Master Thesis 01

Parameter Tuning of Iterative Learning Control Using Reinforcement Learning for Human-Machine Cooperation

Matthias Hirche

14 March 2019

Prüfer: *Prof. Dr.-Ing. Frank Allgöwer*[*]
Betreuer: *Dr. Pierre-Jean Meyer*[†]
Prof. Dr. Murat Arcak[†]
M.Sc. Philipp N. Köhler[*]

[*]Institut für Systemtheorie und Regelungstechnik
Universität Stuttgart
Prof. Dr.-Ing. Frank Allgöwer

[†]Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Contents

Abstract

In this work, a procedure to apply iterative learning control to a nonlinear, continuous-time system is developed, where the control goal is the tracking of a reference trajectory. Iterative learning control updates the input signal to a system that performs a task repeatedly by using the gathered information of past completions of the task. The proposed procedure is motivated by the task of human-machine cooperation in the sense that a human has to learn how to collaborate with a machine. For example, a human with paraplegia needs to cooperate with their powered lower limb orthoses in order to stand up from the sitting position. The human learning is emulated by iterative learning control, therefore a simple proportional-derivative input update law is chosen. This update law features a proportional and a derivative gain, the learning gains, which have to be selected in each iteration of the learning process to facilitate learning. The learning gains are chosen by using reinforcement learning, selecting the gains that minimise a policy value function, which depends on the predicted tracking error and the change in the input signal and defines the performance of the system. Thus, the proposed procedure is as follows: In each iteration of the learning process, the open-loop input signal is applied to the system and the tracking error is recorded. Then, the learning gains are chosen such that the resulting updated input minimises the policy value function. Subsequently, the input signal is updated with the optimal learning gains and again applied to the system, yielding a new tracking error and the steps are repeated. For a specific choice of the policy value function, the tracking improves in each iteration if the optimal learning gains are non-zero. Additionally, if they are zero, the procedure can be stopped, because all future optimal gains will also be zero. To facilitate the use of second-order optimisation methods, conditions on the system and the initial input signal are provided which guarantee the existence and continuity of the first and second derivatives of the policy value function. The procedure is successfully tested on two examples related to the human-machine cooperation as described above.

Deutsche Kurzfassung

In dieser Arbeit wird ein Vorgehen vorgestellt, welches *iterative learning control* auf ein nichtlineares, zeitkontinuierliches System anwendet, mit dem Regelziel einer Referenztrajektorie zu folgen. *Iterative learning control* verändert das Eingangssignal eines Systems, das eine Aufgabe mehrmals ausführt, mithilfe von Informationen aus vorangehenden Ausführungen. Das vorgeschlagene Vorgehen ist motiviert durch eine Mensch-Maschine Kooperation in welcher ein Mensch mit einer Maschine zusammenarbeiten muss. Zum Beispiel muss ein eine Orthese tragender, querschnittgelähmter Mensch mit dieser kooperieren, um aus dem Sitzen aufstehen zu können. Das menschliche Lernen wird durch *iterative learning control* ersetzt, weshalb ein einfaches proportionales und differenzierendes Aktualisierungsgesetz für das Eingangssignal gewählt wird. Dieses Aktualisierungsgesetz beinhaltet einen proportionalen und einen differenzierenden Koeffizienten, die Lernkoeffizienten, welche in jeder Iteration angepasst werden müssen, um einen Lernprozess zu ermöglichen. Diese Lernkoeffizienten werden durch *reinforcement learning* gewählt, so dass eine Kostenfunktion minimiert wird, welche von dem prädizierten Folgefehler und der Änderung im Eingangssignal abhängt. Das Vorgehen ist wie folgt: In jeder Iteration des Lernprozesses wird das System mit dem Eingangssignal angesteuert und der resultierende Folgefehler aufgenommen. Dann werden die Lernkoeffizienten so gewählt, dass sie die Kostenfunktion minimieren und das Eingangssignal wird damit aktualisiert. Dieses wird dann für den nächsten Durchlauf verwendet und die Schritte werden wiederholt. Für eine bestimmte Wahl der Kostenfunktion verbessert sich das Folgeverhalten des Systems in jeder Iteration solange die optimalen Lernkoeffizienten nicht verschwinden. Sollten sie jedoch null sein, kann das Vorgehen angehalten werden, weil alle darauf folgenden Lernkoeffizienten ebenfalls verschwinden werden. Um den Einsatz von Optimierungsmethoden zweiter Ordnung zu ermöglichen, werden Voraussetzungen an das System und das initialisierende Eingangssignal gestellt, welche die Existenz und Stetigkeit der ersten und zweiten Ableitungen der Kostenfunktion garantieren. Das Vorgehen wird erfolgreich an zwei Beispielen getestet, die im Rahmen der oben beschriebenen Mensch-Maschine-Kooperation liegen.

1 Introduction

In our everyday lives, every human has to interact with a machine at some point. Sometimes, however, mere interaction, like telling a machine what to do, is not enough, sometimes a full cooperation between human and machine acting simultaneously is necessary. Take for example a driver who has to drive a car together with embedded control systems or alternatively a paraplegic who has to cooperate with his orthoses in order to walk or perform otherwise seemingly basic actions. This raises the question of whether a human is able to learn how to collaborate with such a device. For humans with paraplegia the simple task of standing up from a sitting position poses a challenge and this sit-to-stand motion has to be learned again by the user of powered orthoses [25]. A study with humans, however, would go beyond the scope of this work and a suitable alternative which acts as a substitute for the human needs to be found. Narvaez-Aroche et al. investigate this cooperation for powered lower limb orthoses and use iterative learning control (ILC) as an alternative for human learning. This is reasonable, because ILC was motivated by human learning in the first place and is also an approximation of the learning process [5]. Many of the available results in ILC are in the domain of linear or nonlinear, discrete-time systems [11]. Yet we all live in a continuous-time and nonlinear world. Hence, an ILC scheme is needed that works for nonlinear, continuous-time systems and appropriately represents human learning. In ILC, the input to the system is updated iteratively and if this update is parameterised it may approximate the human learning process [5]. It is, however, unclear how these parameters can be chosen such that a human-machine cooperation can be emulated for nonlinear, continuous-time systems. This is provided by tuning the parameters of ILC using reinforcement learning (RL) in such a way as that the future performance is improved and ultimately leads to the proposed procedure that, in general, applies ILC to nonlinear, continuous-time systems, whereas it is usually applied to nonlinear, discrete-time systems. This human-machine cooperation can be described as follows: Consider a general nonlinear system in continuous time which has at least two, or without loss

of generality, exactly two inputs

$$\dot{x}(t) = f(x(t), u_1(t), u_2(t)), \tag{1.1}$$

for which a controller that uses both input channels $u_1(t) = K_1(x(t))$ and $u_2(t) = K_2(x(t))$ to complete a specific task has been designed, but only one of the two is actually implemented, leaving the second input open, i.e.

$$\dot{x}(t) = f(x(t), K_1(x(t)), u_2(t)). \tag{1.2}$$

This second input $u_2(t)$ is now assumed by a human or as a substitute by an iteratively learning controller which learns how to successfully cooperate with the partially implemented controller $K_1(x(t))$ such that the task can be completed. This leads to the following, more general, problem statement.

1.1 Problem Statement

Given is the following nonlinear and continuous-time system

$$\dot{x}(t) = f(x(t), u(t)), \tag{1.3a}$$
$$y(t) = h(x(t)), \tag{1.3b}$$
$$x(t_0) = x_{\text{IC}}, \tag{1.3c}$$
$$x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad y \in \mathbb{R}^q \tag{1.3d}$$

that tracks a reference trajectory $y_d(t) \in \mathbb{R}^q$ for $t \in [t_0,\ t_f]$. This tracking is done iteratively, always starting in the same initial position (1.3c) and generates the tracking error

$$e_j(t) = y_j(t) - y_d(t) \tag{1.4}$$

in iteration j. The goal is to use parameter-tuned ILC, i.e. update the input signal in each iteration j according to

$$u_{j+1}(t) = u_j(t) + \Lambda_{j+1} \begin{bmatrix} e_j(t) \\ \dot{e}_j(t) \end{bmatrix} \quad \forall t \in [t_0,\ t_f] \tag{1.5}$$

to improve the performance of the tracking, that is the entire input signal $u_j(t)$ is updated in each iteration. The performance is defined by the policy value ρ_j^*, which is the cost (or reward) in RL, where for this case the lesser

the policy value the better the performance. The learning gain Λ_{j+1} is then chosen by solving

$$\Lambda_{j+1}^{*} = \arg\min_{\Lambda_{j+1}} \rho_j(\Lambda_{j+1}). \tag{1.6}$$

Before a brief outline of the upcoming chapters is given, some notation is defined.

1.2 Mathematical Notation

The following notation for various norms is used:

- For a vector $v \in \mathbb{R}^n$ with elements v_i the infinity norm is defined as

$$\|v\|_\infty := \max_{1 \le i \le n} |v_i|. \tag{1.7}$$

  For a matrix $M \in \mathbb{R}^{m \times n}$ with elements m_{ij} the infinity norm is defined as

$$\|M\|_\infty := \max_{1 \le i \le m} \sum_{j=1}^{n} |m_{ij}|. \tag{1.8}$$

- For a vector-valued function $f(t) \in \mathbb{R}^n$ with elements $f_i(t)$ defined on the interval $t \in [t_0, t_f]$ and a positive scalar $\lambda > 0$, the lambda norm is defined as

$$\|f\|_\lambda := \sup_{t_0 \le t \le t_f} \left\{ e^{-\lambda t} \|f(t)\|_\infty \right\}. \tag{1.9}$$

- For a vector $v \in \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{n \times n}$ the matrix-weighted norm is defined as

$$\|v\|_M := \sqrt{v^T M v}. \tag{1.10}$$

- For a function $f(t)$ whose squared absolute value is Lebesgue integrable over the interval $t \in [a, b]$, the L_2-norm is defined as

$$\|f\|_2 := \left(\int_a^b |f(t)|^2 \, \mathrm{d}t \right)^{1/2} < \infty. \tag{1.11}$$

The natural numbers with 0 included are denoted as

$$\mathbb{N}^0 = \mathbb{N} \cup \{0\}. \tag{1.12}$$

The derivative of the function $f(x) : \mathbb{R}^n \to \mathbb{R}^m$ with respect to the vector $x \in \mathbb{R}^n$ is the $m \times n$ matrix

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \ddots & & \vdots \\ \vdots & & & \\ \frac{\partial f_m}{\partial x_1} & \cdots & & \frac{\partial f_m}{\partial x_n} \end{bmatrix}. \tag{1.13}$$

1.3 Outline

The concept of ILC is covered in the next chapter with an introduction to it and some selected results. In the third chapter, ILC is combined with RL to appropriately tune its parameters for nonlinear, continuous-time systems. The presented learning scheme is subsequently applied to two examples in the fourth chapter and the work is summarised in the fifth chapter. Finally, some auxiliary, additional information is provided in the appendix.

2 Iterative Learning Control

Before the approach on how to use ILC for nonlinear, continuous-time system is introduced, a familiarity with ILC is beneficial. This chapter is about the concept of ILC, giving a short introduction to it in the first section, presenting some selected results and insights in the second section and closing the chapter in the third section with a closer look on norm-optimal ILC, which is a concept with particular relevance to the next chapter.

2.1 Introduction to Iterative Learning Control

When a system repeats the same task over and over again, it only seems reasonable to use information gathered in past iterations to improve its performance. This gives rise to ILC which improves the performance of a system by learning from past iterations. The basic idea of ILC was described earliest in a US patent filed in 1967 [24]. Nonetheless, it only received general interest after publications in 1984 [8, 9], in which Arimoto et al. propose an iterative "betterment process" to improve the tracking of a reference trajectory for a robotic manipulator that performs a task recurrently.

In principle, ILC is closely related to human learning [5] and was motivated by it [9]. For example, a tennis player observes the trajectory of the ball after hitting it and subsequently adjusts his hitting motion if the ball struck the net. Another example is a football player slightly changing her shooting technique after missing the goal by an inch. In both examples repetition and learning from past mistakes enables the players to learn a suitable movement profile, saved in their (muscle) memory. This motion is principally a learned open-loop signal, which can be applied to achieve a certain goal.

ILC is similar; a given input signal is updated after each repetition using the error signal of last iterations until it converges to a signal that leads to a good performance. Essentially, ILC is a model-free open loop control strategy. This makes it especially useful in applications in which unmodelled dynamics or uncertain parameters prevent the desired performance of a controlled system. Of course, those applications have to satisfy certain

conditions, e.g. perform the same task repeatedly. Since ILC is a feedforward controller, it is normally used with a well-designed feedback controller.

ILC is applied arguably the most in the area of robotics [6, 8–10, 12, 13, 15–18, 21, 31, 32, 34, 35], although it also finds application in rotary systems [23, 30, 36], chemical processes [14, 37], bioengineering [5, 19] and many more [1].

In general, ILC can be applied to systems that repeat a task starting in the same initial value in each trial. This resetting of the initial condition is the main difference to repetitive control in which the initial condition is the final value of the previous iteration [11].

Because of its close relationship to human learning, ILC is a good pick to emulate the very same. Arif and Inooka [5] have, in fact, experimentally established a relationship between ILC and human learning. This corroborates using the former in substituting the human in the human-machine cooperation, as is done in [26].

2.2 Selected Results in Iterative Learning Control

Even though ILC is often used in discrete-time systems [11], the focus lies here on the continuous-time case. The general concept in continuous-time ILC is as follows: Consider a system

$$\dot{x}(t) = f(x(t), u(t)), \tag{2.1a}$$
$$y(t) = h(x(t), u(t)) \tag{2.1b}$$

with

$$x(t) \in \mathbb{R}^n, \quad u(t) \in \mathbb{R}^m, \quad y(t) \in \mathbb{R}^q.$$

The control goal is to track a certain reference trajectory $y_d(t) \in \mathbb{R}^q$ over the finite time interval $t \in [t_0, t_f]$; this is done repeatedly, starting in each iteration j from the same initial position

$$x_j(t_0) = x_{\text{IC}} \quad \forall j \in \mathbb{N}^0 \tag{2.2}$$

and a tracking error can be defined as

$$e_j(t) \coloneqq y_j(t) - y_d(t). \tag{2.3}$$

The index j adds the additional iteration domain; together with the time domain this makes the problem at least 2-dimensional. The initial reset (2.2)

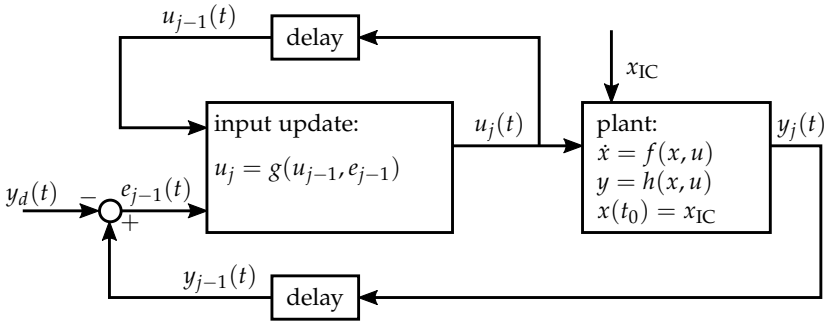

Figure 2.1: Block diagram of the general ILC concept. The input $u_j(t)$ to the plant in iteration j is generated by updating the previous input $u_{j-1}(t)$ with the error $e_{j-1}(t)$ of the previous iteration.

is a pivotal assumption of ILC; naturally some publications explore to weaken it, e.g. [28] or [33].

At the start of the process, the initial input $u_0(t)$, usually chosen by the user, generates an initial error $e_0(t)$. Then after each iteration j the new input signal $u_{j+1}(t)$ is determined through an input update law which is normally a function of the previous error

$$u_{j+1}(t) = g(u_j(t), e_j(t)). \tag{2.4}$$

The general concept of ILC is depicted in Figure 2.1. Unsurprisingly, certain requirements have to be imposed on the general formulation (2.1) – (2.4) to obtain any results regarding the behaviour of the ILC.

The aforementioned 2-dimensionality, that is the time and iteration domains, of the ILC problem greatly increases the difficulty in analysing it. Usually, the following properties are desired:

- Asymptotic stability in terms of ILC [1, 11], i.e. a bounded and convergent input signal sequence

$$\begin{aligned} \|u_j(t)\| \leq \bar{u} < \infty \quad & \forall t \in [t_0, t_f], \quad \forall j \in \mathbb{N}^0, \\ \lim_{j\to\infty} u_j(t) = u_\infty(t) \quad & \forall t \in [t_0, t_f]. \end{aligned}$$

- Convergence of the error [1, 11]

$$\lim_{j\to\infty} e_j(t) = e_\infty(t) \quad \forall t \in \left[t_0, t_f\right].$$

- A good transient learning behaviour, typically a monotonically converging error [11]

$$\|e_j(t)\| \leq \|e_{j-1}(t)\| \quad \forall j \in \mathbb{N}.$$

- Robustness typically related to uncertainty in the plant dynamics [11].

In [8] the linear time-invariant system

$$\dot{x}_j(t) = Ax_j(t) + Bu_j(t), \tag{2.5a}$$
$$y_j(t) = Cx_j(t) \tag{2.5b}$$

is considered. It is assumed that

$$y_j, u_j \in \mathbb{R}^r$$

and

$$\operatorname{rank}(CB) = r, \tag{2.6}$$

that is the matrices C and B have full rank. Moreover, it is explicitly not assumed that the system matrices are perfectly known, only that they are constant and that the system is initialised with the same initial value in each iteration. Then, Arimoto et al. propose a rule to update the input

$$u_{j+1}(t) = u_j(t) - \Gamma\dot{e}_j(t) \tag{2.7}$$

with the time derivative of the error (2.3) and an arbitrary constant $r \times r$ matrix Γ. Using this update law they show that under the assumptions

1. $\|I_r - CB\Gamma\|_\infty < 1$,
2. $u_0(t)$ is continuous and $y_d(t)$ is continuously differentiable on $t \in [0, T]$,

where I_r is the $r \times r$ identity matrix, there exist constants $\lambda > 0$ and $0 \leq \rho < 1$ such that

$$\|\dot{e}_{j+1}\|_\lambda \leq \rho\|\dot{e}_j\|_\lambda \quad \forall j \in \mathbb{N}^0. \tag{2.8}$$

A similar result guaranteeing that the output trajectory converges to the reference trajectory holds for a class of nonlinear systems given by

$$\dot{x}_j(t) = f(t, x_j(t)) + Bu_j(t), \tag{2.9a}$$
$$y_j(t) = Cx_j(t) \tag{2.9b}$$

with the same update law (2.7) and under certain conditions as outlined in [8].

In a later publication [7], Arimoto proposes an enhanced input update law

$$u_{j+1}(t) = u_j(t) - \Gamma\dot{e}_j(t) - \Phi e_j(t) - \Psi \int e_j(t)\,\mathrm{d}t, \tag{2.10}$$

with a proportional, integrating and differentiating (PID) part, calling it a "PID-type iterative algorithm". He then shows that for a class of linear time-invariant systems it is enough to only use a suitable proportional learning gain, that is $\Gamma = \Psi = 0$, to guarantee that the output $y_j(t)$ converges to the reference $y_d(t)$ for each fixed $t \in [t_0, t_f]$ as $j \to \infty$. In addition, Arimoto proves analogous convergence theorems with the proportional and differentiating (PD) update law

$$u_{j+1}(t) = u_j(t) - \Gamma\dot{e}_j(t) - \Phi e_j(t) \tag{2.11}$$

if certain conditions, as outlined in [7], are satisfied. This PD-type input update law finds application especially for nonlinear systems [11], for example, for the control of an electrohydraulic injection molding machine [20] or of robotic manipulators [7, 9, 10, 12].

For all the above mentioned results on convergence, the choice of the learning gains is critical. They have to be selected with respect to the system dynamics and it is not immediately clear what a particularly good pick would be. One way to circumvent this decision is provided by norm-optimal iterative learning control.

2.3 Norm-Optimal Iterative Learning Control

In norm-optimal iterative learning control [2, 4, 27] the objective of ILC, i.e. reducing the tracking error, is embedded into a cost criterion

$$J_{j+1}(u_{j+1}(t)) = \|e_{j+1}(t)\|_2^2 + \|u_{j+1}(t) - u_j(t)\|_2^2 \tag{2.12}$$

comprising the norm of the future error and the change in the input signal. This norm is induced by the inner product of the Hilbert spaces containing the output and input. In continuous time, usually the L_2-norm is chosen. Consider a general linear system model expressed as

$$y_{j+1}(t) = Gu_j(t) + d_j(t) \tag{2.13}$$

where G is a bounded linear operator mapping the input space to the output space and $d_j(t)$ represents initial conditions and other influences in iteration j [27]. Then, the input in the next trial $u_{j+1}(t)$ is computed by minimising the cost

$$u_{j+1}(t) = \arg\min_u {}_{j+1}(t)J_{j+1} \tag{2.14}$$

and the optimal control input in the next iteration is

$$u_{j+1}(t) = u_j(t) + G^* e_{j+1}(t) \tag{2.15}$$

where G^* denotes the adjoint operator of G [2].

More explicitly, let G be given in state-space form

$$\begin{aligned} \dot{x}_j(t) &= Ax_j(t) + Bu_j(t), \\ y_j(t) &= Cx_j(t), \\ e_j(t) &= y_d(t) - y_j(t) \end{aligned}$$

with $t \in [0, T]$ and $x_j(0) = 0$ and the cost as

$$\begin{aligned} J_{j+1} = \int_0^T & e_{j+1}(t)^T Q e_{j+1}(t) + (u_{j+1}(t) - u_j(t))^T R(u_{j+1}(t) - u_j(t))\,\mathrm{d}t \\ & + e_{j+1}^T(T) F e_{j+1}(T). \end{aligned} \tag{2.16}$$

The seemingly non-causal update (2.15) can be realised using linear quadratic regulator (LQR) theory [3]. The resulting input, as outlined in [3], is

$$u_{j+1}(t) = u_j(t) + R^{-1}B^T \left(K(t)(x_j(t) - x_{j+1}(t)) + \xi_{j+1}(t) \right) \tag{2.17}$$

with the feedback gain matrix $K(t)$ as the solution of the Riccati differential equation

$$\dot{K}(t) = -A^T K(t) - K(t)A + K(t)BR^{-1}B^T K(t) - C^T Q C, \tag{2.18a}$$

$$K(T) = C^T F C \tag{2.18b}$$

and the predictive filter $\xi_{j+1}(t)$

$$\dot{\xi}_{j+1}(t) = -\left(A - BR^{-1}B^T K\right)^T \xi_{j+1}(t) + C^T Q e_j(t), \tag{2.19a}$$

$$\xi_{j+1}(T) = -C^T F e_j(T). \tag{2.19b}$$

If, however, the transition into nonlinear systems is made, LQR theory does no longer apply and finding an explicit input equation like (2.17) is considerably harder. This gives rise to a modification of the optimisation problem. Instead of minimising the cost (2.12) with respect to the input signal $u_{j+1}(t)$, it is minimised with respect to the learning gains in the input update law (2.10). Owens calls this approach "Parameter Optimal Iterative Control" in his book [27] and discusses its properties in the linear case. In the following, the focus lies on the nonlinear case and on using an appropriate input update law that can reasonably resemble human learning.

3 Iterative Learning Control with Parameter Tuning

After reviewing past results in ILC, a new approach to apply ILC to nonlinear, continuous-time systems is proposed, which can also be used to examine a possible collaboration between a human agent and a controller. This procedure uses RL to tune the learning gains in a parameterised input update law. In the first section, the framework is constructed and the problem is cast in the light of RL. Moreover, some assumptions on human learning are set out. The second section focuses on the policy value function, a cost that is minimised in each iteration to find the learning gain, and its properties. In the third section, the first and second derivatives of the policy value function are calculated and conditions on the plant are imposed for them to exist and be continuous.

3.1 Framework and General Assumptions on Human Learning

Conceptually, ILC is a form of reinforcement learning. In each trial the agent, i.e. the controller, selects a policy. After executing this policy, the agent gathers some kind of feedback, called the reward or reinforcement [29] and using this reinforcement, the agent then chooses the next policy. In ILC this policy is the input signal for an iteration and the reward is, for example, the resulting norm of the error. A function ρ which defines this reward is called the policy value function [29]. In principle, ILC is then a kind of policy search and Russel and Norvig [29] suitably describe the idea of both ILC and policy search: "the idea is to keep twiddling the policy as long as its performance improves, then stop."

Before an appropriate policy is stated, the environment has to be defined. From now on the plant is assumed to be from a class of deterministic,

nonlinear systems described by

$$\dot{x}(t) = f(x(t), u(t)), \tag{3.1a}$$
$$y(t) = h(x(t)), \tag{3.1b}$$
$$x(t_0) = x_{\text{IC}}, \tag{3.1c}$$
$$x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m, \quad y \in \mathbb{R}^q. \tag{3.1d}$$

The objective is to track a defined reference trajectory $y_d(t)$ over the finite time interval $t \in [t_0, t_f]$, generating a tracking error

$$e(t) = y(t) - y_d(t). \tag{3.2}$$

This is done iteratively while changing the input $u_j(t)$ in each iteration j using ILC. It is assumed that the initially applied input $u_0(t)$ generates a bounded error signal $e_0(t) = y_0(t) - y_d(t)$. In addition, the initial condition is the same in each iteration, that is $x_j(t_0) = x_{\text{IC}}\ \forall j \in \mathbb{N}^0$.

The input update law is the policy the agent adopts between each trial, and it is parameterised using a slight adaption of the familiar rule (2.7) of Arimoto et al. [8]

$$u_{j+1}(t) = u_j(t) + \Gamma_{j+1} e_j(t). \tag{3.3}$$

In (3.3) the learning gain $\Gamma_{j+1} \in \mathbb{R}^{m \times q}$ may change from iteration to iteration. This is because here ILC is used to substitute a human operator and it is intuitive that a human would focus on different elements of the gathered information from trial to trial. One might think of a tennis player focusing first on the proper hitting spot on the racket for some tries before the focus shifts on errors in the applied force. This assumption is further corroborated by Arif and Inooka who state that the human "modifies the gains in each iteration" [5].

The learning gain Γ_j is essentially what constitutes the policy in each iteration and its choice should therefore depend on a suitable policy value function $\rho \geq 0$, i.e. the reward or cost resulting from using this policy. Here, the policy value function is chosen as the cost-to-go, that is a measure of the system's deviation from the reference trajectory after updating the input with the learning gain. Then the best policy is the one resulting in the smallest policy value

$$\rho_j^* = \min_{\Gamma_j} \rho(\Gamma_j), \tag{3.4}$$

which is the optimal cost-to-go. The assumption is that if a learning gain which improves the policy value is found, then a human is able to improve and learn from the last try as well.

Instead of only using the information provided in the error signal in the update rule (3.3), more could be included. As it turns out, a human also learns from the information encompassed in the derivative of the error [5]. This leads to the familiar PD-type update law (2.11)

$$\begin{aligned} u_{j+1}(t) &= u_j(t) + \Gamma_{j+1} e_j(t) + \Delta_{j+1} \dot{e}_j(t) \\ &= u_j(t) + \begin{bmatrix} \Gamma_{j+1} & \Delta_{j+1} \end{bmatrix} \begin{bmatrix} e_j(t) \\ \dot{e}_j(t) \end{bmatrix} \\ &= u_j(t) + \Lambda_{j+1} \tilde{e}_j(t) \end{aligned} \tag{3.5}$$

with the learning gain matrix $\Lambda_{j+1} \in \mathbb{R}^{m \times 2q}$ and the extended error $\tilde{e}_j$.

It is immediately clear that this extended update law can be reduced to (3.3) by choosing $\Delta_{j+1} = 0$ as the zero matrix. Thus, the optimal policy value in each iteration can only be better, i.e. smaller, if (3.5) is used instead of (3.3).

In summary, the problem is

$$\rho^*_{j+1} = \min_{\Lambda_{j+1}} \rho(\Lambda_{j+1}) \tag{3.6a}$$

subject to

$$j \in \mathbb{N}^0, \quad t \in \left[t_0, t_f\right], \tag{3.6b}$$

$$u_{j+1}(t) = u_j(t) + \Lambda_{j+1} \tilde{e}_j(t) = u_j(t) + \Lambda_{j+1} \begin{bmatrix} e_j(t) \\ \dot{e}_j(t) \end{bmatrix}, \tag{3.6c}$$

$$\dot{x}_{j+1}(t) = f(x_{j+1}(t), u_{j+1}(t)), \tag{3.6d}$$

$$y_{j+1}(t) = h(x_{j+1}(t)), \tag{3.6e}$$

$$e_{j+1}(t) = y_{j+1}(t) - y_d(t) \tag{3.6f}$$

and the main assumptions about human learning are:

- Human learning can be emulated by ILC.
- In each iteration a human applies an individual learning gain.
- If a learning gain that reduces the policy value, i.e. leads to a better performance, can be found, a human will also be able to perform better in the next iteration.
- A human learns from both the error and its derivative.

In Section 2.3, the optimisation problem (2.14)

$$u_{j+1} = \arg\min_{u_{j+1}} J_{j+1}$$

was solved for a linear system. Unfortunately, it is not as straightforward to find a solution for this problem for a nonlinear system. The parameterised input update law (3.6c), however, offers the advantage that the dimensional complexity of the optimisation problem (3.6a) is reduced to the dimensions of the learning gain Λ_{j+1}. While the policy value function ρ is an arbitrary cost, a closed form will result in a standard optimisation problem [29] and the next section will make it explicit.

3.2 Policy Value Function

From now on, the general time argument will be dropped for the sake of brevity whenever possible. Nevertheless, it is worth noting that all time signals are in continuous time and, for example, u_j denotes the continuous-time input signal in iteration j.

Although the policy value function can be defined arbitrarily, the following LQR resembling cost, which is also used in the norm-optimal ILC scheme presented in Section 2.3, will be the policy value function

$$\begin{aligned}\rho_{j+1}(\Lambda_{j+1}) = \int_{t_0}^{t_f} e_{j+1}^T E e_{j+1} + (u_{j+1} - u_j)^T U (u_{j+1} - u_j)\,\mathrm{d}t \\ + e_{j+1}^T(t_f) T e_{j+1}(t_f)\end{aligned} \tag{3.7}$$

where $0 \leq E \in \mathbb{R}^{q\times q}$, $0 \leq U \in \mathbb{R}^{m\times m}$ and $0 \leq T \in \mathbb{R}^{q\times q}$ are positive semi-definite weighting matrices. This allows individual weighting of error components, control over how much the input signal is changed in each iteration and an additional emphasis on the terminal error.

The approach is now as follows:

1. In iteration j, apply the input u_j.
2. Solve the optimisation problem (3.6)

   $$\rho_{j+1}^* = \min_{\Lambda_{j+1}} \rho(\Lambda_{j+1})$$

   to determine the learning gain

   $$\Lambda_{j+1}^* = \arg\min_{\Lambda_{j+1}} \rho(\Lambda_{j+1}).$$

3. Update the input according to the input update law (3.5)

$$u_{j+1} = u_j + \Lambda_{j+1}^* \tilde{e}_j.$$

4. Proceed to the next iteration (increment j and go to step 1).

This procedure has the following properties:

P1: The policy values are a monotonically decreasing sequence

$$\rho_{j+1}^* \leq \rho_j^* \quad j \in \mathbb{N}^0. \tag{3.8}$$

P2: The following implications hold:

$$\Lambda_j^* = 0 \Rightarrow \Lambda_{j+1}^* = 0 \text{ and } \rho_{j+1}^* = \rho_j^* \quad \forall j \in \mathbb{N} \tag{3.9}$$

and additionally

$$\Lambda_j^* \neq 0 \text{ and } U > 0 \Rightarrow \rho_{j+1}^* < \rho_j^* \quad \forall j \in \mathbb{N}, \tag{3.10a}$$

$$\Lambda_{j+1}^* \neq 0 \text{ and } U = 0 \Rightarrow \rho_{j+1}^* < \rho_j^* \quad \forall j \in \mathbb{N}^0. \tag{3.10b}$$

P3: The policy values converge to a lower limit

$$\lim_{j \to \infty} \rho_j = \rho_\infty \geq 0. \tag{3.11}$$

A consequence of P2 is that as soon as the solution of the optimisation problem (3.6) is the zero matrix, the procedure can be stopped, because the policy value cannot be improved in subsequent iterations. This is intuitive, because no new information is gathered by repeating the same input. The difference between (3.10a) and (3.10b) is because $\Lambda_j = 0$ with $U > 0$ always reduces the policy value, that is $\rho_j(\Lambda_j = 0) < \rho_{j-1}^*$, by not changing the input signal and is only a technicality.

In the following proofs, the shortened notation of the matrix-weighted norm is used, its definition is given in Section 1.2. In addition, $e_j^* = e_j(\Lambda_j^*)$ denotes the error generated by the input $u_j^* = u_j(\Lambda_j^*)$ which was obtained by updating with the optimal learning gain Λ_j^*.

Proof of P1: The difference between two consecutive policy values can be estimated by using the fact that an optimal policy is at least as good as an arbitrary one:

$$\begin{aligned}
\rho^*_{j+1} - \rho^*_j &\leq \rho_{j+1}(\Lambda_{j+1} = 0) - \rho^*_j \\
&= \int_{t_0}^{t_f} \|e_{j+1}(\Lambda_{j+1} = 0)\|_E^2 + \| \underbrace{u_{j+1} - u^*_j}_{= \Lambda_{j+1}\big|_{=0}\tilde{e}^*_j = 0} \|_U^2 \,\mathrm{d}t + \|e_{j+1}(\Lambda_{j+1} = 0, t_f)\|_T^2 \\
&\quad - \int_{t_0}^{t_f} \|e^*_j\|_E^2 + \|u^*_j - u^*_{j-1}\|_U^2 \,\mathrm{d}t - \|e^*_j(t_f)\|_T^2 \\
&= - \int_{t_0}^{t_f} \|u^*_j - u^*_{j-1}\|_U^2 \,\mathrm{d}t \\
&\leq 0.
\end{aligned}$$

Note that if $\Lambda_{j+1} = 0$ is chosen, no update takes place and the error signal does not change, that is $e_{j+1}(\Lambda_{j+1} = 0) \equiv e^*_j$.

Proof of P2: In order to show (3.9), let $\Lambda^*_j = 0$ be the solution of (3.6) after iteration $j - 1$ which is

$$\begin{aligned}
\Lambda^*_j &= \arg\min_{\Lambda_j} \int_{t_0}^{t_f} \|e_j\|_E^2 + \|u_j - u^*_{j-1}\|_U^2 \,\mathrm{d}t + \|e_j(t_f)\|_T^2 \\
&= \arg\min_{\Lambda_j} \int_{t_0}^{t_f} \|e_j\|_E^2 + \|\Lambda_j \tilde{e}^*_{j-1}\|_U^2 \,\mathrm{d}t + \|e_j(t_f)\|_T^2
\end{aligned}$$

subject to

$$u_j = u^*_{j-1} + \Lambda_j e^*_{j-1}.$$

Then the input is not updated, i.e. $u^*_j \equiv u^*_{j-1}$, and generates the same error

$e_j^* \equiv e_{j-1}^*$. Now the next minimisation problem is

$$\begin{aligned}\Lambda_{j+1}^* &= \arg\min_{\Lambda_{j+1}} \int_{t_0}^{t_f} \|e_{j+1}\|_E^2 + \|u_{j+1} - u_j^*\|_U^2 \,\mathrm{d}t + \|e_{j+1}(t_f)\|_T^2 \\ &= \arg\min_{\Lambda_{j+1}} \int_{t_0}^{t_f} \|e_{j+1}\|_E^2 + \|u_{j+1} - u_{j-1}^*\|_U^2 \,\mathrm{d}t + \|e_{j+1}(t_f)\|_T^2 \\ &= \arg\min_{\Lambda_{j+1}} \int_{t_0}^{t_f} \|e_{j+1}\|_E^2 + \|\Lambda_{j+1}\tilde{e}_{j-1}^*\|_U^2 \,\mathrm{d}t + \|e_{j+1}(t_f)\|_T^2\end{aligned}$$

subject to

$$u_{j+1} = u_j^* + \Lambda_{j+1}\tilde{e}_j^* = u_{j-1}^* + \Lambda_{j+1}\tilde{e}_{j-1}^*.$$

This is the same problem as the one for Λ_j and hence $\Lambda_{j+1}^* = 0$ is the solution again. Because of that, the input is not updated, $u_{j+1}^* \equiv u_j^* \equiv u_{j-1}^*$, this leads to the same error, $e_{j+1}^* \equiv e_j^* \equiv e_{j-1}^*$, and finally

$$\begin{aligned}\rho_{j+1}^* - \rho_j^* &= \int_{t_0}^{t_f} \|e_{j+1}^*\|_E^2 + \|u_{j+1}^* - u_j^*\|_U^2 \,\mathrm{d}t + \|e_{j+1}^*(t_f)\|_T^2 \\ &\quad - \int_{t_0}^{t_f} \|e_j^*\|_E^2 + \|u_j^* - u_{j-1}^*\|_U^2 \,\mathrm{d}t - \|e_j^*(t_f)\|_T^2 \\ &= 0.\end{aligned}$$

This holds for $U \geq 0$.
Now the second part (3.10) is shown. First, to show (3.10a), let $\Lambda_j^* \neq 0$, that is all solutions of (3.6) are non-zero, and $U > 0$. This also means that $u_j^* - u_{j-1}^* = \Lambda_j^* \tilde{e}_{j-1}^* \neq 0$, because if that would be the case, then $\Lambda_j^* = 0$ would also be a solution. Note that if $\Lambda_{j+1} = 0$ is chosen, no update takes place and the error signal does not change, that is $e_{j+1}(\Lambda_{j+1} = 0) \equiv e_j^*$.

Then, the following holds

$$\begin{aligned}
\rho^*_{j+1} - \rho^*_j &\leq \rho_{j+1}(\Lambda_{j+1} = 0) - \rho^*_j \\
&= \int_{t_0}^{t_f} \|e_{j+1}(\Lambda_{j+1} = 0)\|_E^2 + \| \underbrace{u_{j+1} - u^*_j}_{=\Lambda_{j+1}\tilde{e}^*_j = 0} \|_U^2 \, \mathrm{d}t + \|e_{j+1}(\Lambda_{j+1} = 0, t_f)\|_T^2 \\
&\quad - \int_{t_0}^{t_f} \|e^*_j\|_E^2 + \|u^*_j - u^*_{j-1}\|_U^2 \, \mathrm{d}t - \|e^*_j(t_f)\|_T^2 \\
&= - \int_{t_0}^{t_f} \| \underbrace{u^*_j - u^*_{j-1}}_{\neq 0} \|_U^2 \, \mathrm{d}t \\
&< 0.
\end{aligned}$$

Second, to show (3.10b), let $U = 0$ and $\Lambda^*_{j+1} \neq 0$, i.e. $\Lambda_{j+1} = 0$ is not a solution, then

$$\rho^*_{j+1} < \rho_{j+1}(\Lambda_{j+1} = 0) = \rho^*_j$$

where the equality is because the error does not change between iterations since the input signal is not changed with $\Lambda_{j+1} = 0$ and the inequality holds because $\Lambda_{j+1} = 0$ is not a solution.

Proof of P3: P1 establishes that the policy values are a monotonically decreasing sequence. Furthermore, they are bounded from below by definition.

The main issue, however, with the proposed procedure is that the optimisation problem (3.6) is in general non-convex. Thus global optimisation is not an option and only local heuristics can be applied. To facilitate solving of (3.6) certain conditions on the system dynamics (3.1) are imposed to guarantee a sufficient smoothness of the policy value function ρ.

3.3 Exploiting the Derivatives of the Policy Value Function

Knowledge of the derivatives of the policy value function is useful in order to solve the optimisation problem (3.6), because then familiar optimisation methods can be employed. The first and second derivative of the proposed cost (3.7) are shown in the following and assumptions on the system dynamics (3.1) are proposed to ensure their existence and continuity. It is, without

loss of generality, assumed that the input $u \in \mathbb{R}$ is scalar to significantly simplify the notation. Then the learning gain is a (row) vector

$$\Lambda_j = \lambda_j \in \mathbb{R}^{1\times 2q}.$$

The gradient of the policy value function is

$$\begin{aligned}\nabla_{\lambda_j}\rho_j = 2\int_{t_0}^{t_f} e_j^T E\frac{\partial h}{\partial x}\left(x_j\right)\frac{\partial x}{\partial \lambda_j} + (u_j - u_{j-1})U\tilde{e}_{j-1}^T \,\mathrm{d}t \\ + 2e_j^T(t_f)T\frac{\partial h}{\partial x}\left(x_j(t_f)\right)\frac{\partial x}{\partial \lambda_j}(t_f)\end{aligned} \tag{3.12}$$

with the sensitivity function $\frac{\partial x}{\partial \lambda_j}(t)$ as the solution of the sensitivity equation [22]

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial x}{\partial \lambda_j}(t) = \frac{\partial f}{\partial x}\left(x_j(t), u_j(t)\right)\frac{\partial x}{\partial \lambda_j}(t) + \frac{\partial f}{\partial u}\left(x_j(t), u_j(t)\right)\tilde{e}_j^T(t) \tag{3.13a}$$

$$\frac{\partial x}{\partial \lambda_j}(t_0) = 0. \tag{3.13b}$$

Note that whenever a column vector and a row vector are multiplied in that order, e.g. $\frac{\partial f}{\partial u}\left(x_j(t), u_j(t)\right)\tilde{e}_j^T(t)$, the outer product has to be used. For the gradient of the policy value function (3.12) to exist and be continuous, the following assumptions have to hold:

- The initial input signal $u_0(t)$ is continuous in t.
- The state dynamics $f(x, u)$ are continuously differentiable in both arguments.
- The output equation $h(x)$ is continuously differentiable.

This facilitates the use of first-order methods and the first-order condition for optimality, i.e. $\nabla\rho = 0$, can be checked.

In addition, the second derivative of the policy value function is

$$\begin{aligned}\frac{\mathrm{d}^2\rho_j}{\mathrm{d}\lambda_j^2} = 2\int_{t_0}^{t_f} &\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j}\right)^T E \frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j} + e_j^T E\left(\frac{\partial^2 h}{\partial x^2}\left(\frac{\partial x}{\partial \lambda_j}\right)^2 + \frac{\partial h}{\partial x}\frac{\partial^2 x}{\partial \lambda_j^2}\right) \\ &+ \tilde{e}_{j-1} U \tilde{e}_{j-1}^T \,\mathrm{d}t \\ + 2\left(\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j}\right)^T T \frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j} + e_j^T T\left(\frac{\partial^2 h}{\partial x^2}\left(\frac{\partial x}{\partial \lambda_j}\right)^2 + \frac{\partial h}{\partial x}\frac{\partial^2 x}{\partial \lambda_j^2}\right)\right)&\Bigg|_{t=t_f}\end{aligned} \tag{3.14}$$

with $\frac{\partial^2 x}{\partial \lambda^2}(t)$ as the solution of the additional initial value problem

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial^2 x}{\partial \lambda^2} = &\frac{\partial f}{\partial x}\frac{\partial^2 x}{\partial \lambda^2} + \frac{\partial^2 f}{\partial x^2}\left(\frac{\mathrm{d}x}{\mathrm{d}\lambda_j}\right)^T \frac{\mathrm{d}x}{\mathrm{d}\lambda_j} + \frac{\partial^2 f}{\partial x \partial u}\tilde{e}_{j-1}^T \left(\frac{\mathrm{d}x}{\mathrm{d}\lambda_j}\right)^T \\ &+ \frac{\partial^2 f}{\partial u \partial x}\frac{\mathrm{d}x}{\mathrm{d}\lambda_j}\tilde{e}_{j-1} + \frac{\partial^2 f}{\partial u^2}\tilde{e}_{j-1}\tilde{e}_{j-1}^T\end{aligned} \tag{3.15a}$$

$$\frac{\partial^2 x}{\partial \lambda^2}(t_0) = 0. \tag{3.15b}$$

For the second derivative to also exist and be continuous, the following assumptions have to hold:

- The initial input signal $u_0(t)$ is continuous in t.
- The state dynamics $f(x, u)$ are twice continuously differentiable in both arguments.
- The output equation $h(x)$ is twice continuously differentiable.

The derivation of the first and the second derivative and the reasoning behind the assumptions can be found in Section 6.1 of the appendix. If the assumptions are met, the optimisation problem (3.6) can then be solved locally using second-order methods. With the groundwork laid out, the approach is ready for application.

4 Examples

The introduced ILC procedure is tested on two nonlinear systems for a task that can be controlled by an LQR. The two examples are also motivated by the concept of human-machine-cooperation, where the proposed ILC scheme has to cooperate with an only partially implemented controller in order to execute a task together. While both examples feature a double pendulum, the structure is different. The first one is a double pendulum on a cart with three outputs but only two inputs, it is therefore underactuated. The second one is a fixed double pendulum with two outputs and two inputs in addition to a harder task.

4.1 Double Pendulum on a Cart

The considered application is a double pendulum attached to a cart. A diagram of the system is depicted in Figure 4.1. The two inputs, the force F and the torque τ, are used to influence the configuration of the system. It is described by the position of the cart x, the angle of the closer pendulum φ and the angle of the further pendulum ψ and together with their velocities they constitute the state vector

$$z = \begin{bmatrix} x & \varphi & \psi & \dot{x} & \dot{\varphi} & \dot{\psi} \end{bmatrix}^T. \tag{4.1}$$

The equations of motion of the double pendulum on a cart are given in the form

$$\dot{z} = f(z, u), \tag{4.2}$$

with

$$u = \begin{bmatrix} F \\ \tau \end{bmatrix}. \tag{4.3}$$

The explicit formulation and derivation can be found in Section 6.2 of the appendix. The parameters of the system are chosen as $m_1 = m_2 = m_3 = 1\,\mathrm{kg}$, $l_2 = l_3 = 1\,\mathrm{m}$ and $g = 9.81\,\mathrm{kg/ms^2}$ for simplicity.

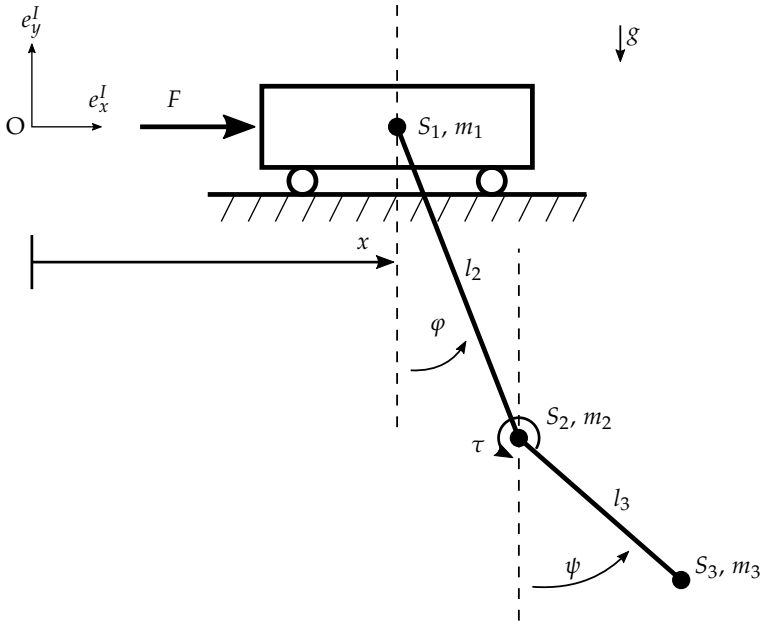

Figure 4.1: Diagram of the double pendulum on a cart. The two inputs are the force F on the cart and the torque τ on the link between the two rods.

The idea is to design an LQR for both inputs, but to only implement the torque $\tau = \tau_{\text{LQR}}(z)$, changing the system into

$$\dot{z} = f\left(z, \begin{bmatrix} F \\ \tau_{\text{LQR}}(z) \end{bmatrix}\right) = \tilde{f}(z, F), \tag{4.4}$$

with the force F left open for learning. The question is then if a human can learn to apply $F(t)$ in such a way that the system can perform its task which is tracking a reference trajectory. To emulate this human learning, the ILC with parameter tuning as proposed in Chapter 3 is used.

The task the system has to perform can roughly be described as moving the cart from the position $x_d(0) = -1$ to $x_d(10) \approx 0$ while having the pendulum hang down with (almost) no swinging, i.e. $\varphi_d(t) \approx 0$ and $\psi_d(t) \approx 0$ for $t \in [0, 10]$. With this, the output equation is

$$y = h(z) = \begin{bmatrix} x \\ \varphi \\ \psi \end{bmatrix}. \tag{4.5}$$

If the system would be initialised in the standing position, the double pendulum would presumptively topple and it would be hard to learn from the resulting error signal.

The first step is to design the LQR for both inputs which tracks the desired trajectory. The feasible input that generates the reference trajectory is denoted as $u_d = \begin{bmatrix} F_d & \tau_d \end{bmatrix}^T$ and the corresponding differential equation is

$$\dot{z}_d = f(z_d, u_d). \tag{4.6}$$

The derivation of the feasible input and the reference trajectory can be found in Section 6.3 in the appendix. Note that the feasible inputs not only realise the reference trajectory, that is y_d, but in fact a reference for all the states. While the LQR will track those additional desired states as well, they will be later neglected in the ILC. The difference between the actual states and the desired ones is defined as

$$\delta z := z - z_d. \tag{4.7}$$

Differentiation and a linear Taylor approximation yield

$$\begin{aligned}\dot{\delta z} &= \dot{z} - \dot{z}_d \\ &= f(z,u) - f(z_d,u_d) \\ &\approx f(z_d,u_d) + \frac{\partial f}{\partial z}(z_d,u_d)(z-z_d) + \frac{\partial f}{\partial u}(z_d,u_d)\underbrace{(u-u_d)}_{:=\delta u} - f(z_d,u_d) \\ &= \underbrace{\frac{\partial f}{\partial z}(z_d,u_d)}_{:=A(t)}\delta z + \underbrace{\frac{\partial f}{\partial u}(z_d,u_d)}_{:=B(t)}\delta u,\end{aligned}$$

which is a linear time-variant system

$$\dot{\delta z}(t) = A(t)\delta z + B(t)\delta u \tag{4.8}$$

for which a finite horizon LQR can be designed. The cost is

$$J_{\text{LQR}} = \int_0^{10} \delta z^T Q \delta z + \delta u^T R \delta u \,\mathrm{d}t + \delta z(10)^T S \delta z(10) \tag{4.9}$$

with the weighting matrices

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix},$$

$$S = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The weights in Q are chosen that way because completing the move from left to right is deemed more important than keeping the pendulum straight, but both is more important than tracking the velocities. In addition, the cart's final position should correspond to the desired one, whereas the pendulum's

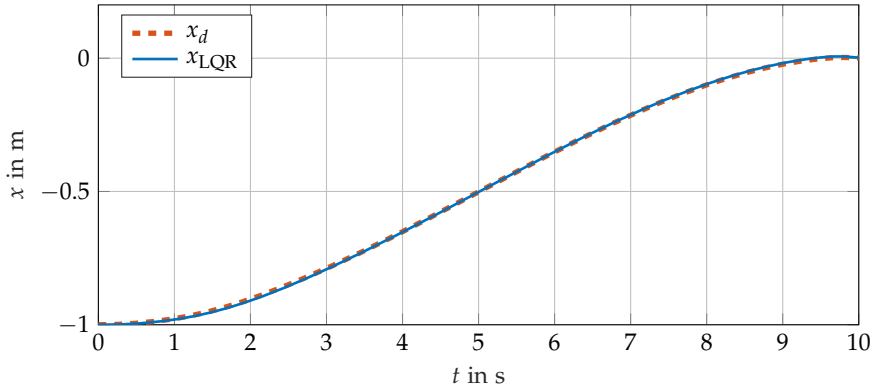

Figure 4.2: Resulting position of the cart when the full LQR is applied to the double pendulum on a cart.

final angles are not as important and tracking in general has priority over small input signals, thus the weights in S and R. The optimal feedback is

$$\delta u_{\text{LQR}}(t) = -R^{-1}B(t)^T P(t)\delta z(t) \tag{4.10}$$

with $P(t)$ as the solution of the Riccati differential equation

$$\dot{P}(t) = P(t)B(t)R^{-1}B(t)^T P(t) - P(t)A(t) - A^T(t)P(t) - Q, \tag{4.11a}$$
$$P(10) = S. \tag{4.11b}$$

The tracking when δu_{LQR} is applied can be seen in figures 4.2 – 4.6 together with the reference trajectory and the feasible inputs that realise it. Nevertheless, only the torque feedback τ_{LQR} is realised and the force F has to be learned with ILC over a number of iterations.

In order to emulate a human learning to cooperate with the partially implemented LQR, the ILC procedure as outlined in Chapter 3 is applied. With the tracking error defined as

$$e = y - y_d \tag{4.12}$$

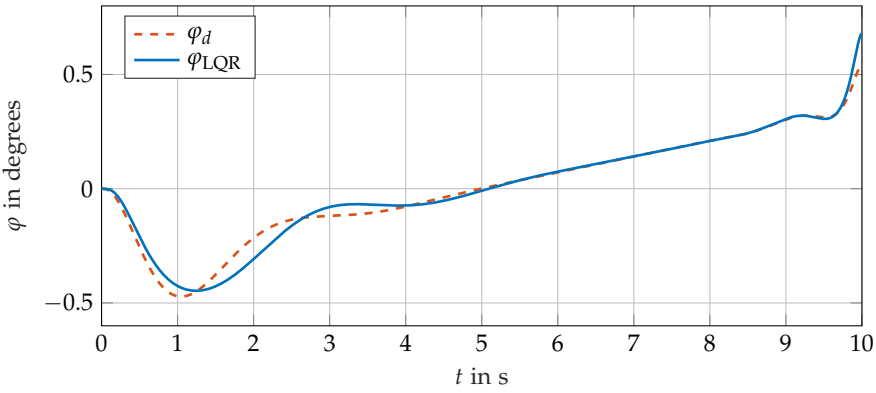

Figure 4.3: Resulting trajectory of the closer angle when the full LQR is applied to the double pendulum on a cart.

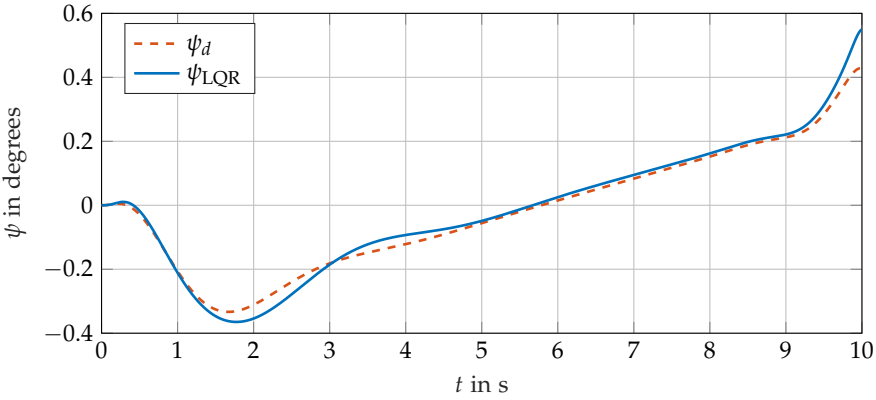

Figure 4.4: Resulting trajectory of the further angle when the full LQR is applied to the double pendulum on a cart.

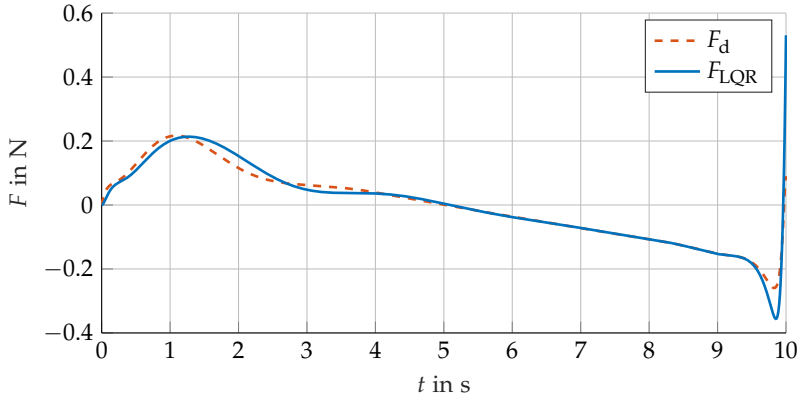

Figure 4.5: Input force of the LQR. The corresponding system is the double pendulum on a cart.

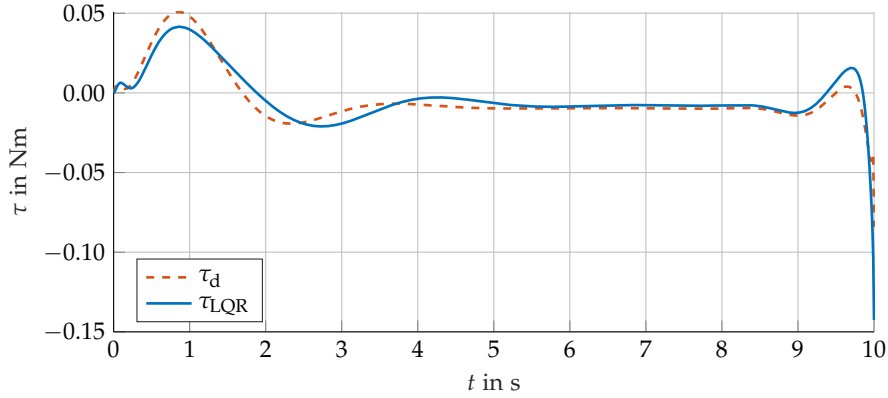

Figure 4.6: Input torque of the LQR. The corresponding system is the double pendulum on a cart.

the associated policy value function, which is comparable to the LQR cost, is

$$\begin{aligned}\rho_{j+1}(\Lambda_{j+1}) &= \int_0^{10} e_{j+1}^T E e_{j+1} + (F_{j+1} - F_j)^T U (F_{j+1} - F_j)\,\mathrm{d}t + e_{j+1}^T(t_f) T e_{j+1}(t_f) \\ &= \int_0^{10} e_{j+1}^T E e_{j+1}\,\mathrm{d}t + e_{j+1}^T(t_f) T e_{j+1}(t_f) \end{aligned} \tag{4.13}$$

with the weighting matrices

$$E = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}, \quad U = 0, \quad T = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and $j \in \mathbb{N}^0$ is again the iteration index. Since $\delta z = \begin{bmatrix} e^T & \dot{x} & \dot{\varphi} & \dot{\psi} \end{bmatrix}^T$, the ILC weighting matrices are chosen corresponding to the LQR matrices, that is

$$Q = \begin{bmatrix} E & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}, \quad S = \begin{bmatrix} T & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \tag{4.14}$$

As mentioned before, the ILC only tracks the output, i.e. it only penalises the error in x, φ and ψ. The policy value when the full LQR is applied is $\rho_{\mathrm{LQR}} = 1.09 \times 10^{-3}$. The cooperation between ILC and LQR is deemed successful if it performs better than the fully-implemented LQR, that is if $\rho_j < \rho_{\mathrm{LQR}}$ for some j.

The ILC procedure includes solving the non-convex optimisation problem (3.6) with

$$\rho_{j+1}^* = \min_{\Lambda_{j+1}} \rho(\Lambda_{j+1})$$

where $\Lambda_{j+1} \in \mathbb{R}^{1\times 6}$ is a 6-dimensional decision variable. In order to find a solution of (3.6), several local searches are performed and the best solution is taken. Unfortunately, this might mean that the optimal solution is missed. If the solution is the zero learning gain, the procedure can be stopped, because the policy value cannot be improved in future iterations, as shown in Section 3.2. Thus it is sensible to include the zero gain as an initial guess. For the minimisation MATLAB's `fminunc` function with a quasi-Newton algorithm is used. Because this is a second order method, it uses the gradient

and the Hessian of the objective function. The gradient (3.12) is calculated and passed to `fminunc` whereas the Hessian (3.14) is not computed but approximated by `fminunc`'s default algorithm. The initial guesses for the minimisation are given by a grid over the search space; each component of the learning gain is either -1, 0 or 1 and all possible combinations are used. This leads to 729 initial guesses in total that are explored in each iteration using MATLAB's `multiStart` function because it supports parallel computing.

The initial input is chosen as the zero signal

$$F_0 \equiv 0, \tag{4.15}$$

which can be interpreted as doing nothing and observing how the half-implemented LQR reacts before any action is taken. The input force is subsequently updated according to the input update law (3.6c)

$$F_{j+1} = F_j + \Lambda^*_{j+1} \begin{bmatrix} e_j \\ \dot{e}_j \end{bmatrix}, \quad j \in \mathbb{N}^0.$$

On a PC with four cores at 3.40 GHz one iteration takes about 3190 s or 53 min 10 s on average. Learning takes place for 36 iterations and the resulting policy values are shown in Figure 4.7. The learned input cooperates successfully with the half-implemented LQR, surpassing the performance of the fully implemented LQR with regards to the policy value after 10 iterations. The last policy value is $\rho_{36} = 3.76 \times 10^{-5}$, whereas the full LQR has a policy value of $\rho_{\text{LQR}} = 1.09 \times 10^{-3}$. The error in the cart's position is depicted in Figure 4.8, it can be seen that the cooperation tracks the reference position more closely. Note that the ILC learns a feedforward signal and can thus anticipate the move, while the LQR has to wait for an error to feed back. The two angles are depicted in figures 4.9, 4.10, 4.11 and 4.12, the plots show that the ILC produces trajectories which undulate close to the reference. The same holds true for the learned and feasible forces shown in figures 4.13 and 4.14, as well as for the torque resulting from the half-implemented LQR seen in Figure 4.15. All inputs can be seen to peak at the end of the time span. A smaller ILC run without a terminal cost, that is $T = 0$, indicates that this peak is not caused by the terminal cost, as seen in figures 4.16 and 4.17. Although it is probable that in each iteration only a sub-optimal learning gain is found, the procedure is able to generate a strictly monotonically decreasing sequence of policy values. Consequently, the ILC scheme learned to cooperate with the half-implemented LQR in

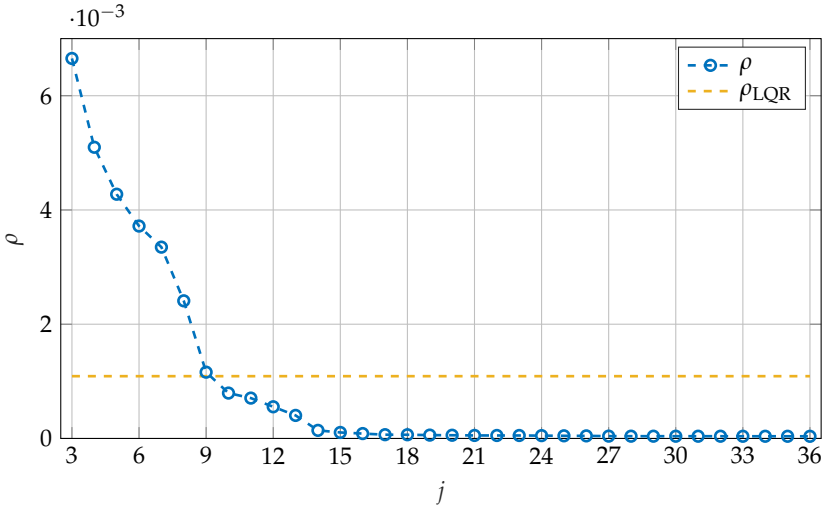

Figure 4.7: Policy values when the ILC is applied starting with a zero input signal. The initial policy value, i.e. before any input updates, $\rho_0 = 116$ and the next two, $\rho_1 = 0.0480$ and $\rho_2 = 0.0105$, are not included. The cooperation performs better than the fully-implemented LQR after 10 iterations. The corresponding system is the double pendulum on a cart.

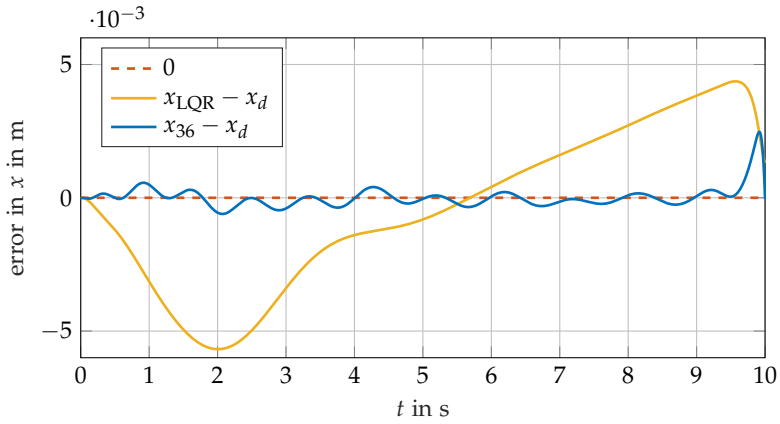

Figure 4.8: Error in the cart's position when the ILC is applied after 36 iterations and when the full LQR is applied. The corresponding system is the double pendulum on a cart.

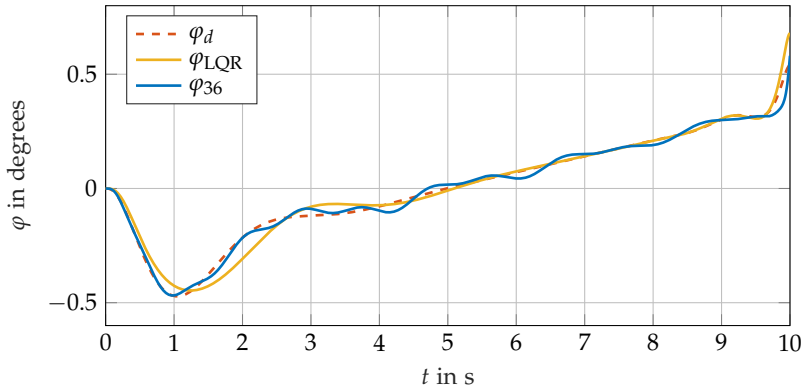

Figure 4.9: Angle of the closer pendulum after 36 iterations of ILC, when the full LQR is applied and the desired angle. The corresponding system is the double pendulum on a cart.

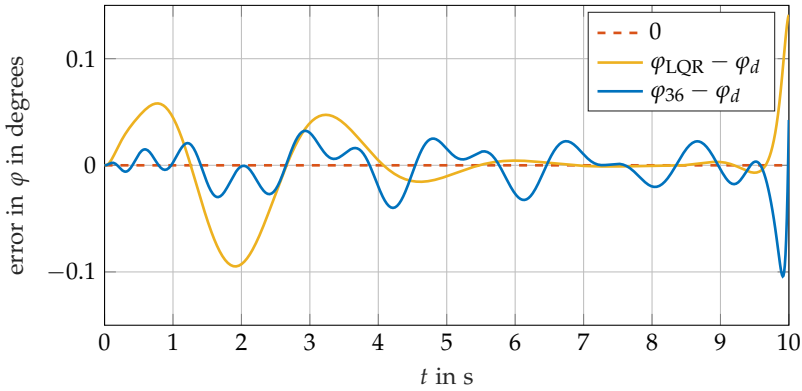

Figure 4.10: Error in the closer angle when the ILC is applied after 36 iterations and when the full LQR is applied. The corresponding system is the double pendulum on a cart.

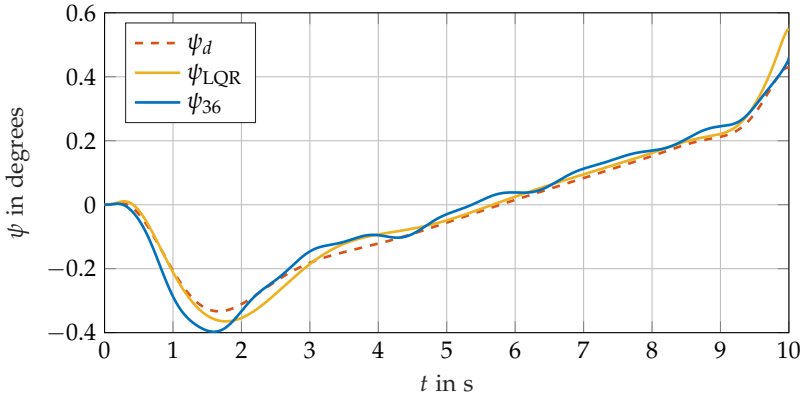

Figure 4.11: Angle of the further pendulum after 36 iterations of ILC, when the full LQR is applied and the desired angle. The corresponding system is the double pendulum on a cart.

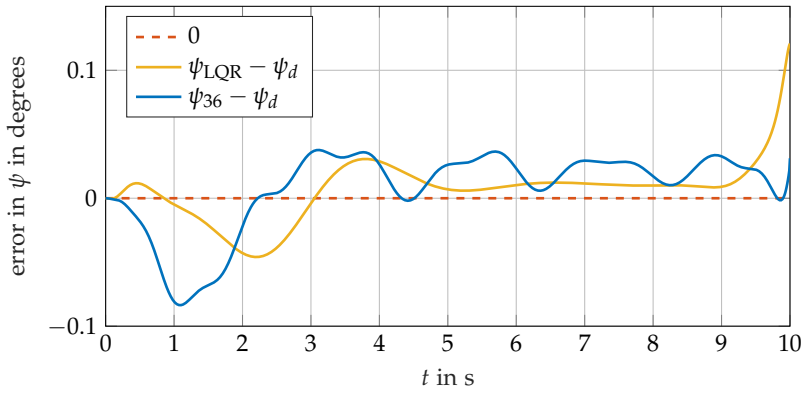

Figure 4.12: Error in the further angle when the ILC is applied after 36 iterations and when the full LQR is applied. The corresponding system is the double pendulum on a cart.

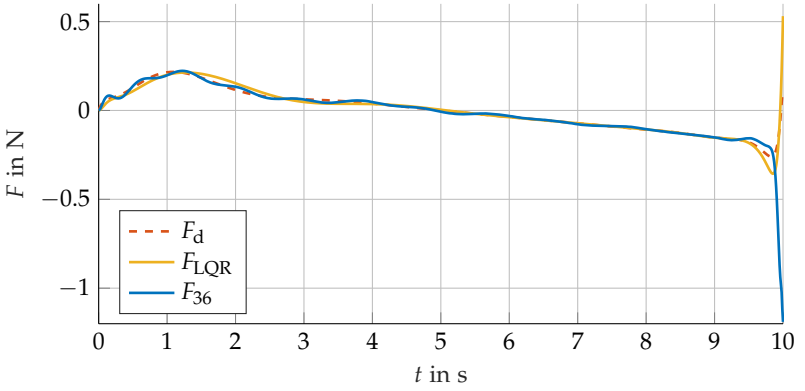

Figure 4.13: Applied force after 36 iterations of ILC. In addition, the force applied when the full LQR is used and the feasible force are shown. The corresponding system is the double pendulum on a cart.

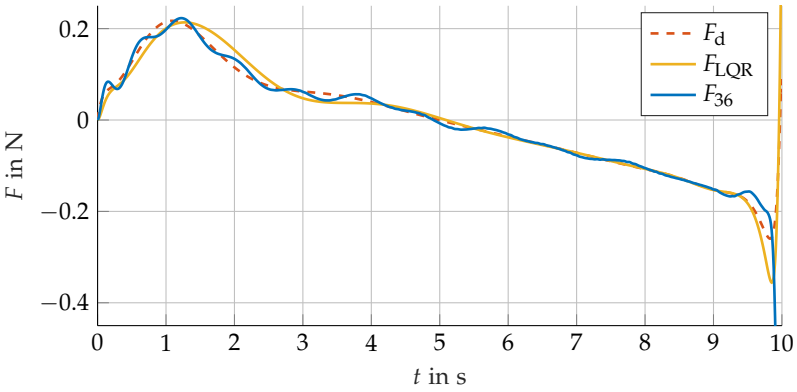

Figure 4.14: Applied force after 36 iterations of ILC. In addition, the force applied when the full LQR is used and the feasible force are shown. The corresponding system is the double pendulum on a cart. (The y-axis is different in comparison to Figure 4.13)

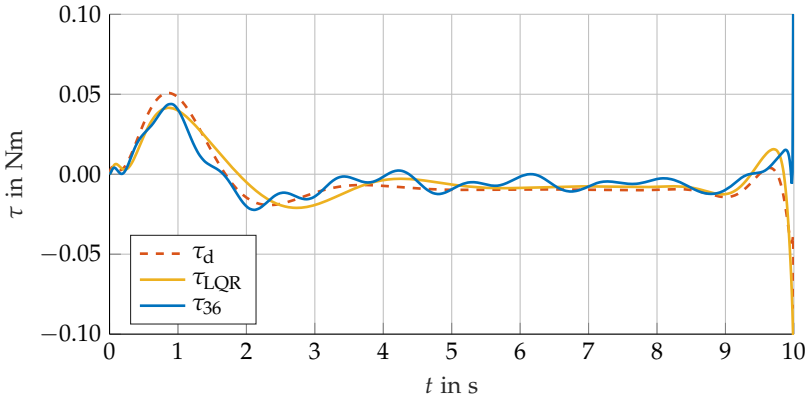

Figure 4.15: Applied torque of the LQR, when the force is learned via ILC, after 36 iterations, the torque applied by the fully implemented LQR and the feasible torque. The corresponding system is the double pendulum on a cart.

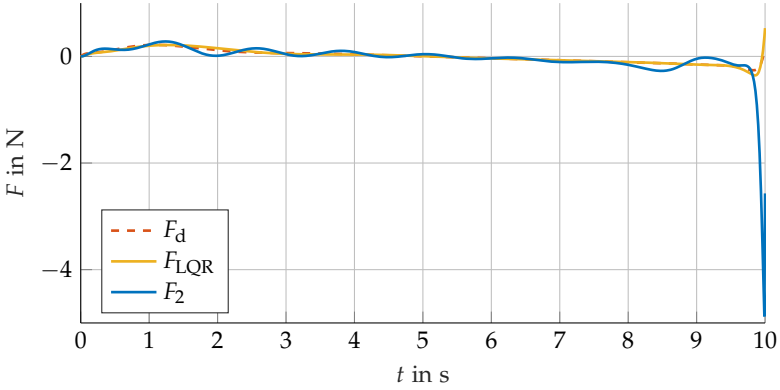

Figure 4.16: Applied force after 2 iterations of ILC without a terminal cost. The peak at the end still persists and thus seems not to be caused by the terminal cost. The corresponding system is the double pendulum on a cart.

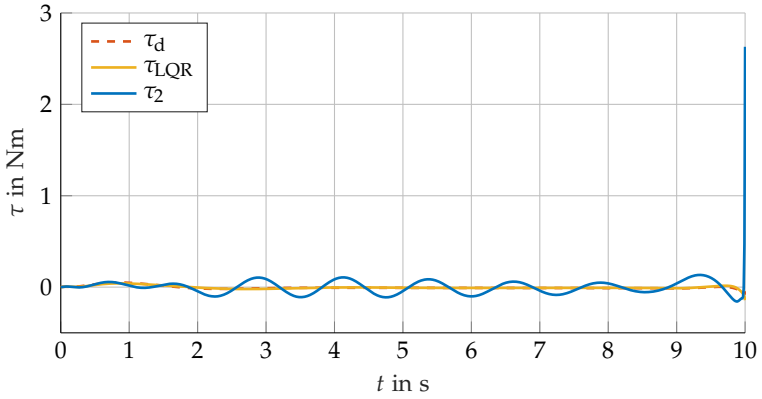

Figure 4.17: Applied torque of the LQR, when the forced is learned via ILC without a terminal cost, after 2 iterations. The peak at the end still persists and thus seems not to be caused by the terminal cost. The corresponding system is the double pendulum on a cart.

order to complete the provided task and a human is arguably able to do the same. A comparison of three different stages in the learning process is provided in figures 4.18, 4.19, 4.20, 4.21, 4.22, 4.23, 4.24 and 4.25. All applied forces are plotted in Figure 4.26 and are of reasonable magnitude, even though with $U = 0$ the change in the input signal is not penalised.

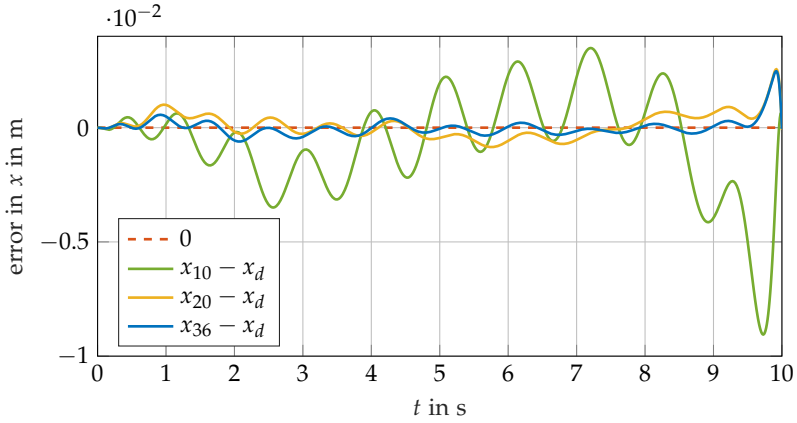

Figure 4.18: Error in the cart's position when the ILC is applied after 10, 20 and 36 iterations. The corresponding system is the double pendulum on a cart.

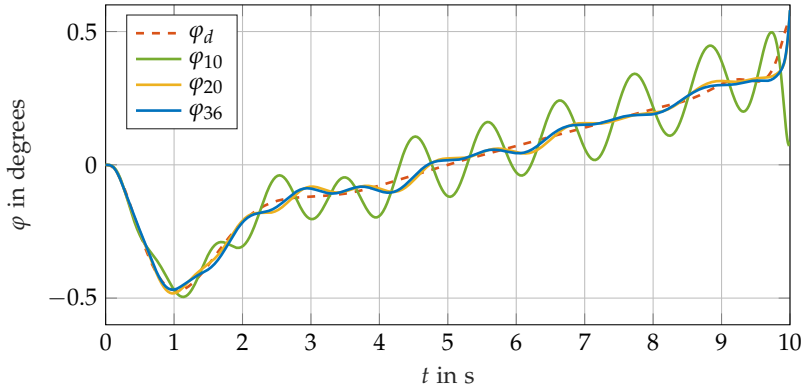

Figure 4.19: Angle of the closer pendulum after 10, 20 and 36 iterations of ILC. The corresponding system is the double pendulum on a cart.

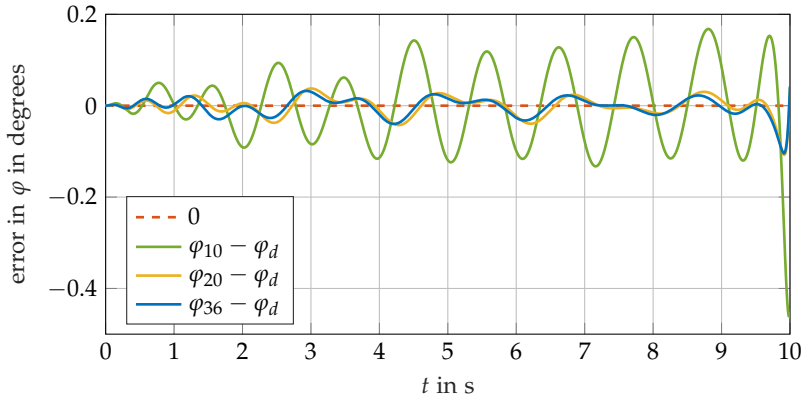

Figure 4.20: Error in the angle of the closer pendulum after 10, 20 and 36 iterations of ILC. The corresponding system is the double pendulum on a cart.

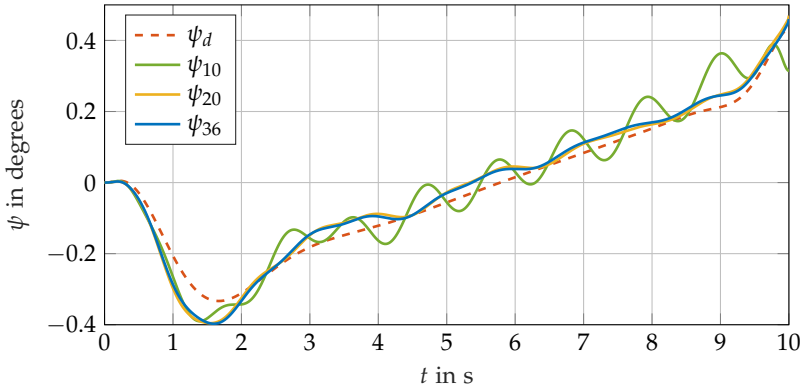

Figure 4.21: Angle of the further pendulum after 10, 20 and 36 iterations of ILC. The corresponding system is the double pendulum on a cart.

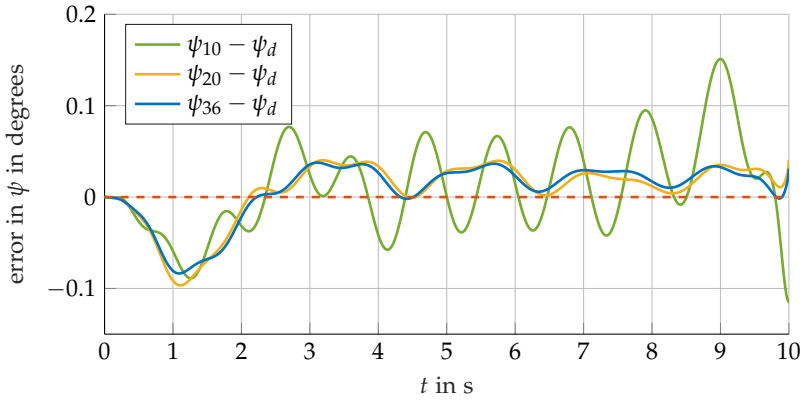

Figure 4.22: Errors in the angle of the further pendulum after 10, 20 and 36 iterations of ILC. The corresponding system is the double pendulum on a cart.

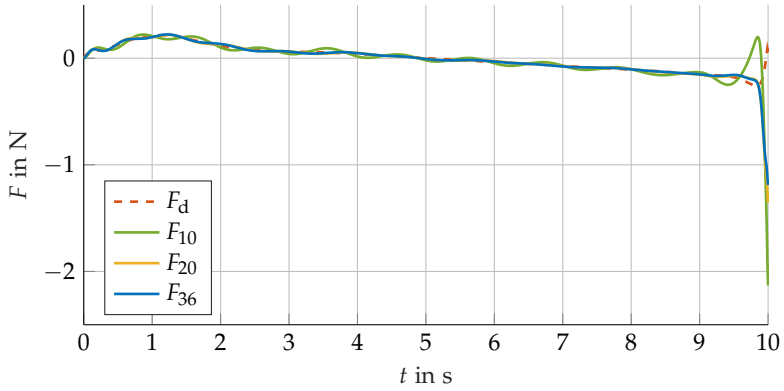

Figure 4.23: Applied force after 10, 20 and 36 iterations of ILC. The corresponding system is the double pendulum on a cart.

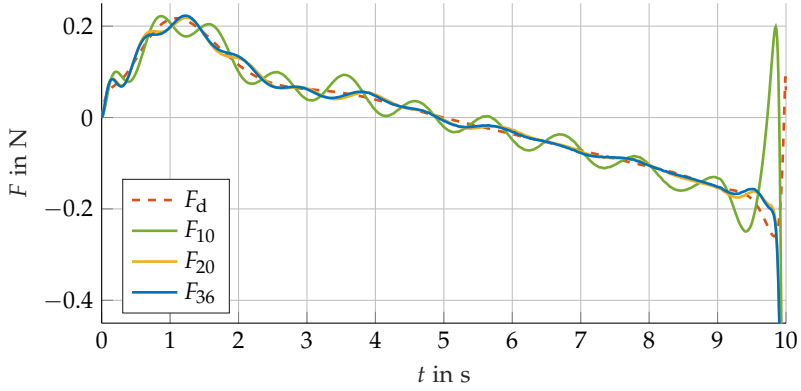

Figure 4.24: Applied force after 10, 20 and 36 iterations of ILC. The corresponding system is the double pendulum on a cart. (This figure has a different y-axis than Figure 4.23)

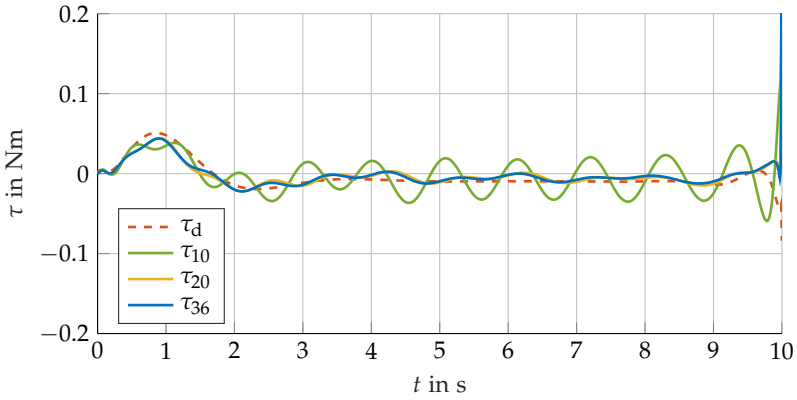

Figure 4.25: Applied torque of the LQR, when the force is learned via ILC, after 10, 20 and 36 iterations. The corresponding system is the double pendulum on a cart.

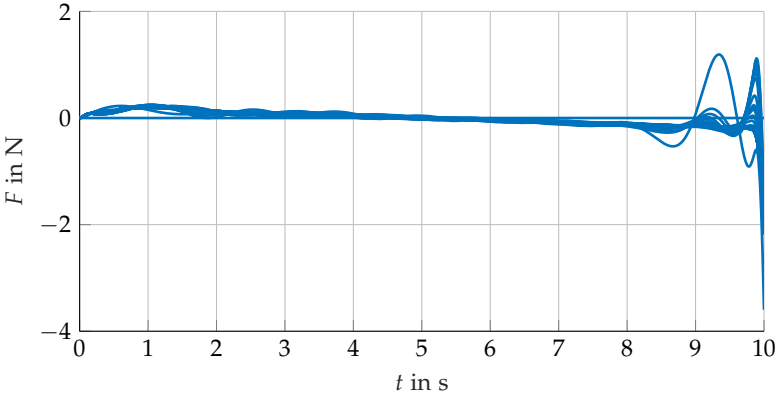

Figure 4.26: All applied forces of the ILC. The corresponding system is the double pendulum on a cart.

4.2 Double Pendulum

The next application is the double pendulum depicted in Figure 4.27. The inputs are the two torques η, which acts on the pivot, and τ, which acts on the link between the rods. For simplicity, the parameters are chosen as $m_1 = m_2 = 1\,\text{kg}$, $l_1 = l_2 = 1\,\text{m}$ and $g = 9.81\,\text{kg/ms}^2$. The system has the states and inputs

$$x = \begin{bmatrix} \varphi \\ \psi \\ \dot{\varphi} \\ \dot{\psi} \end{bmatrix}, \tag{4.16}$$

$$u = \begin{bmatrix} \eta \\ \tau \end{bmatrix} \tag{4.17}$$

and dynamics of the form

$$\dot{x} = f(x, u). \tag{4.18}$$

The derivation of the dynamics (4.18) can be found in Section 6.4 of the appendix. The approach is similar to the one in the previous example, but this time η is learned by the ILC, while τ is controlled by the half-implemented LQR.

The task the system has to perform is a swing-up in 10 s while keeping the double pendulum straight, that is both angles follow the same trajectory. For this, a polynomial is chosen

$$\varphi_d(t) = \psi_d(t) = \pi\left(\frac{10}{10^3}t^3 - \frac{15}{10^4}t^4 + \frac{6}{10^5}t^5\right). \tag{4.19}$$

The output equation is then

$$y = h(x) = \begin{bmatrix} \varphi \\ \psi \end{bmatrix}. \tag{4.20}$$

The feasible input $u_d = \begin{bmatrix} \eta_d & \tau_d \end{bmatrix}^T$ is computed by inserting the reference trajectory (4.19) and its time derivatives into the the equations of motion (6.20) and (6.21) and solving for η_d and τ_d. The corresponding dynamics are denoted as

$$\dot{x}_d = f(x_d, u_d). \tag{4.21}$$

Let the deviation from the desired states be

$$\delta x := x - x_d. \tag{4.22}$$

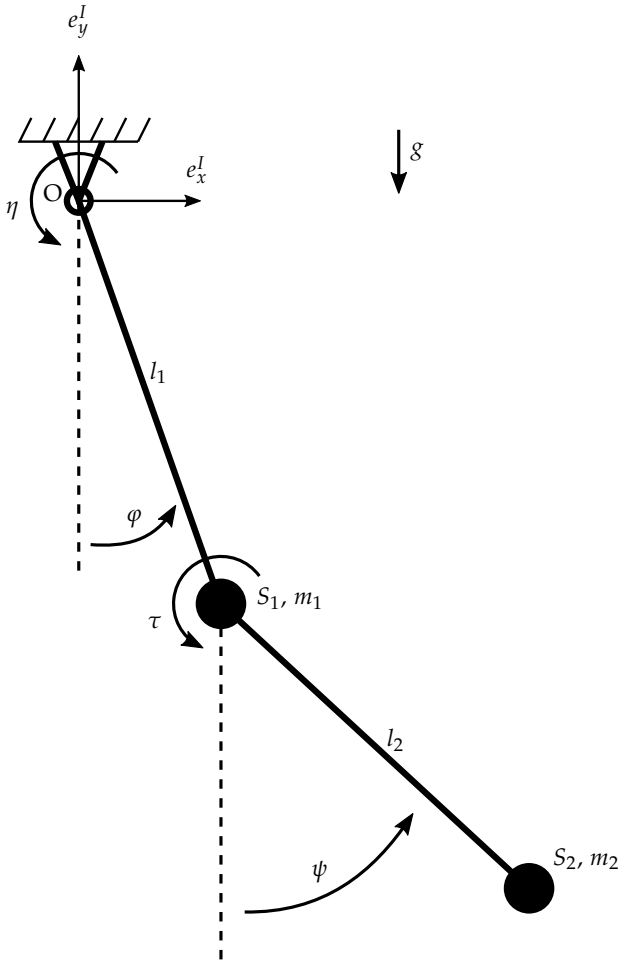

Figure 4.27: Diagram of the double pendulum. The two inputs are the torques η and τ on the pivot and the link between the pendulums. Each rod is assumed to be weightless with a point mass fixed to their ends.

Differentiation and a linear Taylor approximation yield

$$\begin{aligned}\dot{\delta x} &= \dot{x} - \dot{x}_d \\ &= f(x,u) - f(x_d,u_d) \\ &\approx f(x_d,u_d) + \frac{\partial f}{\partial x}(x_d,u_d)(x-x_d) + \frac{\partial f}{\partial u}(x_d,u_d)\underbrace{(u-u_d)}_{:=\delta u} - f(x_d,u_d) \\ &= \underbrace{\frac{\partial f}{\partial x}(x_d,u_d)}_{:=A(t)}\delta x + \underbrace{\frac{\partial f}{\partial u}(x_d,u_d)}_{:=B(t)}\delta u\end{aligned}$$

which is a linear time-variant system

$$\dot{\delta x}(t) = A(t)\delta x + B(t)\delta u. \tag{4.23}$$

Now a finite horizon LQR is designed for (4.23). The cost is

$$J_{\text{LQR}} = \int_0^{10} \delta x^T Q \delta x + \delta u^T R \delta u \,\mathrm{d}t + \delta x(10)^T S \delta x(10) \tag{4.24}$$

with the weighting matrices

$$\begin{aligned}Q &= \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, \\ R &= \begin{bmatrix}0.01 & 0\\ 0 & 0.01\end{bmatrix}, \\ S &= \begin{bmatrix}10 & 0 & 0 & 0\\ 0 & 10 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.\end{aligned}$$

This time, all states are weighted equally, but priority is given to the angles ending up in their desired positions. The optimal feedback is

$$\delta u_{\text{LQR}}(t) = -R^{-1}B(t)^T P(t)\delta x(t) \tag{4.25}$$

with $P(t)$ as the solution of the Riccati differential equation

$$\dot{P}(t) = P(t)B(t)R^{-1}B(t)^T P(t) - P(t)A(t) - A^T(t)P(t) - Q, \tag{4.26a}$$

$$P(10) = S. \tag{4.26b}$$

Only the feedback part corresponding to τ is implemented and η is left open for learning.

The ILC procedure in Chapter 3 is again applied with the policy value function

$$\begin{aligned}\rho_{j+1}(\Lambda_{j+1}) &= \int_0^{10} e_{j+1}^T E e_{j+1} + (\eta_{j+1} - \eta_j)^T U (\eta_{j+1} - \eta_j)\,\mathrm{d}t + e_{j+1}^T(t_f) T e_{j+1}(t_f) \\ &= \int_0^{10} e_{j+1}^T E e_{j+1}\,\mathrm{d}t + e_{j+1}^T(t_f) T e_{j+1}(t_f) \end{aligned} \tag{4.27}$$

where the tracking error is defined as

$$e = y - y_d \tag{4.28}$$

and the difference in the input is neglected with $U = 0$. Because $\delta x = \begin{bmatrix} e & \dot{\varphi} & \dot{\psi} \end{bmatrix}^T$, the weighting matrices on the error are chosen similar to the weights of the LQR

$$E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$
$$T = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}.$$

The cooperation between ILC and LQR is deemed successful if it performs better than the fully-implemented LQR, that is if $\rho_j < \rho_{\mathrm{LQR}}$ for some j.

The torque on the pivot is updated in each iteration according to the update law (3.6c)

$$\eta_{j+1} = \eta_j + \Lambda_{j+1}^* \begin{bmatrix} e_j \\ \dot{e}_j \end{bmatrix}, \quad j \in \mathbb{N}^0$$

where the learning gain is computed by solving (3.6)

$$\rho_{j+1}^* = \min_{\Lambda_{j+1}} \rho(\Lambda_{j+1}).$$

This time, MATLAB's `fminunc` is used without providing the gradient, instead it is approximated by `fminunc` using forward finite differences. In addition, only one initial guess, the zero gain, is provided. The initial input signal is chosen as

$$\eta_0 \equiv 0$$

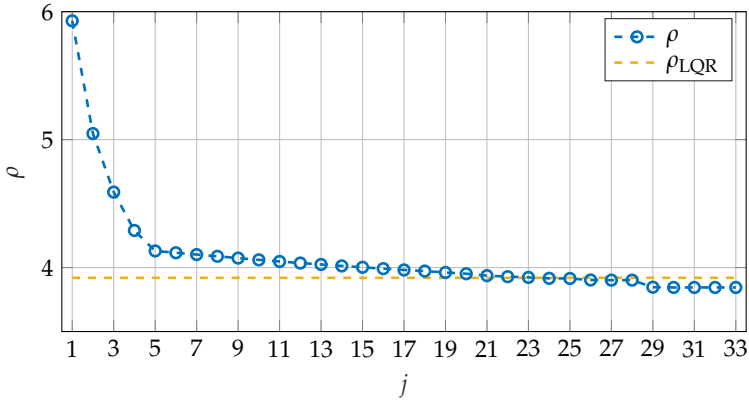

Figure 4.28: Policy values when the ILC is applied starting with a zero input signal. The initial policy value, i.e. before any input updates, $\rho_0 = 160$ is not plotted. The cooperation performs better than the fully-implemented LQR after 24 iterations. The corresponding system is the double pendulum with two torques.

and the ILC is run for 33 iterations. In the last iteration, the learning gain is the zero matrix. The results are plotted in figures 4.28 – 4.32. Despite using only one initial guess in the optimisation problem (3.6), the ILC is able to cooperate reasonably well with the half-implemented LQR, their performance is better than the fully-implemented LQR after 23 iterations. The angle ψ of the further pendulum, depicted in Figure 4.30, follows almost the trajectory of ψ_{LQR}, the angle when the full LQR is used. The last policy value is $\rho_{33} = 3.84$ and only slightly better than the policy value of the LQR $\rho_{\text{LQR}} = 3.92$.

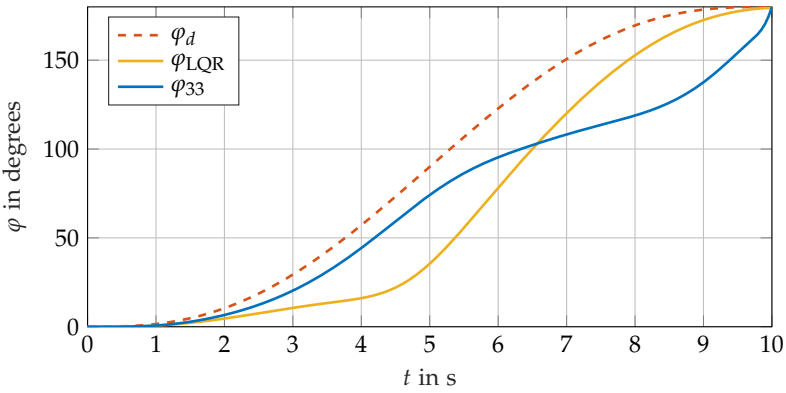

Figure 4.29: Angle of the closer pendulum after 33 iterations of ILC. The corresponding system is the double pendulum with two torques.

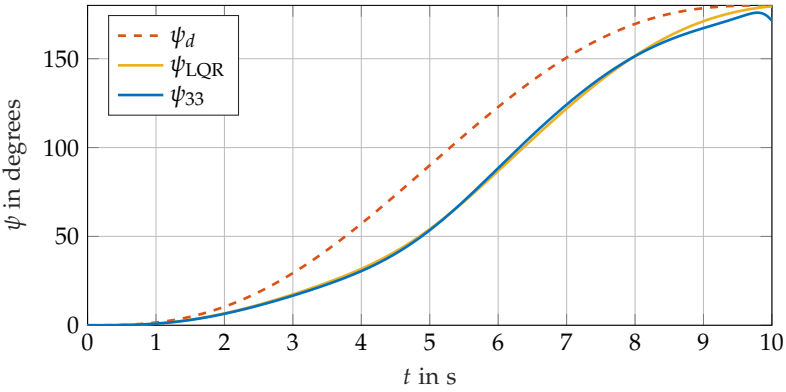

Figure 4.30: Angle of the further pendulum after 33 iterations of ILC. The corresponding system is the double pendulum with two torques.

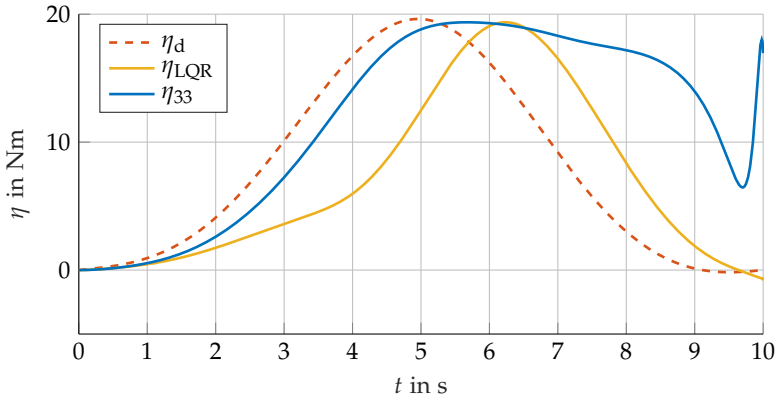

Figure 4.31: Learned torque on the pivot of the double pendulum after 33 iterations of ILC. The corresponding system is the double pendulum with two torques.

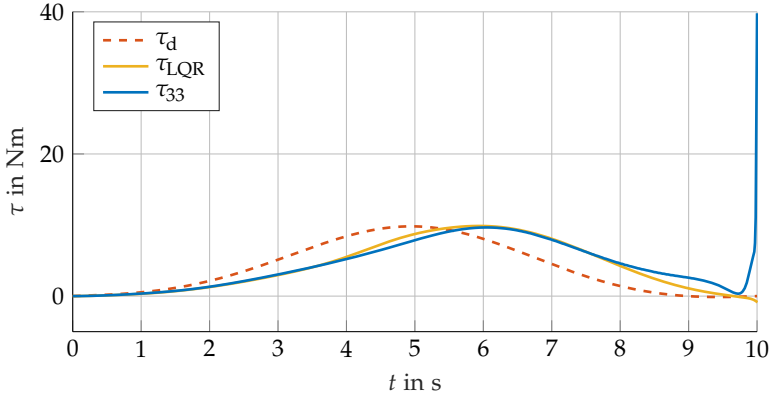

Figure 4.32: Torque τ_{33} of the half-implemented LQR after 33 iterations of ILC. τ_{LQR} is the torque when the full LQR is applied. The corresponding system is the double pendulum with two torques.

5 Summary and Outlook

In this work, an approach that applies iterative learning control (ILC) to nonlinear, continuous-time systems is proposed by tuning the learning gain with reinforcement learning (RL). It was motivated by human-machine-cooperation in which a human has to learn to cooperate with a partially-implemented controller. One example of such a human-machine-cooperation are actuated orthoses [26]. The human learning process is substituted by ILC, as it is deemed to be an appropriate learning scheme that is able to emulate human learning to some degree [5]. Because the natural environment of a human is the nonlinear world, a procedure is needed to successfully apply ILC for nonlinear systems. This procedure is found by combining ILC with RL. In every iteration of the ILC an optimisation problem is solved for the learning gain that leads to the lowest policy value after updating the input signal with it. The main drawback is the, in general, non-convexity of the problem which only enables a local search for an appropriate (suboptimal) solution. Nevertheless, if a non-trivial solution is found, then a smaller policy value is guaranteed and if the solution is the zero learning gain, then learning stops. Conditions on the nonlinear system are given that guarantee the existence of continuous second-order derivatives. This facilitates the use of second-order optimisation methods. With this in hand, the cooperation between a classic controller, in this case a linear quadratic regulator (LQR), and ILC as a substitute for a human, is tested on a double pendulum on a cart and on a regular double pendulum. The application is successful, the ILC procedure is able to learn to cooperate with the half-implemented LQR and they complete the task.

Although the use of ILC to emulate human learning is reasonable, a more sophisticated scheme is arguably more appropriate to imitate this complex process. The presented examples are simple and there exist certainly more realistic examples in which a human has to assume input channels from the controller for which the procedure could be tested, for example orthoses or prostheses. In this work, the proposed ILC scheme is only used in the framework of this human-machine-cooperation, but if it were to be used for the classical applications of ILC, then several open question have to

be investigated first. For example, the real-time availability of the optimal solution needs to be ensured. More importantly, the scheme heavily relies on exact system knowledge to predict the next policy value. Because ILC is inherently based on uncertainty in the system dynamics, the robustness of the scheme has to be examined before it can be used in this context. In addition, there might be a benefit in looking at the policy value after the next, to solve the optimisation problem for that policy value and to keep it constant for two iterations instead of one. Alternatively, the optimisation problem could be solved for more policy values to come, with a possibly different learning gain in each iteration, but to only update the input for the next iteration and to solve this problem again in a model-predictive-control-like fashion. This, however, is left open for future work.

6 Appendix

The appendix provides additional information that was omitted in the main parts.

6.1 Derivatives of the Policy Value Function

In order to simplify the notation of the derivatives, it is assumed that the learning gain is a row vector $\Lambda_j = \lambda_j \in \mathbb{R}^{1\times 2q}$, that is the input is a scalar. First, the gradient of the policy value function is derived. The following assumptions encompass all requirements that are made in its derivation:

- The initial input signal $u_0(t)$ is continuous in t.
- The state dynamics $f(x,u)$ are continuously differentiable in both arguments.
- The output equation $h(x)$ is continuously differentiable.

The policy value function in iteration j is

$$\rho_j(\lambda_j) = \int_{t_0}^{t_f} e_j^T E e_j + U(u_j - u_{j-1})^2 \,\mathrm{d}t + e_j^T(t_f) T e_j(t_f)$$

and because it is real-valued, its gradient is the total derivative

$$\nabla_{\lambda_j}\rho_j = \frac{\mathrm{d}}{\mathrm{d}\lambda_j}\rho_j.$$

The Leibniz integral rule is used to interchange the integral and the derivative, for this $e_j(t)$ has to be continuous in both t and λ_j and $\frac{\partial e_j(t)}{\partial \lambda_j}$ has to be continuous in both t and λ_j. Then the gradient is

$$\begin{aligned}\nabla_{\lambda_j}\rho_j = 2\int_{t_0}^{t_f} e_j^T E \frac{\partial h}{\partial x}\left(x_j\right)\frac{\partial x}{\partial \lambda_j} + (u_j - u_{j-1}) U \tilde{e}_{j-1}^T \,\mathrm{d}t \\ + 2e_j^T(t_f) T \frac{\partial h}{\partial x}\left(x_j(t_f)\right)\frac{\partial x}{\partial \lambda_j}(t_f).\end{aligned}$$

To calculate $\frac{\partial x}{\partial \lambda_j}$, the fundamental theorem of Lebesgue integral calculus is used. Because $x(t)$ is continuously differentiable, it is also absolutely continuous and it can be expressed at

$$x(t) = x(t_0) + \int_{t_0}^{t} f(x(\tau), u(\tau))\,\mathrm{d}\tau.$$

Differentiation with respect to the learning gain yields

$$\begin{aligned}\frac{\partial x_j}{\partial \lambda_j} &= \frac{\partial}{\partial \lambda_j} x_j(t_0) + \frac{\partial}{\partial \lambda_j} \int_{t_0}^{t} f(x_j(\tau), u_j(\tau))\,\mathrm{d}\tau \\ &= \int_{t_0}^{t} \frac{\partial}{\partial \lambda_j} f(x_j(\tau), u_j(\tau))\,\mathrm{d}\tau \\ &= \int_{t_0}^{t} \frac{\partial f}{\partial x}(x_j(t), u_j(t)) \frac{\partial x_j}{\partial \lambda_j} + \frac{\partial f}{\partial u}(x_j(t), u_j(t)) \frac{\partial u_j}{\partial \lambda_j}\,\mathrm{d}\tau,\end{aligned}$$

if $f(x_j(\tau), u_j(\tau))$ and $\frac{\partial f}{\partial \lambda_j}(x_j(\tau), u_j(\tau))$ are continuous in τ and λ_j. Next, the time derivative is applied yielding

$$\begin{aligned}\frac{\partial}{\partial t}\frac{\partial x_j}{\partial \lambda_j} &= \frac{\partial}{\partial t} \int_{t_0}^{t} \frac{\partial f}{\partial x}(x_j(t), u_j(t)) \frac{\partial x_j}{\partial \lambda_j} + \frac{\partial f}{\partial u}(x_j(t), u_j(t)) \frac{\partial u_j}{\partial \lambda_j}\,\mathrm{d}\tau \\ &= \frac{\partial f}{\partial x}(x_j(t), u_j(t)) \frac{\partial x_j}{\partial \lambda_j} + \frac{\partial f}{\partial u}(x_j(t), u_j(t)) \tilde{e}_{j-1}(t)\end{aligned} \tag{6.1}$$

with the initial condition

$$\left.\frac{\partial x_j}{\partial \lambda_j}\right|_{t=t_0} = 0.$$

If $\frac{\partial f}{\partial x}(x_j(t), u_j(t))$ and $\frac{\partial f}{\partial u}(x_j(t), u_j(t))\tilde{e}_{j-1}(t)$ are piecewise continuous in t, then the sensitivity equation (6.1) has a unique solution over $[t_0, t_f]$ (see Theorem 3.2 in [22]). If further

$$\frac{\partial f}{\partial x}(x_j(t), u_j(t)) \frac{\partial x_j}{\partial \lambda_j} + \frac{\partial f}{\partial u}(x_j(t), u_j(t)) \tilde{e}_{j-1}(t)$$

is continuous in t, $\frac{\partial x_j}{\partial \lambda_j}$ and λ_j and locally Lipschitz in $\frac{\partial x_j}{\partial \lambda_j}$ (uniformly in t and λ_j), then the solution depends continuously on λ_j (see Theorem 3.5 in [22]). Now the error is examined on its continuity in λ_j. It is given by

$e_j = y_j - y_d = h(x_j) - y_d$ and continuous in λ_j if y_j is, which is the case if the input u_j is as well. The input in iteration j is given by

$$u_j = u_0 + \sum_{k=1}^{j} \lambda_k \tilde{e}_{k-1}$$

which is continuous in λ_j, and in t if u_0 is. Therefore, the gradient $\nabla_{\lambda_j}\rho_j$ of the policy value function exists and is continuous in λ_j for all $j \in \mathbb{N}$.

Second, the Hessian, that is the second derivative, of the policy value function is derived. The following assumptions encompass all requirements that are made in its derivation:

- The initial input signal $u_0(t)$ is continuous in t.
- The state dynamics $f(x,u)$ are twice continuously differentiable in both arguments.
- The output equation $h(x)$ is twice continuously differentiable.

To take the second derivative the differential and integral have to be switched again and the Leibniz integral rule is applied once more under the conditions that the integrand and its partial derivative with respect to λ_j are continuous in t and λ_j. Then the second derivative is

$$\begin{aligned}\frac{\mathrm{d}^2\rho_j}{\mathrm{d}\lambda_j^2} = 2\int_{t_0}^{t_f} &\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j}\right)^T E \frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j} + e_j^T E\left(\frac{\partial^2 h}{\partial x^2}\left(\frac{\partial x}{\partial \lambda_j}\right)^2 + \frac{\partial h}{\partial x}\frac{\partial^2 x}{\partial \lambda_j^2}\right) \\ &+ \tilde{e}_{j-1} U \tilde{e}_{j-1}^T \,\mathrm{d}t \\ + 2\left(\left(\frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j}\right)^T T \frac{\partial h}{\partial x}\frac{\partial x}{\partial \lambda_j} + e_j^T T\left(\frac{\partial^2 h}{\partial x^2}\left(\frac{\partial x}{\partial \lambda_j}\right)^2 + \frac{\partial h}{\partial x}\frac{\partial^2 x}{\partial \lambda_j^2}\right)\right)&\Bigg|_{t=t_f}.\end{aligned}$$

An additional second initial value problem has to be solved in order to determine $\frac{\partial^2 x}{\partial \lambda^2}(t)$. It can be derived by starting with

$$\frac{\partial}{\partial \lambda_j}\frac{\partial x_j}{\partial \lambda_j} = \frac{\partial}{\partial \lambda_j}\int_{t_0}^{t} \frac{\partial f}{\partial x}(x_j(t), u_j(t))\frac{\partial x_j}{\partial \lambda_j} + \frac{\partial f}{\partial u}(x_j(t), u_j(t))\frac{\partial u_j}{\partial \lambda_j}\,\mathrm{d}\tau.$$

If the integrand and its partial derivative with respect to λ_j are continuous in t and λ_j, derivation and integration can be switched and differentiation with respect to t afterwards yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial^2 x}{\partial \lambda^2} = \frac{\partial f}{\partial x}\frac{\partial^2 x}{\partial \lambda^2} + \frac{\partial^2 f}{\partial x^2}\left(\frac{\mathrm{d}x}{\mathrm{d}\lambda_j}\right)^T \frac{\mathrm{d}x}{\mathrm{d}\lambda_j} + \frac{\partial^2 f}{\partial x \partial u}\tilde{e}_{j-1}^T \left(\frac{\mathrm{d}x}{\mathrm{d}\lambda_j}\right)^T \\ + \frac{\partial^2 f}{\partial u \partial x}\frac{\mathrm{d}x}{\mathrm{d}\lambda_j}\tilde{e}_{j-1} + \frac{\partial^2 f}{\partial u^2}\tilde{e}_{j-1}\tilde{e}_{j-1}^T, \qquad (6.2a)\end{aligned}$$

$$\frac{\partial^2 x}{\partial \lambda^2}(t_0) = 0. \qquad (6.2b)$$

If the right side of equation (6.2a) is continuous in t, $\frac{\partial^2 x}{\partial \lambda^2}$ and λ_j and locally Lipschitz in $\frac{\partial^2 x}{\partial \lambda^2}$ (uniformly in t and λ_j), then the solution exists on $[t_0, t_f]$ and depends continuously on λ_j (Theorem 3.5 in [22]). With that, the second derivative of the policy value function also exists and is continuous in λ_j for all $j \in \mathbb{N}$.

6.2 Modelling of the Double Pendulum on a Cart

A model of the double pendulum on a cart is derived using Lagrangian mechanics. A diagram of the system is depicted in Figure 6.1. To simplify the equations it is assumed that the pendulums themselves are weightless and have the point masses m_2 and m_3 fixed to their ends. The configuration of the system can be uniquely described by the cart's position x, the angle of the closer pendulum φ and the angle of the further pendulum ψ and together they constitute the generalised coordinates

$$q = \begin{bmatrix} x \\ \varphi \\ \psi \end{bmatrix}. \qquad (6.3)$$

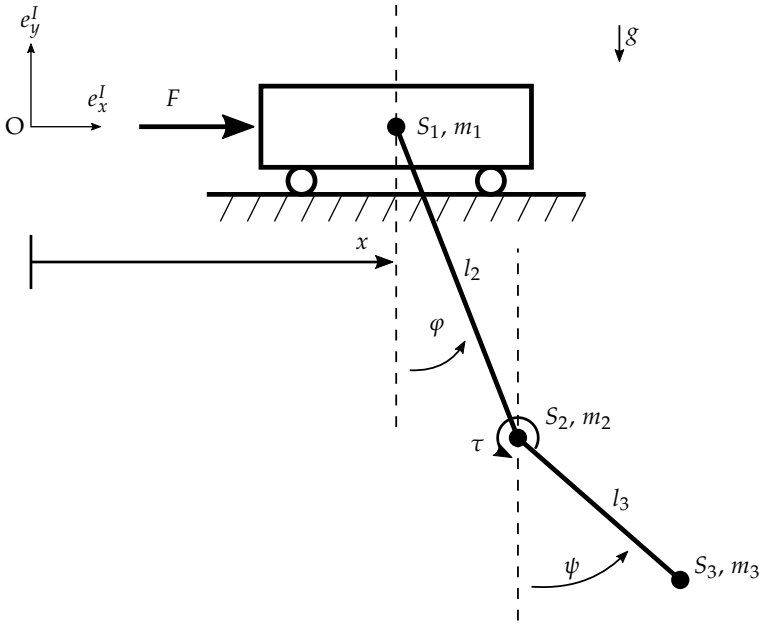

Figure 6.1: Diagram of the double pendulum on a cart. The force F and torque τ are applied to the system. It is assumed that the rods are weightless and have point masses m_2 and m_3 fixed to their ends. The cart's mass is assumed to be in the concentrated in the point mass m_1 as well.

In the inertial frame of reference I, the position and velocities of the centres of mass are

$$
\begin{aligned}
{}_{\mathrm{I}}r_{OS_1} &= \begin{bmatrix} x \\ 0 \end{bmatrix}, & {}_{\mathrm{I}}v_{S_1} &= \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix}, \\
{}_{\mathrm{I}}r_{OS_2} &= \begin{bmatrix} x + l_2 \sin(\varphi) \\ -l_2 \cos(\varphi) \end{bmatrix}, & {}_{\mathrm{I}}v_{S_2} &= \begin{bmatrix} \dot{x} + l_2 \dot{\varphi} \cos(\varphi) \\ l_2 \dot{\varphi} \sin(\varphi) \end{bmatrix}, \\
{}_{\mathrm{I}}r_{OS_3} &= \begin{bmatrix} x + l_2 \sin(\varphi) + l_3 \sin(\psi) \\ -l_2 \cos(\varphi) - l_3 \cos(\psi) \end{bmatrix}, & {}_{\mathrm{I}}v_{S_3} &= \begin{bmatrix} \dot{x} + l_2 \dot{\varphi} \cos(\varphi) + l_3 \dot{\psi} \cos(\psi) \\ l_2 \dot{\varphi} \sin(\varphi) + l_3 \dot{\psi} \sin(\psi) \end{bmatrix}.
\end{aligned}
$$

The kinetic energy is

$$
\begin{aligned}
T &= \frac{1}{2} m_1 \, {}_{\mathrm{I}}v_{S_1}^{\,T} \, {}_{\mathrm{I}}v_{S_1} + \frac{1}{2} m_2 \, {}_{\mathrm{I}}v_{S_2}^{\,T} \, {}_{\mathrm{I}}v_{S_2} + \frac{1}{2} m_3 \, {}_{\mathrm{I}}v_{S_3}^{\,T} \, {}_{\mathrm{I}}v_{S_3} \\
&= \frac{1}{2}(m_1 + m_2 + m_3)\dot{x}^2 + \frac{1}{2} l_2^2 (m_2 + m_3)\dot{\varphi}^2 + \frac{1}{2} m_3 l_3^2 \dot{\psi}^2 \\
&\quad + (m_2 + m_3) l_2 \cos(\varphi) \dot{x} \dot{\varphi} + m_3 l_3 \cos(\psi) \dot{x} \dot{\psi} + m_3 l_2 l_3 \cos(\varphi - \psi) \dot{\varphi} \dot{\psi}
\end{aligned} \tag{6.4}
$$

and the potential energy is

$$
V = -\left((m_2 + m_3) l_2 \cos(\varphi) + m_3 l_3 \cos(\psi)\right) g. \tag{6.5}
$$

The equations of motion are then described by the Euler-Lagrange equation

$$
\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial T}{\partial \dot{q}} - \frac{\partial T}{\partial q} + \frac{\partial V}{\partial q} = f_{\mathrm{NP}} \tag{6.6}
$$

with $f_{\mathrm{NP}} = \begin{bmatrix} F & 0 & \tau \end{bmatrix}^T$. This leads to the three equations

$$
\begin{aligned}
F &= (m_1 + m_2 + m_3)\ddot{x} + (m_2 + m_3) l_2 \cos(\varphi) \ddot{\varphi} \\
&\quad - (m_2 + m_3) l_2 \sin(\varphi) \dot{\varphi}^2 + m_3 l_3 \cos\psi \ddot{\psi} \\
&\quad - m_3 l_3 \sin(\psi) \dot{\psi}^2, \\
0 &= l_2^2 (m_2 + m_3)\ddot{\varphi} + (m_2 + m_3) l_2 \cos(\varphi) \ddot{x} \\
&\quad + m_3 l_2 l_3 \cos(\varphi - \psi)\ddot{\psi} + m_3 l_2 l_3 \sin(\varphi - \psi)\dot{\psi}^2 \\
&\quad + (m_2 + m_3) l_2 \sin(\varphi) g, \\
\tau &= m_3 l_3^2 \ddot{\psi} + m_3 l_3 \cos(\psi) \ddot{x} + m_3 l_2 l_3 \cos(\varphi - \psi)\ddot{\varphi} \\
&\quad - m_3 l_2 l_3 \sin(\varphi - \psi)\dot{\varphi}^2 + m_3 l_3 \sin(\psi) g,
\end{aligned}
$$

which can be solved for the following three equations:

$$\begin{aligned}
\ddot{x} = & \frac{m_2 + m_3 \sin^2(\varphi - \psi)}{m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)} F \\
& - \frac{(m_2 + m_3)\sin(\varphi)\sin(\varphi - \psi)}{m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)} \frac{\tau}{l_3} \\
& + \frac{m_2(m_2 + m_3) l_2 \sin(\varphi)}{m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)} \dot{\varphi}^2 \\
& + \frac{m_2 m_3 l_3 \sin(\varphi)\cos(\varphi - \psi)}{m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)} \dot{\psi}^2 \\
& + \frac{m_2(m_2 + m_3)\sin(\varphi)\cos(\varphi)}{m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)} g,
\end{aligned} \tag{6.7}$$

$$\begin{aligned}
\ddot{\varphi} = & \frac{-m_2 \cos(\varphi) + m_3 \sin(\psi)\sin(\varphi - \psi)}{l_2 \left(m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)\right)} F \\
& - \frac{m_1 \cos(\varphi - \psi) + (m_2 + m_3)\sin(\varphi)\sin(\psi)}{l_2 \left(m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)\right)} \frac{\tau}{l_3} \\
& - \frac{l_2 \left((m_2 + m_3) m_2 \sin(\varphi)\cos(\varphi) + m_1 m_3 \sin(\varphi - \psi)\cos(\varphi - \psi)\right)}{l_2 \left(m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)\right)} \dot{\varphi}^2 \\
& - \frac{l_3 m_3 \left(m_1 \sin(\varphi - \psi) + m_2 \sin(\varphi)\cos(\psi)\right)}{l_2 \left(m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)\right)} \dot{\psi}^2 \\
& - \frac{m_1 m_3 \cos(\psi) sin(\varphi - \psi) + m_2(m_1 + m_2 + m_3)\sin(\varphi)}{l_2 \left(m_1 \left(m_2 + m_3 \sin^2(\varphi - \psi)\right) + m_2(m_2 + m_3)\sin^2(\varphi)\right)} g,
\end{aligned} \tag{6.8}$$

$$
\begin{aligned}
\ddot{\psi} = & \frac{-(m_2+m_3)\sin(\varphi)\sin(\varphi-\psi)}{l_3\left(m_1\left(m_2+m_3\sin^2(\varphi-\psi)\right)+m_2(m_2+m_3)\sin^2(\varphi)\right)}F \\
& + \frac{(m_2+m_3)\left(m_1+(m_2+m_3)\sin^2(\varphi)\right)}{l_3\left(m_1\left(m_2+m_3\sin^2(\varphi-\psi)\right)+m_2(m_2+m_3)\sin^2(\varphi)\right)}\frac{\tau}{l_3 m_3} \\
& + \frac{(m_2+m_3)m_1 l_2\sin(\varphi-\psi)}{l_3\left(m_1\left(m_2+m_3\sin^2(\varphi-\psi)\right)+m_2(m_2+m_3)\sin^2(\varphi)\right)}\dot{\varphi}^2 \\
& + \frac{m_1 m_3 l_3\sin(\varphi-\psi)cos(\varphi-\psi)}{l_3\left(m_1\left(m_2+m_3\sin^2(\varphi-\psi)\right)+m_2(m_2+m_3)\sin^2(\varphi)\right)}\dot{\psi}^2 \\
& + \frac{(m_2+m_3)m_1\cos(\varphi)\sin(\varphi-\psi)}{l_3\left(m_1\left(m_2+m_3\sin^2(\varphi-\psi)\right)+m_2(m_2+m_3)\sin^2(\varphi)\right)}g.
\end{aligned}
\tag{6.9}
$$

If the state vector

$$
z = \begin{bmatrix} x \\ \varphi \\ \psi \\ \dot{x} \\ \dot{\varphi} \\ \dot{\psi} \end{bmatrix}
$$

is introduced, then the state space is

$$
\dot{z} = f(z,F,\tau) = \begin{bmatrix} \dot{x} \\ \dot{\varphi} \\ \dot{\psi} \\ f_4(z,F,\tau) \\ f_5(z,F,\tau) \\ f_6(z,F,\tau) \end{bmatrix}
$$

where f_4 is equation (6.7), f_5 is (6.8) and f_6 is (6.9).

6.3 Derivation of the Reference Trajectory for the Double Pendulum on a Cart

The reference trajectory for the double pendulum on a cart in Section 4.1 is designed by naively applying an LQR controller. First, a preliminary reference trajectory for the states $\bar{z}_d$ is designed. The idea is to move the cart

from $\bar{x}_d(0) = -1$ to $\bar{x}_d(10) = 0$ in 10 seconds while the double pendulum is hanging down without moving at all, that is $\bar{\varphi}_d(t) \equiv \bar{\psi}_d \equiv 0$. A polynomial is designed for the cart's position and its velocity

$$\bar{x}_d(t) = -1 + \left(\frac{3}{10^2} t^2 - \frac{2}{10^3} t^3 \right), \tag{6.10}$$

$$\dot{\bar{x}}_d(t) = \frac{6}{10^2} t - \frac{6}{10^3} t^2. \tag{6.11}$$

Because the input that realises these trajectories is unknown, the force and torque are naively assumed to be zero, that is $\bar{F}_d \equiv \bar{\tau} \equiv 0$. Then a linear time-varying system is derived by linearising the state dynamics $f(z, u)$ of the system about this preliminary reference trajectory

$$\dot{\bar{z}} = \underbrace{\frac{\partial f}{\partial z}(\bar{z}_d, \bar{u}_d)}_{\coloneqq \bar{A}(t)} \bar{z} + \underbrace{\frac{\partial f}{\partial u}(\bar{z}_d, \bar{u}_d)}_{\coloneqq \bar{B}(t)} \bar{u}, \tag{6.12a}$$

$$\bar{z}(0) = \bar{z}_d(0). \tag{6.12b}$$

Now a LQR is designed with the cost

$$\bar{J}_{\text{LQR}} = \int_0^{10} \bar{z}^T Q \bar{z} + \bar{u}^T R \bar{u} \, \mathrm{d}t + \bar{z}(10)^T S \bar{z}(10) \tag{6.13}$$

where the weighting matrices are

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix},$$

$$S = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

They correspond to the weighting matrices that are chosen in the example in Section 4.1. The optimal input is then

$$\bar{u}_{\mathrm{LQR}}(t) = -R^{-1}\bar{B}(t)^T\bar{P}(t)\bar{z}(t) \tag{6.14}$$

with $\bar{P}(t)$ as the solution of the Riccati differential equation

$$\dot{\bar{P}}(t) = \bar{P}(t)\bar{B}(t)R^{-1}\bar{B}(t)^T\bar{P}(t) - \bar{P}(t)\bar{A}(t) - \bar{A}^T(t)\bar{P}(t) - Q, \tag{6.15a}$$

$$\bar{P}(10) = S. \tag{6.15b}$$

The reference trajectory used in the example in Section 4.1 is the resulting trajectory when the feedback (6.14), which is thus the feasible input $u_d = \bar{u}_{\mathrm{LQR}}$, is applied to the system (6.12).

6.4 Modelling of the Double Pendulum

A model of the double pendulum depicted in Figure 6.2 is derived using Lagrangian mechanics. The angle of the closer pendulum φ and the angle of the further pendulum ψ constitute the generalised coordinates

$$q = \begin{bmatrix} \varphi \\ \psi \end{bmatrix}. \tag{6.16}$$

In the inertial frame of reference I, the position and velocities of the centres of mass are

$${}_{\mathrm{I}}r_{OS_1} = \begin{bmatrix} l_1\sin(\varphi) \\ -l_1\cos(\varphi) \end{bmatrix}, \qquad {}_{\mathrm{I}}v_{S_1} = \begin{bmatrix} \dot{\varphi}l_1\cos(\varphi) \\ \dot{\varphi}l_1\sin(\varphi) \end{bmatrix},$$

$${}_{\mathrm{I}}r_{OS_2} = \begin{bmatrix} l_1\sin(\varphi) + l_2\sin(\psi) \\ -l_1\cos(\varphi) - l_2\cos(\psi) \end{bmatrix}, \quad {}_{\mathrm{I}}v_{S_2} = \begin{bmatrix} \dot{\varphi}l_1\cos(\varphi) + \dot{\psi}l_2\cos(\psi) \\ \dot{\varphi}l_1\sin(\varphi) + \dot{\psi}l_2\sin(\psi) \end{bmatrix}.$$

The kinetic energy is

$$T = \frac{1}{2}(m_1 + m_2)l_1^2\dot{\varphi}^2 + \frac{1}{2}m_2l_2^2\dot{\psi}^2 + l_1l_2m_2\cos(\varphi - \psi)\dot{\varphi}\dot{\psi} \tag{6.17}$$

and the potential energy is

$$V = -\left((m_1 + m_2)l_1\cos(\varphi) + m_2l_2\cos(\psi)\right)g. \tag{6.18}$$

The Euler-Lagrange equation is

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial T}{\partial \dot{q}} - \frac{\partial T}{\partial q} + \frac{\partial V}{\partial q} = f_{\mathrm{NP}} \tag{6.19}$$

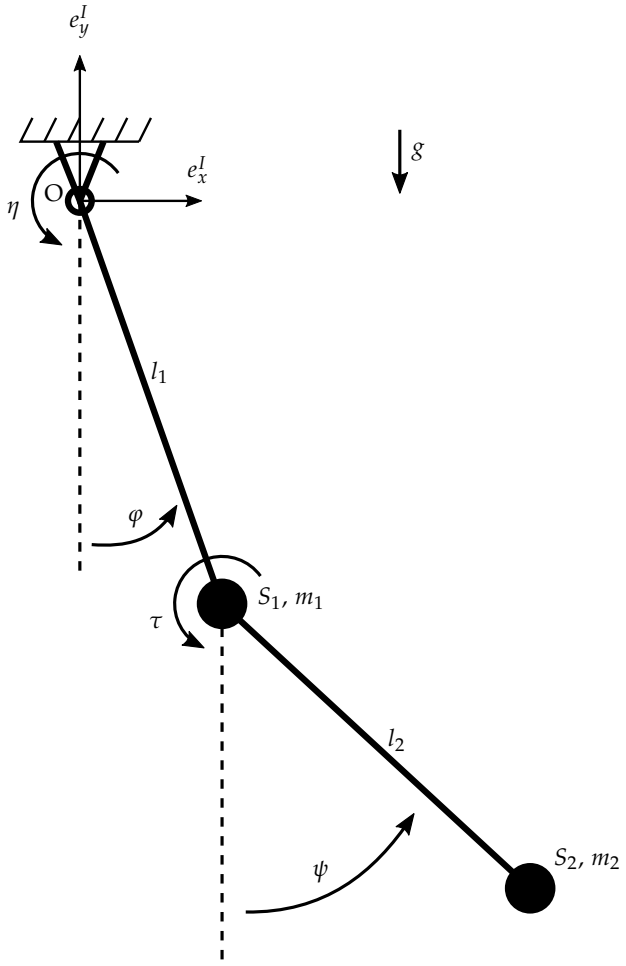

Figure 6.2: Diagram of the double pendulum. The rods of length l_1 and l_2 are assumed to be weightless with two point masses with weights m_1 and m_2 attached to their ends. The torques η and τ are the inputs to the system.

with $f_{\mathrm{NP}} = \begin{bmatrix} \eta & \tau \end{bmatrix}^T$. This leads to the equations of motion

$$\begin{aligned}
\ddot{\varphi} = & \frac{1}{l_1\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\frac{\eta}{l_1} \\
& - \frac{\cos(\varphi-\psi)}{l_1\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\frac{\tau}{l_2} \\
& - \frac{m_1\sin(\varphi) + m_2\cos(\psi)\sin(\varphi-\psi)}{l_1\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}g \\
& - \frac{l_1 m_2\sin(\varphi-\psi)\cos(\varphi-\psi)}{l_1\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\dot{\varphi}^2 \\
& - \frac{l_2 m_2\sin(\varphi-\psi)}{l_1\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\dot{\psi}^2,
\end{aligned} \tag{6.20}$$

$$\begin{aligned}
\ddot{\psi} = & -\frac{\cos(\varphi-\psi)}{l_2\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\frac{\eta}{l_1} \\
& + \frac{m_1+m_2}{l_2\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\frac{\tau}{m_2 l_2} \\
& + \frac{(m_1+m_2)\cos(\varphi)\sin(\varphi-\psi)}{l_2\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}g \\
& + \frac{(m_1+m_2)l_1\sin(\varphi-\psi)}{l_2\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\dot{\varphi}^2 \\
& + \frac{l_2 m_2\sin(\varphi-\psi)\cos(\varphi-\psi)}{l_2\left(m_1 + m_2\sin^2(\varphi-\psi)\right)}\dot{\psi}^2.
\end{aligned} \tag{6.21}$$

With the state vector

$$x = \begin{bmatrix} \varphi \\ \psi \\ \dot{\varphi} \\ \dot{\psi} \end{bmatrix}$$

the state space is

$$\dot{x} = f(x,\eta,\tau) = \begin{bmatrix} \dot{\varphi} \\ \dot{\psi} \\ f_3(x,\eta,\tau) \\ f_4(x,\eta,\tau) \end{bmatrix}$$

where f_3 is equation (6.20) and f_4 is equation (6.21).

7 Acknowledgements

Without a specific order, I thank Prof. Murat Arcak for generously hosting me at the fascinating and seemingly magical university that is UC Berkeley and the people in his lab for their warm welcome. Moreover, I thank Prof. Frank Allgöwer for letting me write my master's thesis abroad and for facilitating the contact. In addition, I thank Dr. Pierre-Jean Meyer for his helpful and excellent advice while writing this thesis. Also, thanks to Octavio Narvaez-Aroche, Stephen Tu and Prof. Andrew Packard for their input. Furthermore, I would like to thank DAAD for their financial support through PROMOS. Additionally, I thank Philipp Köhler for his organisational support. And last but not least, I want to thank Anja Köhler for her unwavering support throughout everything.

Glossary

ILC iterative learning control. 9, 10, 12–15, 17, 21–24, 31, 33, 35, 38–51, 54–57, 59, 60

LQR linear quadratic regulator. 18, 19, 24, 31, 33–45, 50, 51, 53–55, 57, 59, 68, 69

PD proportional and differentiating. 17, 23

PID proportional, integrating and differentiating. 17

RL reinforcement learning. 9, 10, 12, 21, 59

Bibliography

[1] H.-S. Ahn, Y. Chen, and K. L. Moore. "Iterative Learning Control: Brief Survey and Categorization". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), pp. 1099–1121.

[2] N. Amann and D. H. Owens. "Non-minimum phase plants in iterative learning control". In: *Second International Conference on 'Intelligent Systems Engineering'*. Second International Conference on 'Intelligent Systems Engineering' (Hamburg-Harburg, Germany, Sept. 5–9, 1994). IEE, 1994, pp. 107–112. ISBN: 0 85296 621 0. DOI: 10.1049/cp:19940610.

[3] N. Amann, D. H. Owens, and E. Rogers. "Iterative learning control using optimal feedback and feedforward actions". In: *International Journal of Control* 65.2 (1996), pp. 277–293. ISSN: 0020-7179. DOI: 10.1080/00207179608921697.

[4] N. Amann, D. H. Owens, and E. Rogers. "Norm-optimal predictive iterative learning control." In: *Proceedings of European Control Conference*. Vol. 4a. 1995, pp. 2880–2885.

[5] M. Arif and H. Inooka. "Iterative manual control model of human operator". eng. In: *Biological cybernetics* 81.5-6 (1999). Journal Article, pp. 445–455. ISSN: 0340-1200. DOI: 10.1007/s004220050574. eprint: 10592019.

[6] M. Arif, T. Ishihara, and H. Inooka. "Experience-Based Iterative Learning Controllers for Robotic Systems". In: *Journal of Intelligent and Robotic Systems* 35.4 (2002). PII: 5089404, pp. 381–396. ISSN: 09210296. DOI: 10.1023/A:1022399105710.

[7] S. Arimoto. "Mathematical Theory of Learning with Applications to Robot Control". In: *Adaptive and Learning Systems. Theory and Applications*. Ed. by K. S. Narendra. Boston, MA: Springer, 1986, pp. 379–388. ISBN: 978-1-4757-1897-3. DOI: 10.1007/978-1-4757-1895-9_27. URL: https://doi.org/10.1007/978-1-4757-1895-9_27.

[8] S. Arimoto, S. Kawamura, and F. Miyazaki. "Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems". In: *The 23rd IEEE Conference on Decision and Control*. The 23rd IEEE Conference on Decision and Control (Las Vegas, Nevada, USA,). IEEE, 12/12/1984 - 12/14/1984, pp. 1064–1069. DOI: 10.1109/CDC.1984.272176.

[9] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. "Bettering operation of Robots by learning". en. In: *Journal of Robotic Systems* 1.2 (1984), pp. 123–140. ISSN: 1097-4563. DOI: 10.1002/rob.4620010203. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/rob.4620010203.

[10] A. Azenha. "Iterative learning in variable structure position/force hybrid control of manipulators". In: *Robotica* 18 (2000), pp. 213–217. (Visited on 08/22/2018).

[11] D. A. Bristow, M. Tharayil, and A. G. Alleyne. "A Survey of Iterative Learning Control". In: *IEEE Control Systems* 26.3 (2006), pp. 96–114.

[12] C.-J. Chien and L.-C. Fu. "A neural network based learning controller for robot manipulators". In: *Proceedings of the 39th IEEE conference on decision and control*. 39th IEEE Conference on Decision and Control (Sydney, NSW, Australia,). IEEE. Control Systems Society. IEEE, 2000, pp. 1748–1753. ISBN: 0-7803-6638-7. DOI: 10.1109/CDC.2000.912114.

[13] C.-J. Chien and A. Tayebi. "Further results on adaptive iterative learning control of robot manipulators". In: *Automatica* 44.3 (2008), pp. 830–837. ISSN: 00051098. DOI: 10.1016/j.automatica.2007.06.023. URL: http://www.sciencedirect.com/science/article/pii/S000510980700338X.

[14] W. Cho, T. F. Edgar, and J. Lee. "Iterative learning dual-mode control of exothermic batch reactor". In: *2004 5th Asian Control Conference. Melbourne, Australia, 20-23 July, 2004*. Institute of Electrical and Electronics Engineers. Piscataway N.J.: IEEE, 2004, pp. 1270–1275. ISBN: 0780388739.

[15] J.-Y. Choi, J. Uh, and J. S. Lee. "Iterative learning control of robot manipulator with I-type parameter estimator". In: *Proceedings of the 2001 American control conference*. Proceedings of American Control Conference (Arlington, VA, USA,). American Automatic Control Council and Institute of Electrical and Electronics Engineers. IEEE, 2001, 646–651 vol.1. ISBN: 0-7803-6495-3. DOI: 10.1109/ACC.2001.945620.

[16] S. Gopinath and I. N. Kar. "Iterative learning control scheme for manipulators including actuator dynamics". In: *Mechanism and Machine Theory* 39.12 (2004). PII: S0094114X04001533, pp. 1367–1384. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2004.05.021. (Visited on 08/22/2018).

[17] K. Hamamoto and T. Sugie. "Iterative learning control for robot manipulators using the finite dimensional input subspace". In: *Proceedings of the 40th IEEE conference on decision and control*. 40th Conference on Decision and Control (Orlando, FL, USA,). IEEE. Control Systems Society. IEEE, 2001, pp. 4926–4931. ISBN: 0-7803-7061-9. DOI: 10.1109/CDC.2001.980989.

[18] K. Hamamoto and T. Sugie. "Iterative learning control for robot manipulators using the finite dimensional input subspace". In: *IEEE Transactions on Robotics and Automation* 18.4 (2002), pp. 632–635. ISSN: 1042-296X. DOI: 10.1109/TRA.2002.801050.

[19] Y.-C. Haung et al. "Use of PID and Iterative Learning Controls on Improving Intra-Oral Hydraulic Loading System of Dental Implants". In: *JSME International Journal Series C* 46.4 (2003), pp. 1449–1455. ISSN: 1344-7653. DOI: 10.1299/jsmec.46.1449.

[20] H. Havlicsek and A. Alleyne. "Nonlinear control of an electrohydraulic injection molding machine via iterative adaptive learning". In: *IEEE/ASME Transactions on Mechatronics* 4.3 (1999), pp. 312–323. ISSN: 10834435. DOI: 10.1109/3516.789689.

[21] C.-T. Hsu, C.-J. Chien, and C.-Y. Yao. "A new algorithm of adaptive iterative learning control for uncertain robotic systems". In: *2003 IEEE international conference on robotics and automation*. IEEE International Conference on Robotics and Automation. IEEE ICRA 2003 Conference Proceedings (Taipei, Taiwan,). IEEE. Robotics and Automation Society. IEEE, 2003, pp. 4130–4135. ISBN: 0-7803-7736-2. DOI: 10.1109/ROBOT.2003.1242232.

[22] H. K. Khalil. *Nonlinear Systems*. 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[23] B. H. Lam, S. K. Panda, and J.-X. Xu. "Reduction of periodic speed ripples in PM synchronous motors using iterative learning control". In: *IECON 2000. 26th annual conference of the IEEE Industrial Electronics Society*. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation (Nagoya, Japan,). IEEE. Industrial Electronics. IEEE, 2000, pp. 1406–1411. ISBN: 0-7803-6456-2. DOI: 10.1109/IECON.2000.972327.

[24] Murray Garden. "Learning control of actuators in control systems". US3555252A (USA). Jan. 12, 1971.

[25] O. Narvaez-Aroche et al. "Reachability Analysis for Robustness Evaluation of the Sit-To-Stand Movement for Powered Lower Limb Orthoses". In: *Proceedings of the ASME 2018 Dynamic Systems and Control Conference*. Dynamic Systems and Control Conference (Atlanta, Georgia, USA,). American Society of Mechanical Engineers. 2018.

[26] Octavio Narvaez-Aroche et al. *Robust Control of the Sit-to-Stand Movement for a Powered Lower Limb Orthosis*. Nov. 16, 2018. URL: https://arxiv.org/pdf/1811.07011.

[27] David H. Owens. *Iterative Learning Control*. London: Springer London, 2016. ISBN: 978-1-4471-6770-9. DOI: 10.1007/978-1-4471-6772-3.

[28] K.-H. Park and Z. Bien. "A generalized iterative learning controller against initial state error". In: *International Journal of Control* 73.10 (2000), pp. 871–881. ISSN: 0020-7179. DOI: 10.1080/002071700405851.

[29] Stuart J. Russell and Peter Norvig. *Artificial intelligence. A modern approach*. 3rd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010. xviii, 1132. ISBN: 0136042597.

[30] N. C. Sahoo, J. X. Xu, and S. K. Panda. "Low Torque Ripple Control of Switched Reluctance Motors Using Iterative Learning". In: *IEEE Power Engineering Review* 21.12 (2001), p. 66. ISSN: 0272-1724. DOI: 10.1109/MPER.2001.4311224.

[31] D.-M. Shin, J.-Y. Choi, and J. S. Lee. "A P-type Iterative Learning Controller for Uncertain Robotic Systems with Exponentially Decaying Error Bounds". In: *Journal of Robotic Systems* 20.2 (2003), pp. 79–91. ISSN: 07412223. DOI: 10.1002/rob.10069.

[32] D. Sun and J. K. Mills. "Performance Improvement of Industrial Robot Trajectory Tracking Using Adaptive-Learning Scheme". In: *Journal of Dynamic Systems, Measurement, and Control* 121.2 (1999), p. 285. ISSN: 0018-9286. DOI: 10.1115/1.2802467.

[33] M. Sun and D. Wang. "Iterative learning control with initial rectifying action". In: *Automatica* 38.7 (2002), pp. 1177–1182. ISSN: 00051098. DOI: 10.1016/S0005-1098(02)00003-1. URL: http://www.sciencedirect.com/science/article/pii/S0005109802000031.

[34] A. Tayebi. "Adaptive iterative learning control for robot manipulators". In: *Automatica* 40.7 (2004). PII: S0005109804000597, pp. 1195–1203. ISSN: 00051098. DOI: 10.1016/j.automatica.2004.01.026.

[35] Masaki Togai and Osamu Yamano. "Analysis and design of an optimal learning control scheme for industrial robots: A discrete system approach". In: *1985 24th IEEE Conference on Decision and Control*. 1985 24th IEEE Conference on Decision and Control (Fort Lauderdale, FL, USA,). IEEE, 12/11/1985 - 12/13/1985, pp. 1399–1404. DOI: 10.1109/CDC.1985.268741.

[36] J. X. Xu et al. "Improved PMSM pulsating torque minimization with iterative learning and sliding mode observer". In: *IECON 2000. 26th annual conference of the IEEE Industrial Electronics Society*. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation (Nagoya, Japan,). IEEE. Industrial Electronics. IEEE, 2000, pp. 1931–1936. ISBN: 0-7803-6456-2. DOI: 10.1109/IECON.2000.972571.

[37] Xuan Yang and Xiao'e Ruan. "Robustness of reinforced gradient-type iterative learning control for batch processes with Gaussian noise". In: *Chinese Journal of Chemical Engineering* 24.5 (2016), pp. 623–629. ISSN: 1004-9541. DOI: 10.1016/j.cjche.2015.12.011. URL: http://www.sciencedirect.com/science/article/pii/S1004954115004619.

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich, Matthias Hirche, die vorliegende Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäß entlehnte Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen. Weiterhin bestätige ich, dass das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

_____ _____

Ort, Datum Unterschrift