t)\,\dot{\varphi}(t)]^T$. In our experimental setup, the position and angle are directly measured, whereas the velocities are obtained by numerical differentiation. In the context of Equation (1), that means $\mathrm{y}(t) = \mathrm{x}(t)$. The system represents a perfect test-bed for our proposed algorithm, since the input and the state are strongly limited due to the rail length and the physical limitations of the cart actuator:

$$|x| \le 0.5\ m, \quad |\dot{x}| \le 5\ m/s, \quad |u| \le 0.45\,. \tag{39}$$

Moreover, the above system description certainly represents only a simplified model of the real experimental setup, see Figure 2(b), where friction between the cart and the track highly influences the dynamic behavior. In particular the belt drive results in a position-dependent disturbance, which is very difficult to deal with when only performing open-loop control. Based on the cart-pendulum model given by (37), (38), and (39) a reference input trajectory is calculated swinging up the pendulum in the nominal case. In order to give the learning algorithm some space for improvement, the nominal input is obtained under the constraint $|u| \le 0.2$. This input, depicted by a solid black line in Figure 3, is sent to the experiment during the first trial. Note that, in order to protect the motor from high voltage changes, the input signal is filtered by a first-order Butterworth lowpass filter before being sent to the motor. Again, this is not modeled within the above's system description but learned by the real system while performing several iterations. A sampling time of 0.1 s is chosen. Performing the feedforward input over a time horizon of 1.86 s results in a lifted-domain representation of the state $\mathrm{x}(t)$ of dimension $4 * 186 = 744$.

TABLE I: Parameter values of the experimental setup.

	Description	Value
m_p	mass of the pendulum	175 g
m_c	mass of the cart	1.5 kg
l_p	distance from pivot to pendulum's center of mass	28 cm
α_1	motor constant 1 (voltage-to-force)	159 N
α_2	motor constant 2 (electrical resistance-to-force)	-22.5 Ns/m
g	gravitational constant	9.81 m/s^2

B. Learning Parameters and Experimental Results

The swing-up motion is very sensitive to modeling errors and other noise and disturbance influences. By simply applying the nominal input to the system, the pendulum hangs down at the end of the first trial, see Figure 4. The proposed learning algorithm is applied using the following parameters: The system's states $\mathrm{x}(t)$ are scaled in order to all lie in an interval between -1 and 1, cf. Equation (36). A weighting of the states is introduced. Only deviations in the position and angle are penalized in the control step, Section IV. Moreover, during the last second of the trial, the states' weights are multiplied by 100 emphasizing the performance objective of reaching the upright position. The matrix P_0 is assumed to be a diagonal matrix having values 2 on the diagonal and $\widehat{d}_0$ is assumed

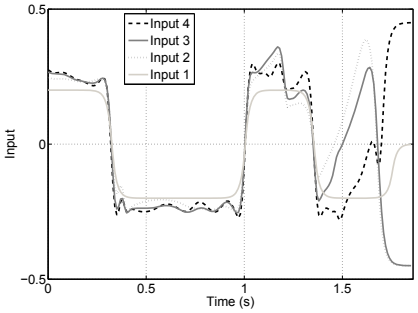


Fig. 3: Evolution of the input trajectories over several trials. Input 1 represents the nominal input.

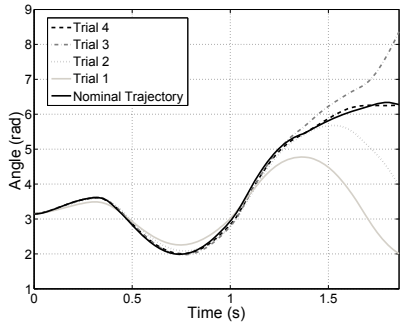


Fig. 4: The pendulum's angle over several trials. The solid line depicts the desired trajectory.

to be zero. In Equation (17), the value ϵ_j is set to $\epsilon_j = 0.3,\ \forall j$ and the covariance matrix M_j are diagonal with entries 0.1. Termination conditions are, in spite of the nonlinearity of the system, not defined.

In this setup, the swing-up is successfully performed in the fourth run. The fast convergence of the proposed learning algorithm, which is illustrated for the pendulum angle in Figure 4 and is also observed for other learning parameter sets. When using CPLEX, cf. [19], on a standard desktop computer (2GB RAM, 2.5GHz) through an interface in MATLAB, the calculation of an updated input trajectory takes less than 0.5 s.

VI. Conclusions and Outlook

In this paper, an optimization-based ILC approach was presented. Optimality is achieved in both the estimation of the modeling error and the following control step, which optimally compensates for the error by an updated input trajectory. While the first method is borrowed from classical control theory, the latter originates from mathematical optimization theory and uses a computationally efficient state-of-the art convex optimization solver. Input and state constraints are are explicitly taken into account incorporated. Depending on the problem under consideration, the overall learning behavior can be controlled by changing the optimization objective or by assigning different weights on different states and different parts of the trajectory. The applicability and reliability is proven by successfully mastering the sensitive problem of swinging up an underactuated pendulum using open-loop control only.

Convergence and robustness properties of the current algorithm are currently being investigated. In addition, an extending horizon approach was tested with encouraging results.

References

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

[2] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[3] H.-S. Ahn, K. L. Moore, and Y. Chen, *Iterative Learning Control: Robustness and Monotonic Convergence for Interval Systems (Communications and Control Engineering)*, 1st ed. Springer, 2007.

[4] M. Q. Phan and R. W. Longman, "A mathematical theory of learning control for linear discrete multivariable systems," in *Proceedings of the AIAA/AAS Astrodynamics Conference*, 1988, pp. 740–746.

[5] K. L. Moore, "Multi-loop control approach to designing iterative learning controllers," in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 1, 1998, pp. 666–671.

[6] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control using optimal feedback and feedforward actions," *International Journal of Control*, vol. 65, no. 2, pp. 277–293, 1996.

[7] K. s. S. Lee, J. Lee, I. Chin, J. Choi, and J. H. Lee, "Control of wafer temperature uniformity in rapid thermal processing using an optimal iterative learning control technique," *Industrial and Engineering Chemistry Research*, vol. 40, no. 7, pp. 1661–1672, 2001.

[8] M. Cho, Y. Lee, S. Joo, and K. S. Lee, "Semi-empirical model-based multivariable iterative learning control of an RTP system," *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 3, pp. 430–439, 2005.

[9] R. Tousain, E. van der Meche, and O. Bosgra, "Design strategy for iterative learning control based on optimal control," in *Proceedings of the 40th IEEE Conference on Decision and Control*, vol. 5, 2001, pp. 4463–4468.

[10] J. K. Rice and M. Verhaegen, "Lifted repetitive learning control for stochastic ltv systems: A structured matrix approach," *Automatica*, 2007, submitted.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] Y. Chen and C. Wen, *Iterative Learning Control: Convergence, Robustness and Applications (Lecture Notes in Control and Information Sciences)*, 1st ed. Springer, 1999.

[13] B. Bamieh, J. B. Pearson, B. A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[14] M. Norrlöf and S. Gunnarsson, "Time and frequency domain convergence properties in iterative learning control," *International Journal of Control*, vol. 75, no. 14, pp. 1114–1126, 2002.

[15] R. W. Longman, "Iterative learning control and repetitive control for engineering practice," *International Journal of Control*, vol. 73, no. 10, pp. 930–954, 2000.

[16] I. Chin, S. J. Qin, K. S. Lee, and M. Cho, "A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection," *Automatica*, vol. 40, no. 11, pp. 1913–1922, 2004.

[17] B. D. O. Anderson and J. B. Moore, *Optimal Filtering (Dover Books on Engineering)*. Dover Publications, 2005.

[18] C. K. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications (Springer Series in Information Sciences)*. Springer, 1998.

[19] ILOG, Inc., "ILOG CPLEX: High-performance software for mathematical programming and optimization," 2008, http://www.ilog.com/products/cplex/.

[20] F. L. Chernousko and S. A. Reshmin, "Time-optimal swing-up feedback control of a pendulum," *Nonlinear Dynamics*, vol. 47, no. 1, pp. 65–73, 2007.

[21] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing up control of inverted pendulum," in *Proceedings of the 1991 International Conference on Industrial Electronics, Control and Instrumentation*, vol. 3, 1991, pp. 2193–2198.

[22] K. Yoshida, "Swing-up control of an inverted pendulum by energy-based methods," in *Proceedings of the American Control Conference the 1999*, vol. 6, 1999, pp. 4045–4047.

[23] D. Chatterjee, A. Patra, and H. s. K. Joglekar, "Swing-up and stabilization of a cart-pendulum system under restricted cart track length," *Systems & control letters*, vol. 47, no. 4, pp. 355–364, 2002.

[24] M. Riedmiller, "Neural reinforcement learning to swing-up and balance a real pole," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 2005, pp. 3191–3196.

[25] S. A. Campbell, S. Crawford, and K. Morris, "Friction and the inverted pendulum stabilization problem," *Journal of Dynamic Systems, Measurement, and Control*, vol. 130, 2008.