Design of Norm-Optimal Iterative Learning Controllers: The Effect of an Iteration-Domain Kalman Filter for Disturbance Estimation

Nicolas Degen

ETH Zürich, 8092 Zürich

Angela Schoellig

UTIAS at University of Toronto, Toronto ON

Abstract

Iterative learning control (ILC) has proven to be an effective method for improving the performance of repetitive control tasks. In a first step, this paper revisits the comparison of three optimization-based ILC algorithms: *(i)* the popular quadratic optimal ILC law (QILC), *(ii)* an estimation-enhanced QILC algorithm (E-QILC) and *(iii)* an estimation-based ILC law based on an iteration-domain Kalman filter (K-ILC). The comparison is extended by including a nonlinear ILC approach (NILC) *(iv)*. In a second step, a more profound analysis of the influence of the tuning parameters, focusing mainly on the QILC and K-ILC approaches is provided including design guidelines, highlighting the advantages obtained from the Kalman-filter based algorithm from the use of a stochastic model. We especially conclude that the estimation-enhanced algorithm with its *iteration-varying optimal* Kalman gains can achieve both fast initial convergence and good noise rejection by optimally adapting the learning update rule over the course of an experiment.

To benchmark the algorithms we used a simulation of a single-input single-output mass-spring-damper system, from which the provided design guidelines for the algorithms are based on.

Key words: ILC, Kalman Filter, Iterative Learning Control

1 Introduction

1.1 ILC Background

In classic control problems, a controller reacts to measured disturbances to follow a desired trajectory. It is based on past measurements and sometimes enhanced with model-based prediction, which is also based only on past measurements. For a repetitive control task however, one can use the measured disturbance of the previous repetition to predict the new disturbance and thus react in advance to upcoming disturbances. This is exactly what iterative learning control (ILC) does.

In detail, ILC algorithms measure the tracking error of a previous iteration of the control task to predict a future tracking error. Based on a system dynamics model, the algorithm calculates an updated system input that compensates for the measured tracking error. As the new update was calculated based on a presumably imperfect model (due to linearization or neglected higher order dynamics for instance), the new input will in general only reduce the error, only approaching the optimal input iteration by iteration, steadily minimizing the tracking error. The algorithm can be thought of as a MIMO MPC controller acting on iteration domain, with the entire iteration trajectory as controlled state.

In contrast to comparable adaptive control schemes which use the 'learned' information to adapt parameters of an underlying feedback controller, ILC algorithms adapt a feed-through input to an arbitrary system. It may also be applied to a system that has a feedback controller integrated, stabilizing the system internally. In such cases, the feedback system reference signal often serves as input of the system (See Fig. 1) with the learning algorithm still able to improve the performance when dealing with iteration-constant disturbances or

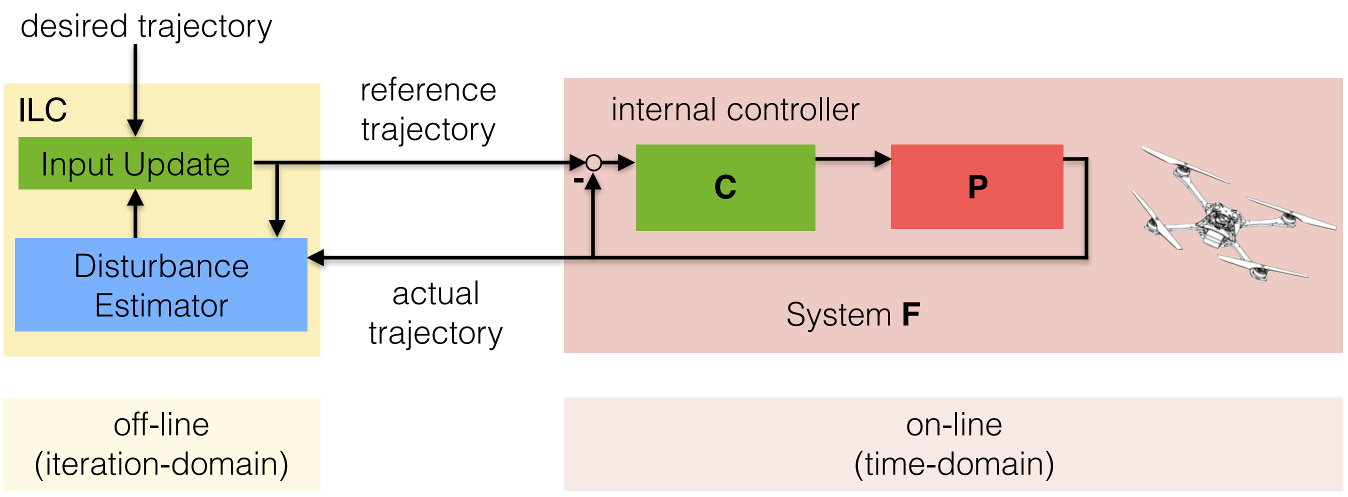

Fig. 1. A signal flow overview of the generalized learning algorithm. The signals outside the system **F** are iteration-domain and used for the learning while the internal signals are real time signals of one single learning iteration. As an example for a possible system **F**, a feedback-controlled quadrocopter is depicted. Iteration-wise, the system dynamics **F** map the input of the system $\mathbf{u}_j$ (with a feedback controlled system, as depicted here, this is the reference trajectory signal given to the feedback controller.) to its output (in this case the actually performed trajectory).

modeling errors in the feedback controller. Even optimal feedback controllers can be improved through ILC as it memorizes repeating disturbances that a real-time feedback cannot foresee. Note that ILC algorithms are in general not able to correct unstable systems as they can not react in real time to divergences. Nevertheless, sufficiently slow unstable states can theoretically be handled by the ILC learning, as the learned task are repetitive and have hence finite-time horizons we have a bound divergence of the unstable states.

1.2 Motivation

The authors' interest on iterative learning originates from the application on quadrocopter control [1]. In recent years, the improvements in battery technology and sensor costs have brought such vehicles a performance increase and reduced cost that led to a broad use in research. For such UAV, the aerodynamics, motion dynamics and electromechanical hardware behavior remain difficult to model and hence also challenging to control. To improve repeated flight maneuvers, ILC algorithms have been implemented successfully, improving the tracking performance of the drones substantially. The background of this paper was the goal to bring the ILC approach, as it was implemented for the quadrocopter, into context to other common ILC algorithms.

Preliminary results have been published [2], which covered mainly the analytic comparison of the algorithms. Also, the Propositions 1 to 5 have been taken over from the preceding paper. This paper now provides a more detailed performance analysis as well a discussion on the influence of ILC setup parameter choice.

1.3 Outline and Overview

In Sect. 2 we revisit four algorithms, introducing a unitized notation: *(i)* the widely-used quadratic norm optimal ILC (QILC) [3,4], *(ii)* an estimation-enhanced quadratic optimal ILC scheme (E-QILC) [5], *(iii)* the Kalman-filter based optimal ILC algorithm (K-ILC) as introduced by Schoellig in [1] and *(iv)* an additional nonlinear ILC formulation as presented by Volckaert *et al.* in [6] for comparison. The first three linear algorithms (see Figure 2) are thereby the main object of interest and have been treated in [2]. Primarily, the difference of the approaches is the method to *predict the tracking error of the next iteration*, which is in turn minimized in function of the next iteration's input. In *(ii)* and *(iii)* (E-QILC, K-ILC), an estimation algorithm is used for the error prediction, taking into account measurements of all past iterations. While *(ii)* uses an estimation gain that is constant for all iterations (as applied in [5]), the algorithm in *(iii)* uses a Kalman filter that adapts the algorithm from iteration to iteration according to its stochastic model.

The four algorithms discussed in the following are split in two steps, consisting of (A) an error prediction (Sect. 2.2) and (B) an input update step (Sect. 2.3). An analytical comparison of the different algorithm structure follows in Sect. 3. In Sect. 5 we provide one simulation example. It serves to study the different algorithms' performance and compare with the theoretical findings of the previous section. Based on that we formulate design guidelines for K-ILC and QILC algorithms.

1.4 Related Work

The first introduction of ILC algorithms to the research community was done by Arimoto *et al.* in 1984, where it is applied to a robotic manipulator arm. These early ILC solutions were oriented to classical control theory, implementing real time and analog controller tools such as a PID part in the ILC controller [7]. In 1996, Amann *et al.* introduced the concept of norm-optimal ILC algorithms. The learned input update is thereby based on a system model in order to optimize a cost criterion penalizing the predicted tracking error [8]. It is an approach that has since been used by various researchers [9,10,4,1]. In [11,5], the concept was extended by Lee *et al.*, where a Kalman estimator was proposed for the first time for the error estimation. The optimization formulation also enables the integration of input and output constraints as shown in [5,12,1]. Issues on convergence and robustness of the algorithm has been studied in [7,3,13,9,5,14]. An discussion on the systematic difficulties of robustness and convergence analysis has furthermore been presented by Longman *et al.* in [15]. Most recently, Son *et al.* have introduced a robust worst-case ILC algorithm guaranteeing robustness [16].

On the applied side, ILC algorithms have found vast application in industry. Its first uses were found in general robotic manipulators as presented in [7,9,17]. Bristow *et al.* showed in [13] the improved precision of a micro scale robotic deposition system using ILC learning, showing the performance improvements. Similar, actuators in semiconductor production are enhanced with ILC [10]. A widespread application is found in hard disk drives, were the positioning of the read head is iterative and controlled with ILC algorithms [18]. Its uses are also found in various process engineering: As presented by Cho *et al.*, the control of temperature in semiconductor production can be improved with the use of ILC algorithms [19]. It is also used for reactor process control, where the dynamics are essentially nonlinear[11,5].

The novel application of ILC learning on UAV trajectory control was first introduced by Schoellig *et al.* to improve fast quadrocopter maneuver performance. The research was presented in various papers as[1,20,21], where a new stochastic model and adapted optimality criterion are discussed, or [22] which presents possibilities of multi-agent ILC learning.

The algorithms presented in those works is the background and motivation for the results presented in this paper. Schoellig introduced a new Kalman-filter-enhanced ILC scheme for the prediction of the next iteration's tracking error (which is used in the optimized cost function) based on a stochastic model of the system. In contrast to time-domain Kalman-based estimation, which has been proven not to be beneficial for ILC [23], the approach in [1,20] and previously in [5] applies the Kalman filter to iteration-domain variables.

For general overviews on the field of iterative learning control, its fundamentals and notation standards consider [3] and more recently the survey papers [5,4,24]. Most recently, Shen *et al.*[25] have published a survey on stochastic ILC. In contrast to Schoelligs approach however, the algorithms mentioned thereby are not based on a minimization of an error norm, similar to the early approach by Arimoto [7].

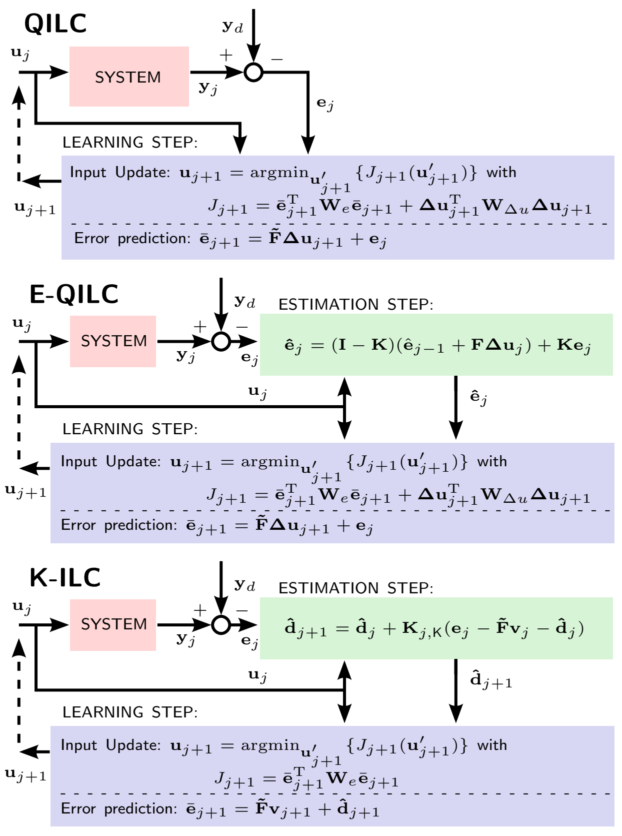
QILC
$\mathbf{u}_j$
SYSTEM
$\mathbf{y}_d$
$+$
$-$
$\mathbf{y}_j$
$\mathbf{e}_j$
LEARNING STEP:
Input Update: $\mathbf{u}_{j+1} = \text{argmin}_{\mathbf{u}'_{j+1}} \{J_{j+1}(\mathbf{u}'_{j+1})\}$ with
$J_{j+1} = \bar{\mathbf{e}}^{\mathrm{T}}_{j+1}\mathbf{W}_e\bar{\mathbf{e}}_{j+1} + \mathbf{\Delta u}^{\mathrm{T}}_{j+1}\mathbf{W}_{\Delta u}\mathbf{\Delta u}_{j+1}$
Error prediction: $\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{\Delta u}_{j+1} + \mathbf{e}_j$
$\mathbf{u}_{j+1}$
E-QILC
$\mathbf{u}_j$
SYSTEM
$\mathbf{y}_d$
$+$
$-$
$\mathbf{y}_j$
$\mathbf{e}_j$
ESTIMATION STEP:
$\hat{\mathbf{e}}_j = (\mathbf{I} - \mathbf{K})(\hat{\mathbf{e}}_{j-1} + \mathbf{F\Delta u}_j) + \mathbf{K}\mathbf{e}_j$
$\mathbf{u}_j$
$\hat{\mathbf{e}}_j$
LEARNING STEP:
Input Update: $\mathbf{u}_{j+1} = \text{argmin}_{\mathbf{u}'_{j+1}} \{J_{j+1}(\mathbf{u}'_{j+1})\}$ with
$J_{j+1} = \bar{\mathbf{e}}^{\mathrm{T}}_{j+1}\mathbf{W}_e\bar{\mathbf{e}}_{j+1} + \mathbf{\Delta u}^{\mathrm{T}}_{j+1}\mathbf{W}_{\Delta u}\mathbf{\Delta u}_{j+1}$
Error prediction: $\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{\Delta u}_{j+1} + \mathbf{e}_j$
$\mathbf{u}_{j+1}$
K-ILC
$\mathbf{u}_j$
SYSTEM
$\mathbf{y}_d$
$+$
$-$
$\mathbf{y}_j$
$\mathbf{e}_j$
ESTIMATION STEP:
$\hat{\mathbf{d}}_{j+1} = \hat{\mathbf{d}}_j + \mathbf{K}_{j,\mathrm{K}}(\mathbf{e}_j - \tilde{\mathbf{F}}\mathbf{v}_j - \hat{\mathbf{d}}_j)$
$\mathbf{u}_j$
$\hat{\mathbf{d}}_{j+1}$
LEARNING STEP:
Input Update: $\mathbf{u}_{j+1} = \text{argmin}_{\mathbf{u}'_{j+1}} \{J_{j+1}(\mathbf{u}'_{j+1})\}$ with
$J_{j+1} = \bar{\mathbf{e}}^{\mathrm{T}}_{j+1}\mathbf{W}_e\bar{\mathbf{e}}_{j+1}$
Error prediction: $\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{v}_{j+1} + \hat{\mathbf{d}}_{j+1}$
$\mathbf{u}_{j+1}$

Fig. 2. Learning algorithm architecture for linear or linearized systems **F**. **Top:** The popular quadratic optimal ILC (QILC) [4]. **Middle:** ILC algorithm with integrated estimation step (E-QILC) as presented by Lee *et al.* [5]. **Bottom:** Kalman-filter-based algorithm (K-ILC) presented by Schoellig *et al.* [1]. The dashed lines represent the input for the next iteration.

2 ILC Algorithm Overview

To compare the four considered algorithm, we separate one ILC algorithm iteration in two steps: (A) the error prediction and (B) the input update. The error prediction step predicts the next iteration's tracking error (see Figure 2) as a function of the new input $\bar{\mathbf{e}}_{j+1}(\mathbf{u}_{j+1})$, taking into account all past measured tracking errors and inputs $\mathbf{e}_j, .., \mathbf{e}_0, \mathbf{u}_j, .., \mathbf{u}_0$. The input update step then computes the input for the next iteration as an optimal solution of a cost function that uses the previously stated error prediction.

In this section, we first introduce the nominal system model and then state the different error prediction methods (Section 2.2) and input update algorithms (Section 2.3) as used in approaches *(i)-(iv)*.

2.1 Nominal System Model

We assume that the nominal behavior of the system $\mathbf{F}$ (cf. Figure 1) is modelled by the general, eventually nonlinear input-output relationship (c.f. [6]):

$$\mathbf{y}_j = \tilde{\mathbf{F}}(\mathbf{u}_j) : \begin{cases} x_j[k+1] = f(x_j[k], u_j[k]) \\ y_j[k] = g(x_j[k], u_j[k]) \end{cases}, \tag{1}$$

describing a dynamical system that may also be the discretization of a continuous-time model. This model maps a discrete-time input signal $\mathbf{u} = [u[1]^{\mathrm{T}}, ..., u[N]^{\mathrm{T}}]^{\mathrm{T}}$ to the corresponding lifted-domain output vector $\mathbf{y} = [y[m]^{\mathrm{T}}, ..., y[m+N]^{\mathrm{T}}]^{\mathrm{T}}$ via the nominal function $\tilde{\mathbf{F}}(\mathbf{u})$. In this context, N is the number of time steps per iteration and m the relative degree of the system. The index j indicates the signals (calculated, estimated or measured) of an iteration $j \in [0, 1, 2, \dots]$. We use $\tilde{(\cdot)}$ to denote the nominal model $\tilde{\mathbf{F}}$ and signals calculated with it. Similar notation can be found in [4,14,21].

From that model, the solution of finding the correct input $\mathbf{u}_j$ that tracks the desired trajectory $\mathbf{y}_d$ is the nominal input $\mathbf{u}_{\mathrm{nom}}$ fulfilling the $\mathbf{y}_d = \tilde{\mathbf{F}}(\mathbf{u}_{\mathrm{nom}})$. Typically, the nominal input $\mathbf{u}_{\mathrm{nom}}$ is chosen as the initial input guess $\mathbf{u}_0$. In practice, applying $\mathbf{u}_{\mathrm{nom}}$ to the real system will lead to a tracking error $\mathbf{e}_j = \mathbf{y}_j - \mathbf{y}_d$, where $\mathbf{y}_j$ denotes the measured output values. The tracking error is usually a result of unmodelled dynamics, unmodelled external disturbances and noise. The learning algorithm will now iteratively adapt the input $\mathbf{u}_j$ to decrease the tracking error $\mathbf{e}_j$.

Most ILC algorithms work with linear models of the system. A nonlinear model as(1) can be linearized around the desired trajectory to obtain a linear mapping with the full-rank matrix $\tilde{\mathbf{F}} \in \mathbb{R}^{N \times N}$, as described in [4,14]. This leads to a linear mapping of an entire trajectories input to the output:

$$\mathbf{y}_j = \tilde{\mathbf{F}}\mathbf{u}_j. \tag{2}$$

For the linear model, we can thus state the nominal error dynamics as

$$\tilde{\mathbf{e}}_j = \tilde{\mathbf{y}}_j - \mathbf{y}_d = \tilde{\mathbf{F}} \cdot (\mathbf{u}_j - \mathbf{u}_{\mathrm{nom}}) \equiv \tilde{\mathbf{F}}\mathbf{v}_j, \tag{3}$$

where we introduce the $\mathbf{v}_j$ as the difference of the actual to the nominal input.

2.2 Error Prediction (A)

In general, the predicted error of the next iteration, $\bar{\mathbf{e}}_{j+1}$, is a function of the new input $\mathbf{u}_{j+1}$ and can takes into account all previously measured iterations' errors $(\mathbf{e}_0, ..., \mathbf{e}_j)$ and applied inputs $(\mathbf{u}_0, ..., \mathbf{u}_j)$. We indicate the predicted value by a bar $\bar{(\cdot)}$.

2.2.1 QILC Error Prediction

The QILC algorithm error prediction can be obtained from mapping the change of the input into the output space (3):

$$\mathbf{e}_{j+1} - \mathbf{e}_j = \tilde{\mathbf{F}}(\mathbf{v}_{j+1} - \mathbf{v}_j) = \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{v}_{j+1}, \tag{4}$$

which, solved for $\mathbf{e}_{j+1}$, holds

$$^{\mathrm{QILC}}\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}} \underbrace{\boldsymbol{\Delta}\mathbf{v}_{j+1}}_{\boldsymbol{\Delta}\mathbf{u}_{j+1} = \mathbf{u}_{j+1} - \mathbf{u}_j} + \mathbf{e}_j. \tag{5}$$

2.2.2 E-QILC Error Prediction

E-QILC extends the prediction procedure of QILC by adding an estimation step [5]. The QILC prediction as given in (5) is used, substituting the measured error $\mathbf{e}_j$ with an estimated error of the last completed iteration $\hat{\mathbf{e}}_j$:

$$^{\text{E-QILC}}\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{u}_{j+1} + \hat{\mathbf{e}}_j. \tag{6}$$

The estimated error is obtained from

$$\hat{\mathbf{e}}_j = (\mathbf{I} - \mathbf{K})\underbrace{(\hat{\mathbf{e}}_{j-1} + \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{u}_j)}_{\bar{\mathbf{e}}_j} + \mathbf{K}\mathbf{e}_j, \tag{7}$$

where a general estimation gain $\mathbf{K}$ weighs the influence of the measured error and the previous prediction. As introduced in [5], an iteration-constant filter gain $\mathbf{K} = \lambda\mathbf{I}$ is used with $\lambda \in \mathbb{R}$ and $\mathbf{I}$ as the identity matrix of matching dimension.

2.2.3 K-ILC Error Prediction

The K-ILC algorithm as proposed in [1] estimates an additive disturbance vector $\mathbf{d}_{j+1}$, which represents the difference between nominal model $\tilde{\mathbf{F}}\mathbf{v}_{j+1}$ (3) and the real system output (i.e. systematic disturbances resulting from model errors along the trajectory as well as repeated external disturbances). This disturbance estimate is used for the error prediction:

$$^{\text{K-ILC}}\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{v}_{j+1} + \hat{\mathbf{d}}_{j+1}. \tag{8}$$

The disturbance estimate is obtained from a Kalman filter based on the following model:

$$\begin{aligned}\mathbf{d}_{j+1} &= \mathbf{d}_j + \omega_j \\ \mathbf{e}_j &= \tilde{\mathbf{F}}\mathbf{v}_j + \mathbf{d}_j + \mu_j,\end{aligned} \tag{9}$$

where $\omega_j \sim \mathcal{N}(0, \mathbf{E}_j)$ and $\mu_j \sim \mathcal{N}(0, \mathbf{H}_j)$ represent non-repetitive, zero-mean Gaussian noise with covariance $\mathbf{E}_j$ and $\mathbf{H}_j$, respectively. The matrices $\mathbf{E}_j$ and $\mathbf{H}_j$ model the expected measurement and process noise, respectively but can be seen as design parameters and have influence on the desired rate of adaptation of the learning. As a first approximation, a common choice is $\mathbf{H}_j = \eta\mathbf{I}, \mathbf{E}_j = \epsilon\mathbf{I}$; $\eta, \epsilon \in \mathbb{R}$.

The Kalman filter [1,26] provides the optimal disturbance estimate $\hat{\mathbf{d}}_{j+1}$ (and with that also the error estimate $\hat{\mathbf{e}}_{j+1}$) based on the stochastic model (9), taking into account all past measured tracking errors:

$$\hat{\mathbf{d}}_{j+1} = \hat{\mathbf{d}}_j + \mathbf{K}_{j,\mathrm{K}}(\mathbf{e}_j - \tilde{\mathbf{F}}\mathbf{v}_j - \hat{\mathbf{d}}_j) \tag{10}$$

$$\hat{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{v}_{j+1} + \hat{\mathbf{d}}_{j+1}. \tag{11}$$

The optimal *iteration-varying* Kalman gain $\mathbf{K}_{j,\mathrm{K}}$ is calculated recursively:

$$\begin{aligned}\mathbf{S}_j &= \mathbf{P}_j + \mathbf{E}_j \\ \mathbf{K}_{j,\mathrm{K}} &= \mathbf{S}_j(\mathbf{S}_j + \mathbf{H}_{j+1})^{-1} \\ \mathbf{P}_{j+1} &= (\mathbf{I} - \mathbf{K}_{j,\mathrm{K}})\mathbf{S}_j.\end{aligned} \tag{12}$$

$\mathbf{P}_0 = E[(\mathbf{d}_0 - \hat{\mathbf{d}}_0)(\mathbf{d}_0 - \hat{\mathbf{d}}_0)^{\mathrm{T}}]$ is the covariance matrix of the expected initial disturbance and can be viewed as a design parameter together with the covariances $\mathbf{H}_j, \mathbf{E}_j$. The initial disturbance estimate $\hat{\mathbf{d}}_0$ is typically modelled to be zero as the nominal model is ideally the best possible guess.

In contrast to the E-QILC approach, this approach is based on a stochastic model of the system and results in optimal iteration-varying gain matrices $\mathbf{K}_{j,\mathrm{K}}$ minimizing the estimation error variance (c.f. [20,27] for detailed derivations in applied examples).

Remark 1. *In the derivations below, we use the fact that for any arbitrary sequence of gains $\mathbf{K}_j$, there exist corresponding matrices $\mathbf{H}_j, \mathbf{E}_j$, and $\mathbf{P}_0$ such that* (12) *is satisfied and $\mathbf{K}_j$ can be interpreted as optimal Kalman gains $\mathbf{K}_{j,K}$.*

2.2.4 NILC Error Prediction

For the nominal, nonlinear model we have a tracking error given by

$$\mathbf{e}_j = \mathbf{F}(\mathbf{u}_j) - \mathbf{y}_d. \tag{13}$$

Even though the nonlinear nominal model has no linearization errors, there will still be modeling errors compared to the real system. To compensate for these, a disturbance term $\bar{\mathbf{d}}$ is added, similar to K-ILC, for the next iterations prediction step:

$$^{\mathrm{NILC}}\bar{\mathbf{e}}_{j+1} = \mathbf{F}(\mathbf{u}_{j+1}) - \mathbf{y}_d + \bar{\mathbf{d}}_{j+1}. \tag{14}$$

The disturbance can thereby be included as a disturbance parameter in the extended nonlinear function $\bar{\mathbf{F}}(\cdot,\cdot)$

$$^{\mathrm{NILC}}\bar{\mathbf{e}}_{j+1} = \bar{\mathbf{F}}(\mathbf{u}_{j+1}, \bar{\mathbf{d}}_{j+1}) - \mathbf{y}_d. \tag{15}$$

Unlike K-ILC, where the disturbance term is estimated, the predicted disturbance as presented in [6] is found by the minimization step of the model error based on last iteration's measurements:

$$\bar{\mathbf{d}}_{j+1} = \arg\min_{\bar{\mathbf{d}}'_{j+1}}\{\|\mathbf{y}_j - \bar{\mathbf{F}}(\mathbf{u}_j, \bar{\mathbf{d}}'_{j+1})\|_p\} \tag{16}$$

As presented by Volckaert *et al.* [6], a $p = 2$ norm is used.

2.3 Input Update (B)

The second step of the generalized ILC algorithm can be stated as a minimization problem of a given cost function that has the updated input as optimal solution. The general form of the cost function of the ILC types *(i)*-*(iv)* is given by:

$$\begin{aligned}J_{j+1}(\mathbf{u}_{j+1}) = {} & \|\mathbf{W}'_e\bar{\mathbf{e}}_{j+1}\|_p^a + \|\mathbf{W}'_{\Delta u}\boldsymbol{\Delta}\mathbf{u}_{j+1}\|_p^a \\ & + \|\mathbf{W}'_u\mathbf{u}_{j+1}\|_p^a + \|\mathbf{W}'_v\mathbf{v}_{j+1}\|_p^a,\end{aligned} \tag{17}$$

using p-vector norms to the power of a. The next iteration's input is then given by

$$\mathbf{u}_{j+1} = \underset{\mathbf{u}'_{j+1}}{\mathrm{argmin}}\{J_{j+1}(\mathbf{u}'_{j+1})\}. \tag{18}$$

The generalized cost function thereby penalizes the predicted tracking error $\bar{\mathbf{e}}_{j+1}$ (essential for the goal of ILC), the change of input $\boldsymbol{\Delta}\mathbf{u}_{j+1}$ from one iteration to the next, and the magnitude of the 'shifted' and absolute input, $\mathbf{v}_{j+1}$ and $\mathbf{u}_{j+1}$, with corresponding weightings. Most literature dealing with ILC considers the squared 2-norm ($[p, a] = 2$) [4]. In principle, as the optimization can be done numerically, any norm may be applied. Also, the optimization step allows the implementation of 'hard' constraints to be applied on the input and output signals as presented in 2.3.6 [5,12,21]. For an analytical comparison, we will focus on *unconstrained cost functions* with a 2-norm ($[p, a] = 2$) since in this case an explicit solution to the optimization problem can be found. In addition, the penalization of the inputs $\mathbf{u}_{j+1}$

and $\mathbf{v}_{j+1}$ will be omitted as it represents 'soft' input constraints. Such constraints are secondary objectives, which can be used to smoothen and minimize the input signals. It further makes a deterministic analysis more difficult without fundamentally affecting the core of the algorithm, its noise robustness and convergence properties. Overall, this leaves us with

$$J_{j+1} = \bar{\mathbf{e}}_{j+1}^{\mathrm{T}} \mathbf{W}_e \bar{\mathbf{e}}_{j+1} + \mathbf{\Delta u}_{j+1}^{\mathrm{T}} \mathbf{W}_{\Delta u} \mathbf{\Delta u}_{j+1}, \tag{19}$$

where $\mathbf{W}_\ell = \mathbf{W}_\ell'^{\mathrm{T}} \mathbf{W}_\ell', \ell \in \{e, \Delta u\}$, is positive definite and $\mathbf{W}_\ell = \mathbf{W}_\ell^{\mathrm{T}}$. The weighting matrices $\mathbf{W}_e$ and $\mathbf{W}_{\Delta u}$ are design parameters that are chosen to reflect the learning objectives.

2.3.1 QILC Input Update

We obtain the updated input $\mathbf{u}_{j+1}$ of the QILC algorithm by inserting ${}^{\mathrm{QILC}}\bar{\mathbf{e}}_{j+1}$ from (5) into the cost function (19). As the cost function is quadratic with positive definite weighting matrices, a minimizing input $\mathbf{u}_{j+1}$ is found by from $\frac{dJ_{j+1}}{d\mathbf{u}_{j+1}} = 0$. We obtain:

$${}^{\mathrm{QILC}}\mathbf{u}_{j+1} = \mathbf{u}_j - L\mathbf{e}_j \tag{20}$$

with

$$L = (\tilde{\mathbf{F}}^{\mathrm{T}} \mathbf{W}_e \tilde{\mathbf{F}} + \mathbf{W}_{\Delta u})^{-1} \tilde{\mathbf{F}}^{\mathrm{T}} \mathbf{W}_e. \tag{21}$$

The detailed derivation of (21) is presented in a separate publication, see [28]. ${}^{\mathrm{QILC}}\mathbf{u}_{j+1}$ can also be stated non-recursively as:

$${}^{\mathrm{QILC}}\mathbf{u}_{j+1} = \mathbf{u}_{\mathrm{nom}} - \textstyle\sum_{i=0}^{j} L\mathbf{e}_i. \tag{22}$$

A special case of QILC is model inversion, where $\mathbf{W}_{\Delta u}$ is not penalized. By only penalizing the error,

$$J_{j+1} = \bar{\mathbf{e}}_{j+1}^{\mathrm{T}} \mathbf{W}_e \bar{\mathbf{e}}_{j+1}, \tag{23}$$

we get $L = -\tilde{\mathbf{F}}^{-1}$; i.e. an inversion of the nominal model:

$$\mathbf{u}_{j+1} = \mathbf{u}_j - \tilde{\mathbf{F}}^{-1}\mathbf{e}_j. \tag{24}$$

2.3.2 E-QILC Input Update

As E-QILC is closely related to QILC, the input update is analogous (same L as in (21)) and

$${}^{\mathrm{E\text{-}QILC}}\mathbf{u}_{j+1} = \mathbf{u}_j + L\hat{\mathbf{e}}_j, \tag{25}$$

where the measured error is substituted by the estimated error (7).

2.3.3 K-ILC Input Update

For K-ILC, the explicit solution for the optimal update is

$$\begin{aligned}\mathbf{v}_{j+1} = {} & (\tilde{\mathbf{F}}^{\mathrm{T}} \mathbf{W}_e \tilde{\mathbf{F}} + \mathbf{W}_{\Delta u})^{-1} \\ & (-\tilde{\mathbf{F}}^{\mathrm{T}} \mathbf{W}_e^{\mathrm{T}} \hat{\mathbf{d}}_{j+1} + \mathbf{W}_{\Delta u}^{\mathrm{T}} \mathbf{v}_j)\end{aligned} \tag{26}$$

which is comparable to Norrlöf's result in [27]. The exact derivation of (26) is again found in [28].

For further analysis, we consider the case for which the estimation is used to enhance a pure model inversion; that is, we choose $\mathbf{W}_{\Delta u} = 0$. This represents the scheme as proposed by Schoellig *et al.* [1] without 'soft' constraints on $\mathbf{\Delta u}_{j+1}$. For this case, (26) is given as

$$\mathbf{v}_{j+1} = -\tilde{\mathbf{F}}^{-1}\hat{\mathbf{d}}_{j+1}, \tag{27}$$

which, for the jth iteration, can be transformed to $\tilde{\mathbf{F}}\mathbf{v}_j = -\tilde{\mathbf{F}}\tilde{\mathbf{F}}^{-1}\hat{\mathbf{d}}_j = -\hat{\mathbf{d}}_j$. Plugged into the estimation rule (10), we get

$$\hat{\mathbf{d}}_{j+1} = \hat{\mathbf{d}}_j + \mathbf{K}_{j,\mathrm{K}}\mathbf{e}_j, \tag{28}$$

from which we can derive a non-recursive equation for $\hat{\mathbf{d}}_{j+1}$:

$$\hat{\mathbf{d}}_{j+1} = \textstyle\sum_{i=0}^{j} \mathbf{K}_{j,\mathrm{K}}\mathbf{e}_i, \tag{29}$$

where we set $\hat{\mathbf{d}}_0 = 0$, which is the best guess we can make initially assuming that the real system behaves like the nominal system model of Sect. 2. Furthermore, we can write

$$\mathbf{v}_{j+1} = -\tilde{\mathbf{F}}^{-1}\hat{\mathbf{d}}_{j+1} = -\tilde{\mathbf{F}}^{-1} \textstyle\sum_{i=0}^{j} \mathbf{K}_{j,\mathrm{K}}\mathbf{e}_i. \tag{30}$$

Note that we refer to a given iteration-dependent sequence of gains as $\mathbf{K}_j$. Such a sequence can always be interpreted as optimal Kalman gains obtained from a Kalman filter with specific covariance models, see Remark 1.

2.3.4 NILC Input Update

Given the nonlinear model, a explicit solution of the cost function $J_{j+1}(\mathbf{u}_{j+1})$ (19) minimization is in general not available. The input update hence relies on numerical minimization of the cost function that uses the nonlinear prediction ${}^{\mathrm{NILC}}\bar{\mathbf{e}}_{j+1}$ from (15). As cost function subject to minimization we have

$$\begin{aligned}J_{j+1} = {} & (\bar{\mathbf{F}}(\mathbf{u}_{j+1}, \mathbf{d}_{j+1}) - \mathbf{y}_d)^{\mathrm{T}} \\ & \mathbf{W}_e(\bar{\mathbf{F}}(\mathbf{u}_{j+1}, \mathbf{d}_{j+1}) - \mathbf{y}_d) \\ & + \mathbf{\Delta u}_{j+1}^{\mathrm{T}} \mathbf{W}_{\Delta u} \mathbf{\Delta u}_{j+1} \\ & + \mathbf{u}_{j+1}^{\mathrm{T}} \mathbf{W}_u \mathbf{u}_{j+1}.\end{aligned} \tag{31}$$

2.3.5 General Weighting Matrices

Although the input update's cost function for the specific algorithms are presented with specific weightings, the cost functions can theoretically be extended with additional penalizations. The weighting matrix are only restricted to be positive definite and thus invertible. This is necessary to find a minimum in the unconstrained 2-norm case, which is in turn needed for the explicit analysis. In the following a short description on the effects of each weighting:

2.3.5.1 Error weighting $\mathbf{W}_e$ The weighting of the error $\mathbf{e}_{j+1}$ is the very essential part of the cost function as it leads to the minimization of the error. It can be set equal to I without loss of generality, as the ratio between the weighting matrices ultimately defines the optimal solution of the cost function. As shown in [29], the weighting matrix can also be used to penalize specific moments along the iteration time axis, leading to a non-Toeplitz matrix. This can be useful when the system has to pass through a specially sensitive region and the error penalization is reduced e.g. by also penalizing the input.

2.3.5.2 Input change penalization $\mathbf{W}_{\Delta u = \Delta v}$ By weighting the rate of change of the input we slow down the convergence of the algorithm. The benefit comes from reducing the sensitivity toward noise by penalizing large changes of the input. For ILC algorithms without an estimation step, penalizing the input change Δu is thus required.

2.3.5.3 Soft constraints $\mathbf{W}_u$ and $\mathbf{W}_v$ In contrast to the previous weighting matrices that do not change the converged solution, the soft constrains will affect it directly. They penalize deviations from zero input ($\mathbf{W}_u$) or from the initial guess $\mathbf{u}_g$ ($\mathbf{W}_v$), pushing the optimal input closer to 0 or $\mathbf{u}_g$ respectively. In practice the absolute input value is rather subject to hard constraints. However, a practical possibility for the soft constraints is to use a second derivative operator for $\mathbf{W}_u$ or $\mathbf{W}_v$ (in time-dimension $[k]$), such that a non-smooth input is penalized.

2.3.6 Constrained Minimization

The minimization steps allows to include constraints on the system. This is generally valuable in practice, as actuators are generally limited in power or force [11,1]. Especially for linear dynamics model, most constraints cannot be considered inherently in the models. The numerical minimization allows however to include hard constraints on the input and the trajectories or any other state of the system. Based on the description of Lee [5] a formulation of inequality expressions is presented to describe constrains on the input and predicted output of an entire iteration. In contrast to the 'soft' constraints from penalizing the input in the cost function (see 2.3.5), this 'hard' constrained formulation brings absolute constraints to the signals. The cost function is thereby optimized considering the inequality

$$\mathcal{C}\mathbf{u}_{j+1} \geq \mathbf{c}_{j+1}. \tag{32}$$

The inequality expression is obtained by rearranging the constrained expressions

$$\begin{aligned} \mathbf{y}_- - \mathbf{s}_{j+1} \leq \mathbf{y}_{j+1} &\leq \mathbf{y}_+ + \mathbf{s}_{j+1} \\ \mathbf{u}_- \leq \mathbf{u}_{j+1} &\leq \mathbf{u}_+ \\ \delta\mathbf{u}_- \leq \delta\mathbf{u}_{j+1} &\leq \delta\mathbf{u}_+ \\ \mathbf{\Delta u}_- \leq \mathbf{\Delta u}_{j+1} &\leq \mathbf{\Delta u}_+ \end{aligned} \tag{33}$$

giving for $\mathcal{C}$ and $\mathbf{c}_{j+1}$ the matrices

$$\mathcal{C} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \\ \mathbf{I} \\ -\mathbf{I} \\ \delta\mathbf{I} \\ -\delta\mathbf{I} \\ \tilde{\mathbf{F}} \\ -\tilde{\mathbf{F}} \end{bmatrix}, \qquad \mathbf{c}_{j+1} = \begin{bmatrix} \mathbf{u}_- \\ -\mathbf{u}_+ \\ \mathbf{I}\mathbf{u}_j + \mathbf{\Delta u}_- \\ -\mathbf{I}\mathbf{u}_j - \mathbf{\Delta u}_+ \\ \delta\mathbf{u}_- \\ -\delta\mathbf{u}_+ \\ \mathbf{y}_- - \mathbf{y}_0 - \mathbf{s}_{j+1} \\ -\mathbf{y}_+ + \mathbf{y}_0 - \mathbf{s}_{j+1} \end{bmatrix}, \tag{34}$$

where $\mathbf{u}_-, \mathbf{u}_+$ denote the corresponding lower and upper limit for the input $\mathbf{u}_{j+1}$, output $\mathbf{y}_{j+1}$, difference in inout $\mathbf{\Delta u}_{j+1}$ and the time step difference vector $\delta\mathbf{u}_{j+1}$. $\delta\mathbf{u}_{j+1}$ is introduced as $\mathbf{u}_j[k] - \mathbf{u}_j[k-1]$, being the difference along the time index. The matrix $\delta\mathbf{I}$ is thereby the differentiation operator on $\mathbf{u}_{j+1}$ such that $\delta\mathbf{I}\mathbf{u}_{j+1} = \delta\mathbf{u}_j$. $\mathbf{s}_j$ is an additional variable that is minimized. This can be achieved by adding an additional term in the cost function $J_{j+1}(\mathbf{u}_{j+1})$:

$$\bar{J}_{j+1}(\mathbf{u}_{j+1}, \mathbf{s}_{j+1}) = J_{j+1}(\mathbf{u}_{j+1}) + \mathbf{s}_{j+1}^{\mathrm{T}}\mathbf{W}_s\mathbf{s}_{j+1}$$

with the minimization step now being

$$[\mathbf{u}_{j+1}, \mathbf{s}_{j+1}] = \underset{\mathbf{u}'_{j+1}, \mathbf{s}'_{j+1}}{\operatorname{argmin}} \{\bar{J}_{j+1}(\mathbf{u}'_{j+1}, \mathbf{s}'_{j+1})\}.$$

Note that the presented constrains cannot describe limitation applying to states $x_j[k]$ of the systems internal states.

3 Comparison of the ILC Algorithms

The goal of this section is to illustrate the similarities and differences of the ILC algorithms introduced in Section 2. In particular, we are interested in understanding how the K-ILC scheme compares to the widely-used QILC and E-QILC schemes in terms of convergence and noise robustness. Although only iteration constant gains $\mathbf{K}$ have been proposed for E-QILC, we generalize it to iteration-varying $\mathbf{K}_j$ for the comparison.

3.1 Equivalency of K-ILC and E-QILC

Proposition 1. *For equal choice of gain matrices $\mathbf{K}_j$, $j = \{0, 1, 2, \dots\}$ and cost function* (19)*, the predicted errors $\bar{\mathbf{e}}_{j+1}$ and resulting inputs $\mathbf{u}_{j+1}$ are identical for K-ILC and E-QILC for $j = \{0, 1, 2, \dots\}$ (assuming equal initial conditions).*

Proof. The prediction of the next iteration's error for the E-QILC and K-ILC approach from Sect. 2 are:

$$^{\text{E-QILC}}\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{v}_{j+1} + \hat{\mathbf{e}}_j$$

and

$$^{\text{K-ILC}}\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{v}_{j+1} + \hat{\mathbf{d}}_{j+1}.$$

Rearranging the prediction of the K-ILC algorithm, we get

$$\begin{aligned} ^{\text{K-ILC}}\bar{\mathbf{e}}_{j+1} &= \tilde{\mathbf{F}}\mathbf{v}_{j+1} + \hat{\mathbf{d}}_{j+1} \\ &= \tilde{\mathbf{F}}(\mathbf{v}_{j+1} - \mathbf{v}_j + \mathbf{v}_j) + \hat{\mathbf{d}}_{j+1} \\ &= \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{v}_{j+1} + \underbrace{\tilde{\mathbf{F}}\mathbf{v}_j + \hat{\mathbf{d}}_{j+1}}_{= ^{\text{E-QILC}}\hat{\mathbf{e}}_j}. \end{aligned}$$

The new formulation suggests the equivalency of $^{\text{E-QILC}}\hat{\mathbf{e}}_j = \tilde{\mathbf{F}}\mathbf{v}_j + \hat{\mathbf{d}}_{j+1}$ [1]. To prove this assumption we substitute $^{\text{E-QILC}}\hat{\mathbf{e}}_j$ in the E-QILC estimation step (7), using $\boldsymbol{\Delta}\mathbf{u}_j = \boldsymbol{\Delta}\mathbf{v}_j$:

$$\begin{aligned} \hat{\mathbf{d}}_{j+1} + \tilde{\mathbf{F}}\mathbf{v}_j &= (\mathbf{I} - \mathbf{K}_{j,\text{K}})((\hat{\mathbf{d}}_j + \tilde{\mathbf{F}}\mathbf{v}_{j-1}) + \mathbf{F}\boldsymbol{\Delta}\mathbf{v}_j) \\ &\qquad + \mathbf{K}_{j,\text{K}}\mathbf{e}_j \\ \Leftrightarrow \hat{\mathbf{d}}_{j+1} &= \hat{\mathbf{d}}_j - \mathbf{K}_j(\hat{\mathbf{d}}_j + \tilde{\mathbf{F}}\mathbf{v}_j - \mathbf{e}_j). \end{aligned}$$

After some transformation and with the assumed matching initial conditions, i.e. $^{\text{K-ILC}}\hat{\mathbf{d}}_1 = ^{\text{E-QILC}}\hat{\mathbf{e}}_0$ [2], the estimation step is identical to (10). With the prediction used for the cost function being equal, the two algorithms become identical. □

Interpretation: Although both ILC schemes are based on different approaches, the formulation is equivalent for equal, arbitrary gain matrices $\mathbf{K}_j$ and equal cost functions $J_{j+1}(\mathbf{u}_{j+1})$. With that, any choice of weighting matrices set and hard constraints will have the same result provided it is the same for E-QILC and K-ILC.

The influence of the choice of estimation filter gain $\mathbf{K}_j$ and cost functions J_{j+1} are hence valid for E-QILC as well as K-ILC equivalently. In the following, we will thus only discuss K-ILC, with the results valid for both estimation-based learning algorithms.

3.2 Comparison of NILC and K-ILC

Proposition 2. *The nonlinear input update 2.3.4 applied on a linear system model is equivalent to the K-ILC input update with $\mathbf{K}_{j,K} = \mathbf{I}$.*

Proof. Applying Volckaerts solution[6] on a linear system model, the prediction for $\mathbf{e}_{j+1}$ will change from

$$^{\text{NILC}}\bar{\mathbf{e}}_{j+1} = \bar{\mathbf{F}}(\mathbf{u}_{j+1}, \mathbf{d}_{j+1}) - \mathbf{y}_d \tag{35}$$

to

$$^{\text{NILC}}\bar{\mathbf{e}}_{j+1} = \tilde{\mathbf{F}}\mathbf{u}_{j+1} + \mathbf{d}_{j+1} - \mathbf{y}_d. \tag{36}$$

Expressing it in terms of $\mathbf{v}_{j+1}$, we can directly compare it with the prediction for $\mathbf{e}_{j+1}$ in the K-ILC algorithm:

$$\begin{aligned} ^{\text{VNS}}\bar{\mathbf{e}}_{j+1} &= \underbrace{\tilde{\mathbf{F}}\mathbf{u}_{j+1} - \mathbf{y}_d}_{\tilde{\mathbf{F}}\mathbf{u}_{j+1} - \tilde{\mathbf{F}}\mathbf{u}_g = \tilde{\mathbf{F}}\mathbf{v}_{j+1}} + \mathbf{d}_{j+1} \\ \hline ^{\text{K-ILC}}\hat{\mathbf{e}}_{j+1} &= \tilde{\mathbf{F}}\mathbf{v}_{j+1} + \hat{\mathbf{d}}_{j+1} \end{aligned} \tag{37}$$

One can see that the only difference remaining is the disturbance term $\mathbf{d}_j$. Volckaert suggests a norm optimization to find the matching correction term of iteration j (see equation (16)), whereas the K-ILC rule uses a stochastic estimation. The approaches are thus comparable, the way of predicting $\mathbf{d}_{j+1}$ is generally even interchangeable. The optimal solution of Volckaerts $\mathbf{d}_{j+1}$ from equation (16) for a square norm in the linear case can be found easily as it can be set directly to zero:

$$\begin{aligned} 0 &= \mathbf{y}_j - \tilde{\mathbf{F}}\mathbf{u}_j - \mathbf{d}_{j+1} \\ \mathbf{d}_{j+1} &= -\tilde{\mathbf{F}}\mathbf{u}_j + \mathbf{y}_j \\ &= -\tilde{\mathbf{F}}\mathbf{u}_j + \mathbf{y}_j + \mathbf{y}_d - \mathbf{y}_d \\ &= \mathbf{e}_j - \tilde{\mathbf{F}}\mathbf{v}_j \end{aligned} \tag{38}$$

[1] Note that the shifted index is no error. It originates as the error prediction $\bar{\mathbf{e}}_{j+1}$ as hereby defined uses estimations with different index, i.e. $\tilde{\mathbf{F}}\mathbf{v}_j + ^{\text{K-ILC}}\hat{\mathbf{d}}_{j+1}$ uses input $\mathbf{v}_j$ and does not include the input update of iteration $j+1$ and is thus the estimate of the measurement $\mathbf{e}_j$ but with measurement update of $\hat{\mathbf{d}}_{j+1}$ compared to $\tilde{\mathbf{F}}\mathbf{v}_j + \hat{\mathbf{d}}_j$.

[2] The index difference is only a definition issue as we stick to Lee's *et al.* definition [5].

Which is exactly the same expression as for $\hat{\mathbf{d}}_{j+1}$ in the K-ILC algorithm when choosing a Kalman gain $\mathbf{K}_j = \mathbf{I}$ (c.f. Proposition 3). $\square$

Interpretation: The nonlinear solution as presented originally by Volckeaert *et al.* [6] can be interpreted in two points: At first, it is a generalization for nonlinear system, with an additive disturbance correction $\mathbf{d}_j$, similar to the K-ILC by Schoellig *et al.*. The second point is the prediction step of the that correction term: In contrast to the estimation with Kalman filter in K-ILC, the error is found adapting the iteration's input to only the measured output error.

3.3 *Comparing K-ILC and QILC with Identical Cost Functions*

Proposition 3. *Choosing* $\mathbf{K}_j = \mathbf{I}$ *(for all* j*) in the K-ILC estimation step, the resulting K-ILC algorithm is identical to the QILC approach given equal cost functions* (19)*.*

Proof. The equivalency of K-ILC and E-QILC being proven, comparing QILC to the E-QILC estimation covers both cases. Setting $\mathbf{K}_j = \mathbf{I}$ in the estimation rule (7), we get $\hat{\mathbf{e}}_j = \mathbf{e}_j$. The estimation for the error becomes thus the last measured error. Plugged into the prediction step of E-QILC, $\hat{\mathbf{e}}_j + \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{u}_{j+1} = \mathbf{e}_j + \tilde{\mathbf{F}}\boldsymbol{\Delta}\mathbf{u}_{j+1}$, which is identical to the QILC prediction, see (5). $\square$

Interpretation: Any arbitrary QILC algorithm is thus a special case of the K-ILC approach with *iteration-constant* gain $\mathbf{K}_j = \mathbf{I}$ and equal cost functions.
Proposition 4. *A system that has negligible output noise, and hence a modelled output noise covariance* $\mathbf{H}_j = 0 \ \forall j$ *results in an optimal Kalman gain* $\mathbf{K}_j = \mathbf{I} \ \forall j$*.*

Proof. If we analyze the Kalman filter equation in (12), we see that the gain matrix is $\mathbf{K}_j = \mathbf{I}$ in the case of $\mathbf{H}_{j+1} = 0$ or, more generally, $\mathbf{H}_{j+1} \ll \mathbf{P}_j + \mathbf{E}_j$ such that $(\mathbf{P}_j + \mathbf{E}_j)(\mathbf{P}_j + \mathbf{E}_j + \mathbf{H}_{j+1})^{-1} \simeq \mathbf{I}$. $\square$

Interpretation: If the output noise is zero, i.e. $\mathbf{H}_j = 0$, the Kalman filter completely trusts the measurements, resulting in $\mathbf{K}_j \simeq \mathbf{I}$. The covariance $\mathbf{H}_j$ models measurement and process noise. We conclude from Prop. 3 and 4 that QILC can be interpreted as a special case of the K-ILC algorithm modelled with *zero output noise* of the system.

3.4 *Comparing QILC and K-ILC with Typical Weighting Matrix Choice*

Note that all statements made in Sect. 3.3 are valid for general cost functions (17). In the following, we will consider the common cost function (19) with typical weighting matrix choice as found in literature. In particular, for the K-ILC, a standard choice is ${}^{\text{K-ILC}}\mathbf{W}_{\Delta u} = 0$. The reason behind it may be illustrated that $\mathbf{W}_{\Delta u}$ takes the role of a noise filter in QILC by minimizing the change in input based on the current error signal. Thus, in the QILC algorithm the cost function couples filtering (in the sense of noise sensitivity reduction through $\mathbf{W}_{\Delta u}$) and input update. In contrast to this, the K-ILC setup has a *separated* estimation filter as part of the error prediction step (A) and a 'clean' input update step with a cost function that is only defined by constraints independent of the noise robustness.

In order to compare QILC and K-ILC in this context, we make use of the two explicit formulas for the updated input $\mathbf{u}_{j+1}$, (22) and (30):

$$
\begin{aligned}
{}^{\text{QILC}}\mathbf{u}_{j+1} &= \mathbf{u}_{\text{nom}} - \textstyle\sum_{i=1}^{j} \underbrace{(\mathbf{W}_{\Delta u} + \tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e\tilde{\mathbf{F}})^{-1}\tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e}_{{}^{\text{QILC}}L}\mathbf{e}_j, \\
{}^{\text{K-ILC}}\mathbf{u}_{j+1} &= \mathbf{u}_{\text{nom}} - \textstyle\sum_{i=1}^{j} \underbrace{\tilde{\mathbf{F}}^{-1}\mathbf{K}_j}_{{}^{\text{K-ILC}}L}\mathbf{e}_j.
\end{aligned}
\tag{39}
$$

Assuming that we choose the same initial guess and nominal input $\mathbf{u}_{\text{nom}}$, the two approaches only differ in terms of ${}^{\text{QILC}}L$ and ${}^{\text{K-ILC}}L$:

$$
\begin{aligned}
{}^{\text{QILC}}L &= (\mathbf{W}_{\Delta u} + \tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e\tilde{\mathbf{F}})^{-1}\tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e, \\
{}^{\text{K-ILC}}L &= \tilde{\mathbf{F}}^{-1}\mathbf{K}_j.
\end{aligned}
\tag{40}
$$

We can compare both algorithms by analyzing the two matrices, ${}^{\text{QILC}}L$ and ${}^{\text{K-ILC}}L$. Note that the two matrices $\mathbf{W}_e$ and $\mathbf{W}_{\Delta u}$ determine ${}^{\text{QILC}}L$ and, therefore, twice as many parameters than degree of freedoms. We can hence set $\mathbf{W}_e = \mathbf{I}$ without loss of generality. Interestingly, with $\mathbf{W}_{\Delta u} = 0$ in the K-ILC setting, the influence of $\mathbf{W}_e$ cancels out.
Lemma 1. *By choosing the design parameters of the K-ILC algorithm* (12) *to be* $\mathbf{P}_0 = p_0\mathbf{X}, \mathbf{H}_j = \eta\mathbf{X}, \mathbf{E}_j = \epsilon\mathbf{X}$; $p_0, \eta, \epsilon \in \mathbb{R}$ *for arbitrary matrix* $\mathbf{X}$ *(commonly* $\mathbf{X} = \mathbf{I}$*), the Kalman gain* $\mathbf{K}_{j,K}$ *is diagonal and of the form* $\mathbf{K}_{j,K} = \lambda_j\mathbf{I}$*.*

Proof. Plugging $\mathbf{P}_0 = p_0\mathbf{X}, \mathbf{H}_j = \eta\mathbf{X}, \mathbf{E}_j = \epsilon\mathbf{X}$ into the Kalman gain equations (12), we see that $\mathbf{P}_j$ and $\mathbf{S}_j$ are of the form $\psi\mathbf{X},\ \psi \in \mathbb{R}$, and hence $\mathbf{K}_{j,\mathrm{K}} = \lambda_j \mathbf{X}\mathbf{X}^{-1}$, i.e. a scaled unity matrix. □

Interpretation: If the covariance of $\mathbf{H}_j, \mathbf{E}_j, \mathbf{P}_0$ are modelled such to be structurally similar, i.e. linear combinations of one single matrix, each iteration's gain $\mathbf{K}_{j,\mathrm{K}}$ will be diagonal ($\mathbf{K}_{j,\mathrm{K}} = \lambda_j \mathbf{I}$). Unless the modelled uncertainty of process noise $\mathbf{e}_j$ and output noise $\mathbf{H}_j$ a Kalman filter is hence useless.

Proposition 5. *For any given QILC with weightings as in* (19)*, there exists a corresponding, general estimation gain* $\mathbf{K}_j$ *resulting in an equivalent K-ILC algorithm with* $\mathbf{W}_{\Delta u} = 0$ *in the cost function* (19)*, i.e. a cost function as in* (23)*.*

Proof. Following directly from the equations (40), we can set ${}^{\mathrm{QILC}}L = {}^{\mathrm{K\text{-}ILC}}L$ and solve for $\mathbf{K}_j$:

$$\mathbf{K}_j = \tilde{\mathbf{F}}(\mathbf{W}_{\Delta u} + \tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e\tilde{\mathbf{F}})^{-1}\tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e.$$

□

Interpretation: The intuitive explanation is as follows: the component penalizing $\mathbf{\Delta u}_j$ in the QILC cost function (19) takes a similar role as the filter in the K-ILC algorithm. However, while in the K-ILC algorithm the filter properties can be separately designed via the initial covariance $\mathbf{P}_0$, $\mathbf{H}_j$, and $\mathbf{E}_j$, the QILC scheme combines filtering and input update in one optimization step. In addition, the filtering in the K-ILC case is designed to be *iteration-varying allowing fast adaptation initially and rejecting noise in later stages.* In the QILC case, the 'filtering' via $\mathbf{W}_{\Delta u}$ is kept constant. Note also that this proof assumes no soft or hard input constraints and the squared 2-norm in the cost function.

We can also turn around the argument to determine the QILC weightings depending on $\mathbf{K}_j$:

Proposition 6. *For any K-ILC with gain matrix* $\mathbf{K}_{j,K}$ *and a cost function as in* (23)*, we can find a QILC set of weighting matrices* $\mathbf{W}_e$ *and* $\mathbf{W}_{\Delta u}$ *that results in an equivalent behavior for that specific iteration* j*.*

Proof. Solving (3.4) for $\mathbf{W}_{\Delta u}$ holds

$$\mathbf{W}_{\Delta u} = \tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e(\mathbf{K}_{j,\mathrm{K}}^{-1} - \mathbf{I})\tilde{\mathbf{F}}. \tag{41}$$

One sees that we can find a matching $\mathbf{W}_{\Delta u}$ for any any arbitrary $\mathbf{W}_e$. The choice of $\mathbf{W}_e$ is in fact redundant for QILC (given it is positive definite) and is usually set to $\mathbf{W}_e = \mathbf{I}$. □

Interpretation: One can now find a QILC equivalent for every possible $\mathbf{K}_{j,\mathrm{K}}$, i.e. for every j-th iteration of one learning process. Hence it is possible to compare the initial and converged behavior of K-ILC with an analogous QILC. Such a comparison is found in Sect. 5.

From Prop. 6 we can derive the formula for the common case of diagonal $\mathbf{K}_j = \lambda_j \mathbf{I}, \lambda_j \in \mathbb{R}$: From (41) we derive $\mathbf{W}_{\Delta u} = \frac{1-\lambda_j}{\lambda_j}\tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e\tilde{\mathbf{F}}$. Note that in this case the weighting matrix is non-diagonal - for QILC setup as introduced we restricted the cases on diagonal weightings.

4 Parameter Design Considerations

4.1 K-ILC Convergence Behavior

As discussed in 2.2.3, the Kalman filter equations (12) provide optimal, iteration-varying gains, which change the characteristics of the learning scheme over the course of a learning experiment. The characteristics of this behavior are studied in the following propositions.

Proposition 7. *For a K-ILC algorithm with cost function* (23)*, choosing a large initial disturbance uncertainty* $\mathbf{P}_0$ *approximates a model inversion as in* (24) *for the initial learning iterations.*

Proof. As follows directly from the Kalman filter equations (12), for $\|\mathbf{P}_0 + \mathbf{E}_0\| \gg \|\mathbf{H}_1\|$ the gain $\mathbf{K}_0 = (\mathbf{P}_0 + \mathbf{E}_0)(\mathbf{P}_0 + \mathbf{E}_0 + \mathbf{H}_1)^{-1} \simeq \mathbf{I}$, and from (39) ${}^{\mathrm{K\text{-}ILC}}\mathbf{u}_1 = \mathbf{u}_{\mathrm{nom}} - \tilde{\mathbf{F}}^{-1}\mathbf{e}_0$, which corresponds to the update rule $\mathbf{u}_{j+1} = \mathbf{u}_j - \tilde{\mathbf{F}}^{-1}\mathbf{e}_j$ of a model-inversion QILC (24). □

Interpretation: This insight has great practical value. As ILC is used to correct for an initially unknown disturbance (i.e. unmodelled dynamics or exogenous disturbances) a large $\mathbf{P}_0$ is effective for most applications when aiming for a fast convergence rate initially. The Kalman filter then adapts its estimation gain over the iterations while compromising optimally between newly collected data and the current estimate of the disturbance (based on previously collected experience). This trade-off is made based on the chosen output and learning covariances $\mathbf{H}_j, \mathbf{E}_j$, see (9).

Proposition 8. *Given a K-ILC learning model* (9) *with negligible output noise; i.e.* $\mathbf{H}_j = 0$ *for all* j *and a cost function* (23)*, the optimal algorithm is a model inversion.*

Proof. As explained in Prop. 4, a modelled output noise of zero implies $\mathbf{K}_{j,\mathrm{K}} = \lambda\mathbf{I} = \mathbf{I}\ \forall j$, and thus $\lambda = 1$. As we have showed in Prop. 5, this is equivalent to a specific QILC algorithm with $\mathbf{W}_{\Delta u} = \frac{1-\lambda}{\lambda}\tilde{\mathbf{F}}^{\mathrm{T}}\mathbf{W}_e\tilde{\mathbf{F}} = 0$, which again is the same as the model inversion presented in (24); i.e. $\mathbf{u}_{j+1} = \mathbf{u}_j - \tilde{\mathbf{F}}^{-1}\mathbf{e}_j$. □

Interpretation: A model with negligible output noise implies $\mathbf{K}_j = \mathbf{I}$, as found in Prop. 7. This corresponds to the situation where the initial disturbance uncertainty $\mathbf{P}_0$ is expected to be large as the disturbance is expected to be important (Prop. 7). Thus, for such situations, the optimal ILC algorithm is a model inversion.

Proposition 9. *For any converged K-ILC gain $\mathbf{K}_{K,\infty}$, a corresponding equivalent Q-ILC case can be found.*

Proof. Being a special case of Proposition 6, applying its result to the converged $\mathbf{K}_{K,\infty}$ we obtain the corresponding QILC weighting matrices straightforward. □

Interpretation: The results helps to interpret the effect of the converged Kalman gain as the weighting matrices for the equivalent QILC case. Vice-versa, one could design the Kalman gain such that it adapts a desired QILC behavior after converging: $\mathbf{K}_{K,\infty}$ can be found by solving a Riccati equation that depends on the filter parameters $\mathbf{I}, \mathbf{E}_\infty = \epsilon\mathbf{I}, \mathbf{H}_\infty = \eta\mathbf{I}$. The solution for diagonal covariance matrices is covered in ([28]). Inserted in the QILC weighting matrix for diagonal $\mathbf{K}_{j,\mathrm{K}}$ it holds

$$\mathbf{W}_{\Delta u} = \frac{1-\lambda_\infty}{\lambda_\infty}\tilde{\mathbf{F}}^\mathrm{T}\mathbf{W}_e\tilde{\mathbf{F}} = \frac{2\eta/\epsilon}{1+\sqrt{1+4\eta/\epsilon}}\tilde{\mathbf{F}}^\mathrm{T}\mathbf{W}_e\tilde{\mathbf{F}}.$$

For the converged behavior of the K-ILC algorithm, the behavior can now be chosen according QILC rules by tuning the covariance parameters.

4.2 Influence of Nondiagonal Covariance Matrix

The design of the covariance matrix is an important parameter of the K-ILC setup and theoretically provides as many parameters as there are entries in $\mathbf{P}_0, \mathbf{E}_j$ and $\mathbf{H}_j$. In contrast to a standard time-domain system, the stochastic model maps entire control tasks at once. It is thus in practice difficult to choose a correct stochastic model, hence the filter is usually designed more heuristically with less degrees of freedom (such as diagonal covariance matrices). In the following we demonstrate the effect of choosing covariances as Toeplitz matrix, i.e. all elements equal along the top-left to bottom right diagonals.

Proposition 10. *A Toeplitz matrix filters a lifted vector form signal equivalently to a FIR with a filter window according to the antidiagonal entries of the Toeplitz matrix.*

Proof. For a Toeplitz matrix $M \in \mathbb{R}^{n\times n}, n \in 2\mathbb{Z}$ and a filter window vector $w \in \mathbb{R}^{2n}$ such that

$$\mathbf{M} = \begin{bmatrix} w[n] & \dots\ w[n+l] & \dots\ w[2n-1] \\ \vdots & \ddots \qquad \ddots & \vdots \\ w[n-l] & w[n] & w[n+l] \\ \vdots & \ddots \qquad \ddots & \vdots \\ w[1] & \dots\ w[n-l] & \dots \quad w[n] \end{bmatrix}.$$

A time discrete signal of length n in a lifted vector form as $\mathbf{e}_j$ multiplied with M, $\mathbf{z}_j = \mathbf{M}\mathbf{e}_j$, is equivalent to a convolution with the mirrored filter window $w[n+1-k] = w'[k]$:

$$\mathbf{z}_j[k] = \sum_{i=1}^{n}\mathbf{M}_{k,i}\mathbf{e}_j[i] = \sum_{i=1}^{n} w[n+i-k]\mathbf{e}_j[i]$$
$$\Leftrightarrow \mathbf{z}_j^\mathrm{T} = \mathbf{e}_j^\mathrm{T} * w'$$

□

Interpretation: At different stages in the algorithms, Toeplitz matrices are multiplied with a lifted vector form signal, usually the measured error. Hence, the structure of the antidiagonal matrix entries define the filter effect of the filter gain $\mathbf{K}_{j,\mathrm{K}}$ for instance.

Proposition 11. *Given an output noise modelled as large white noise ($\mathbf{H}_j = h_j\mathbf{I}, h_j \gg 1$), the covariance matrices $\mathbf{P}_0$ and $\mathbf{E}_j$ define the frequency filtering spectrum of $\mathbf{K}_{j,K}$ which filters the measured error by multiplication, as $\mathbf{K}_{j,K}\mathbf{e}_j$ in the explicit K-ILC form* (30)*. For the initial $\mathbf{K}_0$ this property can be showed analytically:*

Proof. The Kalman gain is given by the function

$$\begin{aligned} \mathbf{S}_j &= \mathbf{P}_j + \mathbf{E}_j \\ \mathbf{K}_{j,\mathrm{K}} &= \mathbf{S}_j(\mathbf{S}_j + \mathbf{H}_{j+1})^{-1} \\ &= (\mathbf{P}_j + \mathbf{E}_j)(\mathbf{P}_j + \mathbf{E}_j + \mathbf{H}_{j+1})^{-1}, \end{aligned} \tag{42}$$

For the case that we have a large white output noise, we have that $\|\mathbf{P}_j + \mathbf{E}_j\| \ll \|\mathbf{H}_{j+1}\|$ and hence the inverted term can be simplified to

$$(\mathbf{P}_j + \mathbf{E}_j + \mathbf{H}_{j+1})^{-1} \simeq (\mathbf{H}_{j+1})^{-1} = \frac{1}{h_{j+1}\mathbf{I}}.$$

With that, the matrix structure of the filter matrix $\mathbf{K}_{j,\mathrm{K}}$ is determined by $(\mathbf{P}_j + \mathbf{E}_j)$, scaled with a factor $\frac{1}{h_{j+1}}$. Assuming that all covariances matrices are Toeplitz, the filtering property of the gain can be determined with the structure of one row of the matrix as shown in Proposition 10. □

Interpretation: Through the covariance of the modelled stochastic variables we can include frequency-domain information about the disturbance. This information is reflected in the structure of the Kalman gain $\mathbf{K}_{j,\mathrm{K}}$. As seen in (30), the gain matrix is multiplied with the error signal, acting as a FIR filter on the measurement.

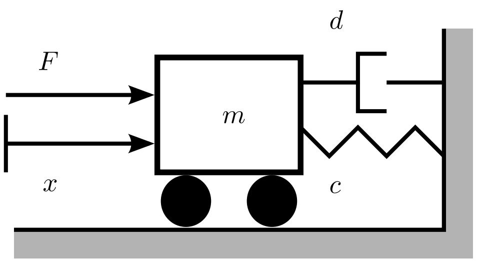

Fig. 3. We use a mass-springer-damper system with the input force F as our simulation example. In contrast to our nominal model $\tilde{\mathbf{F}}$, the simulated system has nonzero damping and spring coefficients.

5 Simulation Example

In the simulation we compare a K-ILC and variants of the QILC algorithm applied on a mass-spring-damper system (see Figure 3). For comparison we include the QILC equivalent of both the initial and converged K-ILC algorithm, highlighting the findings of the previous section.

5.1 Nominal Model and Simulated Mass-Spring-Damper System

For the simulation example we consider a classic 1-D mass-spring-damper system. The equations of motion are:

$$\ddot{x} = \underbrace{F/m}_{\gamma u} - \frac{d}{m}\dot{x} - \frac{c}{m}x. \tag{43}$$

We use two different linear time discrete models: One for the nominal model that is used by the ILC algorithms and a second simulation model to represent a real system to which the learning is applied. The damping and spring effects are neglected in the nominal model in order to include model errors; i.e. $c = d = 0$. For the simulated system we set $c/m = 0.01, d/m = 0.001$. The control of the position of the car happens over the applied force F, which, in the simulated system, will be corrupted by a factor $\gamma = 0.9$ (in the nominal model we have $\gamma = 1$). Discretized with a zero-order-hold and sample time T_s we obtain the following discrete-time system as nominal model:

$$\begin{aligned} x[k+1] &= \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} x[k] + \begin{bmatrix} T_s^2/2 \\ T_s \end{bmatrix} u[k] \\ y[k] &= \begin{bmatrix} 1 & 0 \end{bmatrix} x[k] + \begin{bmatrix} 0 \end{bmatrix} u[k], \end{aligned} \tag{44}$$

where $x[k] \in \mathbb{R}^2$ represents position and velocity at time kT_s. From that time-domain model we construct the iteration-domain $\tilde{\mathbf{F}}$, refer to [14] on how $\tilde{\mathbf{F}}$ is obtained from (44). Note that only 'real' system (43) is asymptotically stable. The nominal dynamics (44) have two eigenvalues $= 1$ which makes it by definition unstable, but diverging slowly, i.e. linearly.

The 'real' system is also discretized with zero-order-hold. As additional disturbances we add both a static disturbance $d_{\text{static}}[k] = 0.2\sin^2(\frac{4\pi k}{N})$ and Gaussian noise ($\mathcal{N}(0,\sigma)$) to the output signal $y[k]$. The desired trajectory is set to $\mathbf{y}_d[k] = \sin^2(k\frac{2\pi}{2N})$ and determines the desired position of the car.

Table 1
Parameters used for simulations in Figure 4 and converged error norm $\|\mathbf{e}_\infty\|$. The converged error norm was approximated with the average error norms of iteration 190 to 200.

	$\mathbf{W}_{\Delta u}$	$\mathbf{W}_e$	$\|\mathbf{e}_\infty\|_{\sigma_I=0.1}$	$\|\mathbf{e}_\infty\|_{\sigma_{II}=0.01}$
K-ILC	0	$\mathbf{I}$	0.1201	0.0115
QILC	$0.3\mathbf{I}$	$\mathbf{I}$	0.1159	0.0114
QILC-c	$\lambda_\infty \tilde{\mathbf{F}}^{\mathrm{T}}\tilde{\mathbf{F}}$	$\mathbf{I}$	0.1197	0.0117
QILC-i	0	$\mathbf{I}$	0.1482	0.0152
K-ILC	$\mathbf{P}_0 = 100\mathbf{I}$, $\mathbf{H}_j = 0.1\mathbf{I}$, $\mathbf{E}_j = 0.001$, $\lambda_\infty = 0.095$			

5.2 Simulation Setup

In the simulation four different algorithms are compared and applied to the simulated 'real' system. Two cases (I and II) are done with a noise variance at $\sigma_I = 0.01$ and $\sigma_{II} = 0.01$. The learning algorithms are each set to run over 30 iterations of $N = 160$ time steps. All algorithms use the same linear, discrete-time model $\tilde{\mathbf{F}}$ based on (44). The first input guess $\mathbf{u}_0 = \mathbf{u}_{\text{nom}}$ is the nominal input given by the model inversion $\tilde{\mathbf{F}}^{-1}\mathbf{y}_d = \mathbf{u}_{\text{nom}}$. The design parameters are chosen as stated in Table 1. To compare the algorithm's behavior the evolution of each learning iteration's error 2-norm ($\|\mathbf{e}_j\|$) is plotted. Also, we average over 120 repetitions of the entire learning process to cancel out the noise.

5.3 Learning Algorithm Setup

In the following we explain the choice of parameters for the different learning algorithms used for the simulation setup. Note that for the comparison we will use the nominal input as initial input guess for all algorithms. We will compare the K-ILC and QILC algorithm and denote as QILC-i and QILC-c the QILC algorithms that are equivalent to the K-ILC setup for chosen iterations. *K-ILC:* The estimation-based learning algorithm is used with optimal Kalman gains based on diagonal covariance matrices for the modelled stochastic variables similar to [1], see Table 1. The output noise covariance matrix $\mathbf{H}_j$ models the actual output noise for $\sigma_I = 0.1$ optimally and is modelled too large for $\sigma_{II} = 0.01$. That choice is nevertheless also realistic from the point of view that for robustness one would rather overestimate than underestimate the modelled noise. Further, the dynamic noise covariance $\mathbf{E}_j$ models process noise which leads to a shifting mean disturbance. However, besides accounting for actual occurring randomness in the system, both

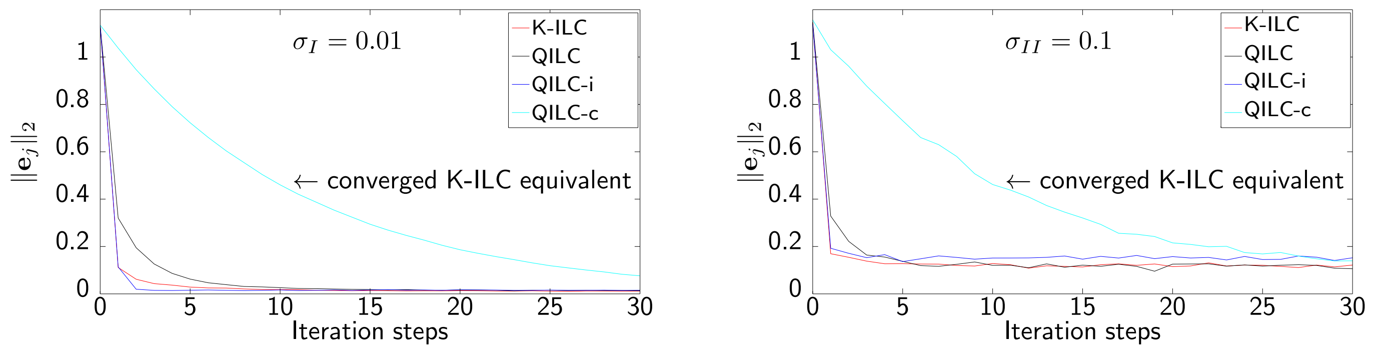

Fig. 4. Average errors of the K-ILC and QILC algorithms as well as initial and converged QILC equivalent of K-ILC with a white output noise with variance $\sigma_I = 0.01$(**Left**) and $\sigma_{II} = 0.1$(**Right**). **Left**: Initially, K-ILC and the equivalent QILC-i perform best. QILC is slower at reaching the converged state. Due to the relatively small noise all algorithms converge to similar error levels. On a closer look one can see that K-ILC performs better than QILC-i after iteration 15 and QILC-c has not converged entirely after 30 iterations. **Right**: The large noise brings the K-ILC, QILC and QILC-c setups to similar converged level, as all approach the "best feasible" performance. The larger noise sensitivity of the model inversion QILC-i setup is clearly visible, as it performs weaker than all others once it is converged. The K-ILC setup converges faster than Q-ILC but performs similar once converged.

random variables also have to take into account the errors of the imperfect nominal system model and the resulting correction from the algorithm. Hence, a part of the stochastic disturbance $\mathbf{d}_j + \eta_j$ in (9) accounts for the unmodelled dynamics and eventual linearization errors. As these modeling errors are usually deterministic and can hence be driven to zero by the algorithm, they can be modelled by the initial disturbance $\mathbf{P}_0$. As the disturbance is not simply a initial uncertainty, the step by step correction of the disturbance by the algorithm has also to be modelled by the process noise covariance $\mathbf{E}_j$, representing a random walk process. As the hereby modelled system has no actual process noise and the influence of modeling errors diminish with convergence, $\mathbf{E}_j$ should vanish for later iterations. As we restricted the modelled covariances to iteration-constant values we chose a small $\mathbf{E}_j = 0.001$. Thus the modeling error have to be mainly covered by the initial uncertainty $\mathbf{P}_0$, which was hence chosen to be large in order to compensate for the initial influence of the unmodelled dynamics. It was also known from experience that a large $\mathbf{P}_0$ brings good initial convergence.

To sum up the design procedure, we can state following for the parameters: That $\mathbf{P}_0$ accounts for the initial, deterministic error of the simulated system and also covers erroneous dynamics, $\mathbf{E}_j$ models process noise of the converged solution and $\mathbf{H}_j$ models the additive white output noise that can also account for unmodelled high frequency, zero-mean dynamics.

QILC: The QILC setup uses parameters as commonly chosen in literature, cf. [5,29]: a unit weighting for the error and a scalar factor for the input change penalization $\mathbf{W}_{\Delta u}$, cf. Table 1. The value of ${}^{\mathrm{QILC}}\mathbf{W}_{\Delta u}$ was found by trial, such that it has both a reasonable converged error (Significant only for the large noise $\sigma_{II} = 0.1$ case) and good initial convergence. Comparably, in literature values are in the magnitude of $0.001 - 0.1$ [5,29]. The relatively large output noise of the 'real' system resulted in a more robust, rather large $\mathbf{W}_{\Delta u}$.

QILC-i: Chosen according to Proposition 8, QILC-i is the QILC equivalent of the initial K-ILC algorithm as used for the simulation. As $\mathbf{P}_0$ is chosen to be large in the K-ILC setup, QILC-i approximates a model inversion resulting in an input update as described in Proposition 7.

QILC-c: The converged K-ILC-equivalent QILC setup uses parameter values according to Prop. 5. The converged λ_∞ can be evaluated numerically or found analytically for diagonal covariance matrices, see [28]. The error of such a QILC setup is adapted to pure noise rejection as the deterministic error is already corrected and is thus converging extremely slow.

5.4 Performance Analysis

In this section the main objective is to compare the performance of the K-ILC versus the QILC setup in the simulations. QILC-i and QILC-c serve as additional algorithms to better show the initial and converged K-ILC behavior. As expected, the error norms do match the K-ILC algorithms in the respective region, as seen in Tab. 1 for the converged state and in Fig. 4 for the initial iteration. We will split the performance analysis in two points, i.e. the initial and converged behavior.

Initial convergence rate: Like for system applicable to iterative learning in general, for our simulation the initial error due to static disturbance and modeling error is more important than the measurement and process noise. This explains the good convergence performance of the QILC-i algorithm: It represents the initial K-ILC which has been showed to be approximately a model inversion for large $\mathbf{P}_0$ (See Prop. 7) as set chosen hereby. As the filter gain $\mathbf{K}_{j,\mathrm{K}}$ converges to $\mathbf{K}_{\infty,\mathrm{K}}$ after the first iteration and with that $\mathbf{P}_j$ to $\mathbf{P}_\infty$, the convergence rate diminishes compared to QILC-i for $\sigma_I = 0.01$. For large output noise however ($\sigma_{II} = 0.1$), the K-ILC algorithm outperforms even the QILC-i setup. This is probably due to the high noise sensitivity of QILC-i and the fact that the error already attains the converged magnitude after one iteration, which prevents a better convergence than attained after the first iteration. As expected, the QILC equivalent of the converged K-ILC algorithm (QILC-c) has a weak initial convergence. The converged K-ILC algorithm is optimized to filter noise, which leads to the trade-off of slow convergence. The convergence rate of the QILC setup is reasonable for both cases although clearly weaker than K-ILC in the first 3 iterations.

Converged performance: The performance of the converged algorithm is closely related to its noise robustness: In the simulation the disturbance is static, except for the white gaussian output noise. The initial static disturbance should therefore be perfectly corrected by the updated input, with only output noise disturbing the trajectory. As expected, the model inverting QILC-i algorithm performs weakest, both with large and small noise. As the converged error is only noise, a model inversion misinterprets the measurement entirely and reacts very sensitive to it. Both K-ILC and QILC algorithms perform better in this case: The K-ILC setup (and its converged equivalent QILC-c) perform about 20% better than QILC-i for the small noise variance σ_{II} and about 14% for the larger noise σ_I.

For larger noise σ_I, QILC has a converged error about the same as KILC. Both having small noise sensitivity, the converged error norm is approximately the error due to the norm itself as the input has converged perfectly. The difference of the approximated $\|\mathbf{e}_\infty\|$ of QILC-c and K-ILC setups has to be attributed to the slow convergence of QILC-c setup and the noise that is not cancelled entirely by averaging over 120 simulations.

Adaptive behavior: The main difference between QILC and K-ILC setups as discussed here is that QILC learning algorithm is iteration-constant whereas K-ILC changes its behavior over the iterations according to the modelled covariances. Adapting the learning setup is an advantage that allows optimization for initial and converged performance. As modelled with the large initial disturbance covariance $\mathbf{P}_0$ we assume that during the learning process, the disturbance diminishes. The improvement of the K-ILC setup comes from the additional information of the system: Whereas the QILC setup relies on the linear model, through the stochastic system model used for the Kalman gain calculation the K-ILC setup benefits from better information about the system. Nevertheless, this can be a disadvantage in special cases: If additional perturbation occurs during the learning process, e.g. in case a static disturbance is added to the output after an unknown number of iterations, K-ILC would perform much worse to correct it than iteration-constant QILC. In other words, the K-ILC loses certain robustness against additional, non gaussian disturbances that are not included in the stochastic model, e.g. additional wind coming up after some learning iterations when learning a precise flight trajectory. However, these could be considered by expanding covariance matrix design to iteration-dependent values when the time of the additional disturbance is known.

5.5 *Overview on Design Guidelines*

The ILC setups discussed in this paper are all time invariant learning systems as the simulation example is also time invariant. With that we mean that the covariance matrices, the weighting matrices as well as the system dynamics $\tilde{\mathbf{F}}$ are all modelled, or designed respectively, to be Toeplitz. Further, only diagonal matrices have been used as weighting matrices in the QILC setups for reasons of analysis. It reduces the number of design parameters to manageable numbers, due to the lacking of design guidelines for other. For the Kalman gain, whose design relies entirely on the stochastic model parameters, diagonal covariance matrices have been chosen as first approach model of the noise covariances (See Tab. 1).

Another simplification was to use iteration constant covariance matrices in the stochastic model (e.g. $\mathbf{E}_j = E$ $\forall j$) as well as iteration constant weighting matrices in the QILC setup. This allows a more straightforward parameter design, but implies losing the possibility of including more system information to the algorithms that are not taken into account with the linear dynamics model and/or stochastic model.

Under these assumptions one can formulate following design guidelines:

5.5.1 *Classic QILC parameter design*

As discussed in Prop. 6, there is only one degree of freedom for diagonal QILC weighting matrices. One can thus set $\mathbf{W}_e = \mathbf{I}$ and only tune $\mathbf{W}_{\Delta u} = w_{\Delta u}\mathbf{I}$. There exist few guidelines in literature for the design of $w_{\Delta u}$, e.g. in context with a robustness limit as given by Bristow in [29]. In [5] the scaling parameter is found by trial and error, where the tuning has been between $w_{\Delta u} = \{0.001, 0.5\}$. As systems are different for every case, it is unavoidable to be evaluated again for every setup, such as for the simulation example QILC setup. With only one degree of freedom from the parameter, the design is essentially a trade-off between fast initial convergence for small $\mathbf{W}_{\Delta u}$ and a noise-robust converged state for larger $\mathbf{W}_{\Delta u}$. After some trials the actual value was found in our example.

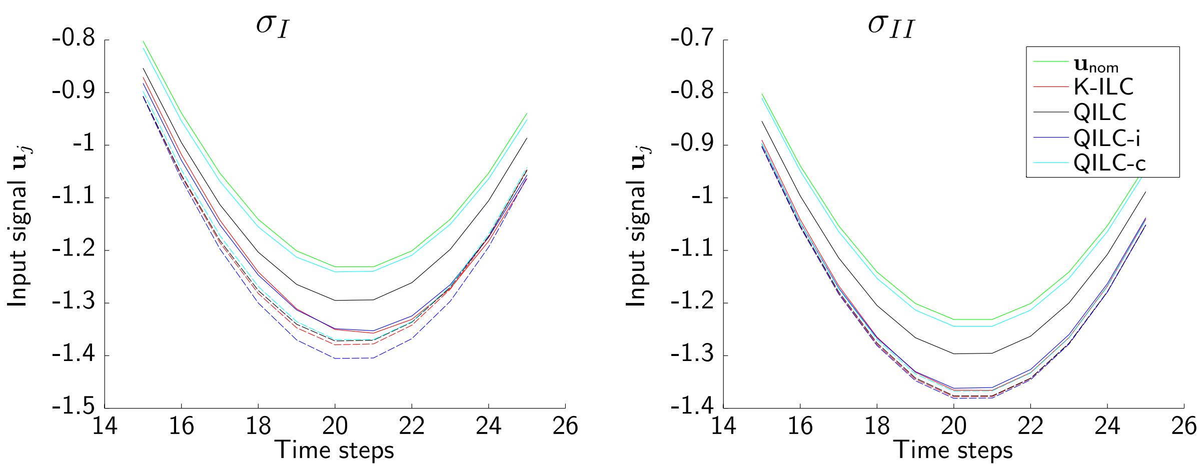

Fig. 5. Input signal trajectory for large and lower noise (left and right). The full line is the first iteration of the algorithms, the dashed belongs to the 30th iterations of the learning algorithms. One sees that the **first** iteration convergence is slowest for the converged K-ILC equivalent (QILC-c), whereas the K-ILC and the similar QILC-i perform best. For the small amplitude noise (right), **after 30 iterations** the algorithms have already converged much more than closer than for larger noise. The results are averaged over 100 repeated runs to cancel noise effects.

5.5.2 Classic K-ILC parameter design

The entire K-ILC setup as used in the simulation, similar to the one as presented in [1,20], is defined over three scalars: $\mathbf{P}_0 = p\mathbf{I}, \mathbf{E}_j = E\mathbf{I}$ and $\mathbf{H}_j = H\mathbf{I}$. Although these are more parameters to set than we need for QILC, through their physical meaning a design guideline is given. p should be in the order of magnitude of at least the average expected disturbance squared, such as to force a quick convergence rate. As the amplitude of the disturbance can be difficult to guess, it is safer to choose the value too large, approximating the first learning step as model inversion (See Prop. 8). The modelled output noise H is self-explanatory; The sensor accuracy is usually well known. The covariance E models two aspects: Firstly, for the part accounting for modeling errors, but should be kept small as it is iteration constant and its influence is small once converged. However, a 'too small' $\mathbf{E}_j$ can be corrected by a even larger $\mathbf{P}_0$. Secondly, the covariance E also accounts for the process noise. The two aspects can be added due to the linearity of gaussian distributions.

5.5.3 Additional ILC setup design considerations

In this paragraph we present additional thoughts on the design of ILC parameters:

- Iteration-varying dynamic noise covariance for K-ILC setup: The covariance $\mathbf{E}_j$ of the dynamic noise models two things at the same time: Firstly, the dynamic process noise of the system such as a time varying, non-zero-mean disturbance. Secondly, it stands for the modeling errors that affect the change of output caused by the input update. As the input converges to the optimal input, the modeling errors do not affect the output anymore and only process and measurement noise affect the output. As the process noise is usually small compared to the modeling errors, one should consider a covariance matrix $\mathbf{E}_j$ that significantly diminishes with increasing iteration count. Ideally, it should vanish as soon as the input update has converged. This can be modelled with iteration-varying covariances that would ideally diminish at the conversion rate of the algorithm. This would however difficult to determine in advance.
- Iteration-varying QILC weighting matrices: The problem of the trade-off between fast initial convergence and good converged performance in QILC setups can also be addressed by introducing iteration-varying parameters: By scaling up $\mathbf{W}_{\Delta u}$ after conversion of the system, the noise robustness could be raised for the converged algorithm. However, the choice if this happens gradually or after how many iterations is another design parameter that would need to be found by trial. Nevertheless it may improve the performance considerably.
- As introduced in Prop. 11, if the frequency spectrum of the expected disturbance is known the covariances $\mathbf{P}_0$ and $\mathbf{E}_j$ can be chosen accordingly. With that, the filter gain $\mathbf{K}_{j,\mathrm{K}}$ can filter noise frequencies that are not significant for the learning.
- A similar property could be designed for QILC: as we have $\mathbf{e}_j^{\mathrm{T}}\mathbf{W}_{\Delta u}\mathbf{e}_j = \mathbf{e}_j^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}_{\Delta u}\mathbf{W}'_{\Delta u}\mathbf{e}_j$ from which we can extract $\mathbf{W}'_{\Delta u}\mathbf{e}_j$ as the weighted error signal, a filtering $\mathbf{W}'_{\Delta u}$ as proposed in Prop. 10 can also filter noise frequencies that are eventually not part of the expected disturbance spectrum.
- Toeplitz constraints: As applied by Schoellig *et al.* in [21] an additional penalization of the input $\mathbf{u}_j$ with a weighting $\mathbf{W}_u = \mathbf{W}_u^{\mathrm{T}}\mathbf{W}'_u$ in the cost function also be used for reducing the noise sensitivity of the converged

input. By choosing $\mathbf{W}_u'$ as a second derivative operator one forces a smooth input update. With that, one can steer the converged input in a frequency spectrum that reduces instabilities and noise.
- Presented by Bristow in [29], one can also shape the weighting matrix along the iteration time steps (making it non-Toeplitz) to highlight specific time windows of one iteration. This can especially improve the convergence if a disturbance is known to happen at certain times of one iteration.

We shall note at this place that it can be difficult to provide guaranteed information about the robustness of the ILC algorithm. Even the convergence may, if the model is too inaccurate, be in question. Ultimately, the performance of the system depends on the quality of the information given about the system, which can be improved by the stochastic model based estimator with the K-ILC setup in contrast to the QILC approach.

6 Conclusion

The comparison with E-QILC has shown it to be closely related to K-ILC, having a different cost function and a simple filter gain, avoiding a stochastic model. Similarly, the nonlinear formulation can be seen as a generalized form of K-ILC, with out any filtering (i.e. $\mathbf{K}_{j,\mathrm{K}} = \mathbf{I}$). With a comparison of QILC and K-ILC on an analytical level we could demonstrate certain fundamental parallels. In fact, for every QILC setting an equivalent K-ILC setting can be found with $\mathbf{W}_{\Delta u} = 0$, equivalently vice-versa. Further, if a system is modelled without output noise, the approaches are indistinguishable for arbitrary, equal weighting matrices. While especially the QILC setting has a lot of design parameters, the degrees of freedom are usually artificially restricted for a practicable tuning process.

Without the parameter of $\mathbf{W}_{\Delta u}$, the stochastic model used for the estimator in the K-ILC algorithm is the single design tool. It provides an accessible guideline for parameter choice. With that, the design task moves from abstract weighting matrices to a stochastic model of the system - which helps to understand the meaning of the chosen parameters and thus also to choose them correctly. In contrast to standard parameter design rules for QILC settings, both initial and converged behavior can be tuned independently with the K-ILC design parameters of noise covariance. The better performance of the K-ILC setup can ultimately be explained through the the additional information about the system that is provided with a stochastic model compared to the linearized dynamics only.

In the simulation example, we compared the initial and converged K-ILC behavior to the corresponding equivalent QILC designs, and saw that an optimal Kalman gain results in an iteration-varying learning action - even for a basic stochastic model with iteration-constant diagonal covariance matrix choice. The resulting learning performance is improved, as it changes from fast convergence and high sensitivity (model inversion) for the initial iterations to less sensitive converged $\mathbf{K}_{\infty,\mathrm{K}}$. As seen in the simulation example, an estimation gain changes over the course of a learning experiment allows optimization of the scheme for initial as well as converged behavior. With that, it also improves the general robustness towards uncertainties, as it outperforms the QILC setting for small noise σ_I and larger noise σ_{II} (see Fig. 4).

It is thus through the clear separation of estimation and noise filtering from the input update step that the K-ILC takes a better performance compared to a QILC algorithm tuned as presented in literature. Through the use of an iteration-varying Kalman filter we can furthermore achieve both fast initial convergence rate (As for QILC-i) while still achieving a robust converge state (As for QILC-c).

References

[1] A. P. Schoellig, R. D'Andrea, Optimization-based iterative learning control for trajectory tracking, in: Proceedings of the European Control Conference, 2009, pp. 1505–1510.

[2] N. Degen, A. P. Schoellig, Design of norm-optimal iterative learning controllers: The effect of an iteration-domain kalman filter for disturbance estimation, Proceedings of the 53rd IEEE Conference on Decision and Control 1.

[3] K. L. Moore, Iterative Learning Control for Deterministic Systems, Springer-Verlag, London, 1993.

[4] D. A. Bristow, M. Tharayil, A. G. Alleyne, A survey of iterative learning control, IEEE Control Systems Magazine (2006) 96–114.

[5] J. H. Lee, K. S. Lee, W. C. Kim, Model-based iterative learning control with a quadratic criterion for time-varying linear systems, Automatica 36 (2000) 641–657.

[6] M. Volckaert, M. Diehl, J. Swevers, Generalization of norm optimal ilc for nonlinear systems with constraints, Mechanical Systems and Signal Processing 39 (1-2) (2013) 280–296.

[7] S. Arimoto, S. Kawamura, F. Miyazaki, Bettering operations of dynamic systems by learning: A new control theory for servomechanism or mechatronic systems, in: Proceedings of Conference on Decision and Control, 1984, pp. 1054–1069.

[8] N. Amann, D. H. Owens, E. Rogers, Iterative learning control using optimal feedback and feedforward actions, International Journal of Control 65 (2) (1996) 277–293.

[9] S. Gunnarsson, M. Norrlöf, On the design of ilc algorithms using optimization, Automatica 37 (2001) 2011–2016.

[10] B. G. Dijkstra, O. Bosgra, Convergence design considerations of low order q-ilc for closed loop systems , implemented on a high precision wafer stage (December) (2002) 2494–2499.

[11] K. S. Lee, J. H. Lee, Model predictive control for nonlinear batch processes with asymptotically perfect tracking, Computers Chemical Engineering 21.

[12] B. Chu, D. H. Owens, Iterative learning control for constrained linear systems, International Journal of Control 83 (7) (2010) 1397–1413.

[13] D. A. Bristow, A. G. Alleyne, A high precision motion control system with application to microscale robotic deposition, Transacions on Control Systems Technology 14 (6) (2006) 1008–1020.

[14] N. Amann, D. H. Owens, E. Rogers, Iteratice learning control for discrete-time systems with exponential rate of convergence, Proceedings-Control Theory and Applications 143 (2) (1996) 217–224.

[15] R. W. Longman, K. A. Alnajjar, X. Ji, Comments on how a new engineering field develops : A case study from iterative learning and repetitive control, in: Intelligent Technologies and Engineering Systems (ICITES2013), 2nd International Conference on, pp. 1273–1279.

[16] T. Son, G. Pipeleers, J. Swevers, Robust optimal iterative learning control with model uncertainty, in: Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on, 2013, pp. 7522–7527. doi:10.1109/CDC.2013.6761084.

[17] D. Owens, J. Hätönen, Iterative learning control - an optimization paradigm, Annual Reviews in Control 29 (1) (2005) 57–70.

[18] Y. Chen, K. L. Moore, J. Yu, T. Zhang, Iterative learning control and repetitive control in hard disk drive industry a tutorial.

[19] M. Cho, Y. Lee, S. Joo, K. Lee, Semi-empirical model-based multivariable iterative learning control of an rtp system, IEEE Transactions on Semiconductor Manufacturing 18.

[20] F. L. Mueller, A. P. Schoellig, R. D'Andrea, Iterative learning of feed-forward corrections for high-performance tracking, in: International Conference on Intelligent Robots and Systems, 2012, pp. 3276–3281.

[21] A. P. Schoellig, F. L. Mueller, R. D'Andrea, Optimization-based iterative learning for precise quadrocopter trajectory tracking, Autonomous Robots 33 (1-2) (2012) 103–127.

[22] A. Schollig, J. Alonso-Mora, R. D'Andrea, Independent vs. joint estimation in multi-agent iterative learning control, in: Decision and Control (CDC), 49th IEEE Conference on, IEEE, 2010, pp. 6949–6954.

[23] B. Panomruttanarug, R. W. Longman, Using Kalman filter to attenuate noise in learning and repetitive control can easily degrade performance, in: SICE Annual Conference, Ieee, 2008, pp. 3453–3458.

[24] H.-S. Ahn, Y. Chen, K. L. Moore, Iterative learning control: Brief survey and categorization, IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 37 (6) (2007) 1099–1121.

[25] D. Shen, Y. Wang, Survey on stochastic iterative learning control, Journal of Process Control.

[26] C. K. Chui, G. Chen, Kalman Filtering with Real-Time Applications, Vol. 6, Springer-Verlag, London, 1987.

[27] M. Norrlöf, An adaptive iterative learning control algorithm with experiments on an industrial robot, IEEE Transactions on Robotics and Automation 18 (2) (2002) 141–146.

[28] N. Degen, A. P. Schoellig, Additional derivations related to the analysis of the kalman-filter-based ilc scheme.
URL http://www.tiny.cc/DerivationsKalmanILC

[29] D. A. Bristow, Weighting matrix design for robust monotonic convergence in norm optimal iterative learning control, American Control Conference (2008) 4554–4560.