Citation	Volckaert, M., Diehl, M., Swevers, J. (2013) Generalization of norm optimal ILC for nonlinear systems with constraints Mechanical Systems and Signal Processing, 39 (1-2), 280-296
Archived version	Author manuscript: the content is identical to the content of the published paper, but without the final typesetting by the publisher
Published version	http://dx.doi.org/10.1016/j.ymssp.2013.03.009
Publisher homepage	http://www.sciencedirect.com/science/journal/08883270
Author contact	E-mail: jan.swevers@kuleuven.be Phone number: +32 16 32 25 40
IR	https://lirias.kuleuven.be/handle/123456789/402449

(article begins on next page)

A general iterative learning control approach for nonlinear systems

Marnix Volckaert, Moritz Diehl, Jan Swevers

Abstract—This paper discusses a generalized approach to iterative learning control (ILC) for nonlinear systems. The approach is general in the sense that it can use a broad class of nonlinear models, including black box models; it can take input- and other constraints into account; and it is applicable to both tracking control and point to point motions. The algorithm introduces a nonparametric correction to the nominal model of the system, in the form of an extra input signal. An optimal value for the model correction is estimated after each trial. The solution is then used during a second step to calculate an optimal input signal for the next trial. Both estimation and control steps are NLP problems with a very similar, sparse block structure. The efficient solution of this class of optimization problems, using a sparse implementation of an interior point method, is also discussed. It is shown that a well known linear ILC approach is a special case of the presented

approach under certain conditions. The developed approach is validated in experiments and simulation on a nonlinear system, both in a tracking control and a point to point application.

I. INTRODUCTION

Mechatronic systems, such as production machines or industrial robots, often perform the same task repeatedly. In many applications the task is represented by a reference signal y_r that needs to be tracked by the system's output y . Traditional controllers provide the same performance each time the motion is repeated, even if that performance is suboptimal, for example due to model plant mismatch or repeating disturbances. Moreover, many factors can reduce the performance over time, such as slowly changing operating conditions or dynamics. Typically these problems lead to costly recalibration procedures.

Iterative learning control (ILC) is an open loop control strategy that aims to improve the tracking control of repeating motions by using

This work benefits from FWO-project G.0422.08, K.U.Leuven-BOF PFV/10/002 Center-of-Excellence Optimization in Engineering (OPTEC), IWT-SBO-project 80032 (LeCoPro) and the Belgian Programme on Interuniversity Attraction Poles, initiated by the Belgian Federal Science Policy Office (DYSCO).

Marnix Volckaert is with the Department of Mechanical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium. Address all correspondence to: marnix.volckaert@mech.kuleuven.be.

information about the tracking performance of the previous motion, also called trial or iteration. Traditional ILC methods formulate an update equation to calculate the input signal for the next trial as a function of the tracking error of the current trial. Widely regarded as the first international publication on ILC is [1], which proposes an update law of the form $\mathbf{u}_{i+1} = \mathbf{u}_i + \Gamma \mathbf{e}_i$, with $\mathbf{u}_i$ the input of trial i and $\mathbf{e}_i$ the tracking error, defined as $\mathbf{e}_i = \mathbf{y}_r - \mathbf{y}_i$, and Γ a scalar. This update law is called P-type ILC, because the update to the input signal is proportional to the error. This concept is easily extended to more complex update laws with dynamic, model based operators, typically of the following form [2]:

$$\mathbf{u}_{i+1} = Q[\mathbf{u}_i + L(\mathbf{e}_i)], \quad (1)$$

with $Q(\cdot)$ and $L(\cdot)$ a robustness and learning operator respectively. The learning behavior can be manipulated by shaping these (linear) operators.

A common notation in ILC research is to consider the signals $\mathbf{u}$, $\mathbf{y}$ and $\mathbf{e}$ as vectors, and to write linear systems as Markov matrices. This notation is called the lifted systems notation. For example, consider a discrete-time LTI system with relative degree 1, characterized by the finite impulse response $[p_1, p_2, \dots, p_N]$. Assuming an initial state of zero, the input-

output relation of this system can be written as:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \underbrace{\begin{bmatrix} p_1 & 0 & \cdots & 0 \\ p_2 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_N & p_{N-1} & \cdots & p_1 \end{bmatrix}}_{\mathbf{P}} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix}, \quad (2)$$

with $p_1 \neq 0$ because the output is shifted with the relative degree of the system. The operators $Q(\cdot)$ and $L(\cdot)$ can also be written in matrix form as $\mathbf{Q}$ and $\mathbf{L}$. These are not necessarily lower triangular matrices, since $Q(\cdot)$ and $L(\cdot)$ can be noncausal systems. However, for LTI systems, $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{L}$ are Toeplitz. (1) has been extended to LTV systems as well [3], [4], [5].

The lifted systems notation makes it possible to write the learning algorithm as a large MIMO system in the trial domain, while the time domain dynamics are captured in the matrix $\mathbf{P}$. Substituting $\mathbf{e}_i = \mathbf{y}_r - \mathbf{P}\mathbf{u}_i$ in equation (1) and rearranging leads to the trial domain dynamic equation:

$$\mathbf{u}_{i+1} = \mathbf{Q}(\mathbf{I} - \mathbf{L}\mathbf{P})\mathbf{u}_i + \mathbf{Q}\mathbf{L}\mathbf{y}_r \quad (3)$$

from which the well known necessary and sufficient condition for asymptotic stability of the ILC algorithm is derived:

$$\rho[\mathbf{Q}(\mathbf{I} - \mathbf{L}\mathbf{P})] < 1. \quad (4)$$

This condition is often used to shape the matrices $\mathbf{Q}$ and $\mathbf{L}$, to guarantee stability over a broad frequency range. For this reason $\mathbf{Q}$ is often a zero phase low pass filter, with a magnitude of 1 at low frequencies.

A great deal of research is focused on using optimization in the design of the ILC controller. The main idea is to design an update law that minimizes the expected tracking error of the next trial, i.e. to minimize a cost function of the form [2]:

$$J_{i+1}(\mathbf{u}_{i+1}) = \mathbf{e}_{i+1}^T \mathbf{Q}_j \mathbf{e}_{i+1} + \mathbf{u}_{i+1}^T \mathbf{R}_j \mathbf{u}_{i+1} + \Delta \mathbf{u}_{i+1}^T \mathbf{S}_j \Delta \mathbf{u}_{i+1}, \quad (5)$$

where $\Delta \mathbf{u}_{i+1} = (\mathbf{u}_{i+1} - \mathbf{u}_i)$, $\mathbf{Q}_j$ is an $N \times N$ positive-definite matrix, and $\mathbf{R}_j$ and $\mathbf{S}_j$ are $N \times N$ positive-semidefinite matrices. The term $\mathbf{u}_{i+1}^T \mathbf{R}_j \mathbf{u}_{i+1}$ is used to shape the input signal, for example to penalize large control effort, while the term $\Delta \mathbf{u}_{i+1}^T \mathbf{S}_j \Delta \mathbf{u}_{i+1}$ can be used to improve the convergence of the algorithm. Note that (5) depends on the unknown next trial tracking error $\mathbf{e}_{i+1}$. Therefore the cost function needs to be approximated, by using a model $\hat{P}(\mathbf{u})$ for the system $P(\mathbf{u})$, or in lifted form $\hat{\mathbf{P}} \approx \mathbf{P}$.

A common optimization based ILC approach is found by using the update law (1) and designing the matrices $\mathbf{Q}$ and $\mathbf{L}$ such that (5) is minimized. This approach is called quadratically optimal ILC [2]. If the change in $\mathbf{u}$

from iteration i to $i + 1$ is denoted by δ_i , then $\mathbf{u}_{i+1} = \mathbf{u}_i + \delta_i$, and $\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{P}\delta_i$. Therefore the true value $\mathbf{e}_{i+1}$ can be written as $\mathbf{e}_{i+1} = \mathbf{y}_r - \mathbf{y}_i - \mathbf{P}\delta_i$. Using $\hat{\mathbf{P}} \approx \mathbf{P}$ the error can be approximated by $\hat{\mathbf{e}}_{i+1} = \mathbf{y}_r - \mathbf{y}_i - \hat{\mathbf{P}}\delta_i$. Applying the stationary condition for optimality of (5) leads to

$$\begin{aligned} \frac{\partial J_{i+1}}{\partial \mathbf{u}_{i+1}} &= -\hat{\mathbf{P}}^T \mathbf{Q}_e (\mathbf{y}_r - \mathbf{y}_i + \hat{\mathbf{P}} \mathbf{u}_i - \hat{\mathbf{P}} \mathbf{u}_{i+1}) \\ &\quad + \mathbf{R}_u \mathbf{u}_{i+1} + \mathbf{S}_j \Delta \mathbf{u}_{i+1} = 0 \end{aligned} \quad (6)$$

which after rearranging leads to the conventional ILC update law (1) with the following operators:

$$\begin{aligned} \mathbf{Q} &= (\hat{\mathbf{P}}^T \mathbf{Q}_j \hat{\mathbf{P}} + \mathbf{R}_j + \mathbf{S}_j)^{-1} (\hat{\mathbf{P}}^T \mathbf{Q}_e \hat{\mathbf{P}} + \mathbf{S}_j) \\ \mathbf{L} &= (\hat{\mathbf{P}}^T \mathbf{Q}_j \hat{\mathbf{P}} + \mathbf{S}_j)^{-1} \hat{\mathbf{P}}^T \mathbf{Q}_j. \end{aligned} \quad (7)$$

Note that if $\mathbf{Q}_j = \mathbf{I}^{N \times N}$ and $\mathbf{R}_u = \mathbf{S}_j = 0$, the optimal operators are found to be $\mathbf{Q} = \mathbf{I}^{N \times N}$ and $\mathbf{L} = \hat{\mathbf{P}}^{-1}$, which is the simplest form of model based linear ILC. A drawback of the quadratically optimal approach is that it cannot take constraints into account, for example to avoid actuator saturation.

Other ILC approaches have been developed based on the same cost function (5). For example, the norm-optimal approach described in [6] uses an indirect approach, with $\mathbf{R}_j = 0$

and $S_j \neq 0$, using the solution of the Riccati equation. A faster implementation of this approach is described in [7], and an extension to the algorithm, including future trials in the cost function, is described in [8]. Drawbacks of the norm-optimal approach are that it requires full state knowledge during the trial, and that it cannot take constraints into account.

A promising research direction is to solve the minimization of (5) by a direct dynamic optimization approach. This requires a good approximation of $\mathbf{e}_{i+1}$ by $\hat{\mathbf{e}}_{i+1}$. The authors of [9] propose a disturbance model structure for the trial domain dynamics in the following form:

$$\begin{aligned}\mathbf{x}_i &= \mathbf{F}\mathbf{u}_i + \mathbf{d}_i + N_\xi \boldsymbol{\xi}_i \\ \mathbf{y}_i &= \mathbf{G}\mathbf{x}_i + \mathbf{H}\mathbf{u}_i + N_v \mathbf{v}_i\end{aligned}\tag{8}$$

with $\mathbf{x}_i$, $\mathbf{u}_i$ and $\mathbf{y}_i$ the lifted system state, input and output vectors respectively, and $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{H}$ lower triangular matrices representing the system dynamics, such that $\mathbf{P} = \mathbf{G}\mathbf{F} + \mathbf{H}$. The vectors $\boldsymbol{\xi}_i$ and $\mathbf{v}_i$ represent process and measurement noise respectively. The vector $\mathbf{d}_i$ is added to represent the model errors and any repeating disturbances. $\mathbf{d}_i$ can be seen as a nonparametric correction to the nominal model, so a good estimate of $\mathbf{d}_i$ leads to a good estimate of $\hat{\mathbf{e}}_{i+1}$. The algorithm described in [9] estimates the vector $\mathbf{d}_i$ using a Kalman filter in the trial domain, using the measured output $\mathbf{y}_i$ after

each trial. This estimate is then used to solve a convex optimal control problem based on (8) to find the next trial's input signal. However, In order to apply this approach to nonlinear systems, the model should be linearized around the reference signal, which can be difficult if the nonlinear model is for example a black box model. Furthermore, this approach cannot be used for point to point motions, where an exact reference is unknown.

This paper introduces a general approach to iterative learning control for nonlinear systems with constraints, based on an explicit and nonparametric model correction. It uses only output measurements of the previous trial(s), and directly solves a nonlinear optimal control problem, both for the estimation of the model correction and for the calculation of the input signal, since it will be shown that both problems have a very similar structure. The efficient solution of such problems for very long data records, using a sparse implementation of an interior point method, is another contribution of this paper.

Section II will discuss the efficient solution of nonlinear optimal control problems. This solution method will be used in section III to create a generalized ILC algorithm, after introducing the explicit model correction. A comparison with a common linear ILC approach is also discussed here. Section IV describes

the results of the experimental and numerical validation of the algorithm, and conclusions are drawn in section V.

II. EFFICIENT SOLUTION OF NONLINEAR OPTIMAL CONTROL PROBLEMS

This section introduces a general formulation for nonlinear optimal control problems as nonlinear programming (NLP) problems, and then discusses an efficient solution method using an interior point algorithm. This approach allows the exploitation of the sparse structure of the NLP problem. It will be shown in the next section that this solution method can be used to create a versatile ILC algorithm for nonlinear systems.

A. General problem formulation

The general optimal control problem consists of finding control inputs $\mathbf{u}_c$, given a model of the system, exogenous inputs $\mathbf{w}$, and an initial state $\mathbf{x}_{\text{init}}$, such that a norm of a set of signals $\mathbf{e}$ is minimized. This problem is illustrated in figure 1. The functions f and h are the state and output equations respectively. $\mathbf{x} \in \mathbb{R}^{n \times N}$ contains the differential states, with n the order of the model and N the length of the signal, while $\mathbf{z} \in \mathbb{R}^{s \times N}$ contains algebraic states. These are defined by a function c , and are added to the problem in order to take multiple objectives or constraints into account.

Fig. 1. General formulation of a nonlinear optimal problem

$\mathbf{w} \in \mathbb{R}^{q \times N}$ contains the exogenous inputs that are assumed to be known. $\mathbf{u}_c \in \mathbb{R}^{m \times N}$ contains the control inputs that need to be calculated, and $\mathbf{e} \in \mathbb{R}^{p \times N}$ contains the signals to be minimized.

In general $\mathbf{x}$, $\mathbf{u}_c$, $\mathbf{w}$, $\mathbf{z}$ and $\mathbf{e}$ are two dimensional matrices for MIMO systems, with one dimension being time instants k and the other one the number of individual signals. However, in the construction of the optimization problem, each term is stacked in a one dimensional supervector, noted as $\bar{\mathbf{x}}$, $\bar{\mathbf{u}}_c$, $\bar{\mathbf{w}}$, $\bar{\mathbf{z}}$ and $\bar{\mathbf{e}}$, on a sample by sample basis, such that:

$$\bar{\mathbf{e}} = [e_0(0), e_1(0), \dots, e_{p-2}(N-1), e_{p-1}(N-1)]^T. \quad (9)$$

It is assumed that $\bar{\mathbf{e}}$ is a function of $\bar{\mathbf{x}}$, $\bar{\mathbf{u}}_c$ and $\bar{\mathbf{w}}$ only, and is independent of the algebraic states $\bar{\mathbf{z}}$.

The considered application minimizes the 2-norm of $\bar{\mathbf{e}}$, such that the objective function

for the general optimal control problem can be written as:

$$J = \bar{\mathbf{e}}^T \mathbf{Q} \bar{\mathbf{e}} + \bar{\mathbf{u}}_c^T \mathbf{R} \bar{\mathbf{u}}_c + \bar{\mathbf{z}} \mathbf{S} \bar{\mathbf{z}}. \quad (10)$$

The matrix $\mathbf{Q} \in \mathbb{R}^{pN \times pN}$ is positive definite, while $\mathbf{R} \in \mathbb{R}^{mN \times mN}$ and $\mathbf{S} \in \mathbb{R}^{sN \times sN}$ are positive semidefinite. These matrices determine the weights of the multiple objectives in J , and are assumed to be diagonal and stacked in the same order as equation (9).

An efficient and robust formulation of the optimization problem can be achieved by adding $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$ to the optimization vector, containing $\bar{\mathbf{u}}_c$. This vector $\mathbf{v}$ is therefore of length $(n+m+s)N$ and is stacked as follows:

$$\mathbf{v} = [x_0(0), x_1(0), \dots, u_0(0), u_1(0), \dots, z_0(0), z_1(0), \dots, x_0(1), \dots] \quad (11)$$

This definition of the optimization vector leads to the following equality constraints:

$$g(\mathbf{v}) = \begin{cases} \mathbf{x}(0) - \mathbf{x}_{\text{init}} \\ [\mathbf{z}(k) - c[\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)]]_k^{0 \rightarrow N-1} \\ [\mathbf{x}(k+1) - f[\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)]]_k^{0 \rightarrow N-1} \end{cases} \quad (12)$$

for which it must hold that $g(\mathbf{v}) = 0$, with $g(\mathbf{v}) \in \mathbb{R}^{(n+s)N \times 1}$. The derivative of $g(\mathbf{v})$ with respect to $\mathbf{v}$, the Jacobian matrix $\mathbf{J}_g$, plays a crucial role in the minimization of (10) subject to (12). Since $\mathbf{v}$ and $g(\mathbf{v})$ contain $(n+m+s)$

and $(n+s)$ elements per time instant k , and since the constraints are evaluated at each time instant separately, the matrix $\mathbf{J}_g$ has dimension $(n+s)N$ by $(n+m+s)N$, but it has a sparse block structure and contains one block for each $k = 0, \dots, N-1$. As an example, consider the case $n = m = s = 1$, such that

$$\mathbf{J}_g = \left[\begin{array}{ccc|cc} 1 & 0 & 0 & 0 & \dots \\ -\frac{\partial c}{\partial x(0)} & -\frac{\partial c}{\partial u(0)} & 1 & 0 & \dots \\ -\frac{\partial f}{\partial x(0)} & -\frac{\partial f}{\partial u(0)} & 0 & 1 & \dots \\ \hline 0 & 0 & 0 & -\frac{\partial c}{\partial x(1)} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array} \right]. \quad (13)$$

This means that $\mathbf{J}_g$ can be efficiently stored, and that it is easy to construct it by calculating partial derivatives of f and c , avoiding the need to calculate $\mathbf{J}_g$ completely.

Another important matrix that is used to solve the optimization problem is the Hessian H of the Lagrangian function $\mathcal{L}$, defined as $\mathcal{L} = J(\mathbf{v}) - \boldsymbol{\lambda}_g^T g(\mathbf{v}) - \boldsymbol{\mu}_h^T h(\mathbf{v})$. Since (10) is the objective function of a nonlinear least squares problem, the Hessian can be approximated by $\mathbf{J}_j^T \mathbf{J}_j$, with $\mathbf{J}_j$ the Jacobian matrix of the residual function. The matrix H has a similar sparse block structure as $\mathbf{J}_g$, and can be easily constructed using partial derivatives of h . The main advantage of the

favorable structure of J_g and H is that the problem scales very well for longer signals, since the number of nonzeros of both matrices increases linearly with N . To exploit this advantage, the optimization algorithm should process these matrices in a sparse, efficient manner. The application of such an algorithm is discussed in the next section.

The optimization vector $\mathbf{v}$ is bounded by $\mathbf{v}_l$ and $\mathbf{v}_u$ as lower and upper bounds respectively, leading to the following inequality constraints:

$$h(\mathbf{v}) = \begin{cases} \mathbf{v} - \mathbf{v}_l \\ \mathbf{v}_u - \mathbf{v} \end{cases} \quad (14)$$

for which it must hold that $h(\mathbf{v}) \geq 0$. By combining equations (10), (12) and (14), the general formulation can be summarized as:

$$\underset{\mathbf{v}}{\text{minimize}} J(\mathbf{v}) \quad (15a)$$

$$\text{subject to } g(\mathbf{v}) = 0 \quad (15b)$$

$$h(\mathbf{v}) \geq 0. \quad (15c)$$

This is an NLP problem, since (15a) and (15b) are nonlinear functions of the variable vector $\mathbf{v}$. Note that the bounds on $\mathbf{v}$ are the only inequality constraints. For applications with inequality constraints on (linear or nonlinear) functions of $\bar{\mathbf{x}}, \bar{\mathbf{u}}$ or $\bar{\mathbf{w}}$, any number of algebraic states can be added to $\mathbf{v}$, in combination with the equality constraint

$\mathbf{z}(k) - c[\mathbf{x}(k), \mathbf{u}(k), \mathbf{w}(k)] = 0$ for every k , and then a bound can be put on $\bar{\mathbf{z}}$. In this way all inequality constraints become bounds on $\mathbf{v}$. This is advantageous in the application of a particular interior point method to solve (15).

B. Implementation using IPOPT

A necessary condition for optimality of an optimizer $\mathbf{v}^*$ of (15) is the following Karush-Kuhn-Tucker equation [10]:

$$\nabla J(\mathbf{v}^*) - \mathbf{J}_g^T(\mathbf{v}^*)\boldsymbol{\lambda}_g - \mathbf{J}_h^T(\mathbf{v}^*)\boldsymbol{\mu}_h = 0, \quad (16)$$

with $\nabla J(\mathbf{v})$ the gradient of $J(\mathbf{v})$, $\mathbf{J}_g$ and $\mathbf{J}_h$ the Jacobian matrices of $g(\mathbf{v})$ and $h(\mathbf{v})$ respectively, and $\boldsymbol{\lambda}_g, \boldsymbol{\mu}_h$ the corresponding Lagrange multipliers. In order to treat the inequality constraints $h(\mathbf{v}) \geq 0$, the following conditions also have to be satisfied:

$$h(\mathbf{v}) \geq 0$$

$$\boldsymbol{\mu}_h \geq 0 \quad (17)$$

$$h(\mathbf{v})\boldsymbol{\mu}_h = 0$$

These are non-smooth conditions, preventing the use of Newton's method. The idea of an interior point method is to replace the conditions (17) by the approximation $h(\mathbf{v})\boldsymbol{\mu}_h = \tau$, with $\tau > 0$ small. This results in smooth KKT conditions, such that Newton's method can be applied to yield estimates $\mathbf{v}(\tau), \boldsymbol{\lambda}(\tau)$ and $\boldsymbol{\mu}_h(\tau)$. These estimates can then be used as starting values for a problem defined by a

lower value of τ . It can be shown that if $\tau \rightarrow 0$ then $\mathbf{v}(\tau) \rightarrow \mathbf{v}^*$.

A particular implementation of an interior point method for large scale sparse problems is the open source package IPOPT, written in C++ and available under the Eclipse Public License [11]. The user of the algorithm must create a number of functions that define the optimization problem. The IP method itself is provided by IPOPT and is called to solve the user defined problem in the main function. The user defined functions include:

- *get_nlp_info*, *get_bounds_info*: these functions define the number of non zeros in J_g and H , for efficient memory allocation, and define the bounds on $\mathbf{v}$. Bounds do not have to be declared separately as inequality constraints.
- *eval_f*: evaluation of the objective function, J .
- *eval_grad_f*: evaluation of the gradient, ∇J .
- *eval_g*: the equality constraint function $g(\mathbf{v})$.
- *eval_jac_g*: the Jacobian J_g . Only the non zero elements have to be provided, with their row and column index.
- *eval_h*: the Hessian H . Since this is a symmetric matrix, only the lower left half has to be provided, again in sparse form.

III. GENERALIZED NONLINEAR ILC

This section presents a generalized ILC approach for nonlinear systems. The developed approach is model based, however the only assumption is that the model $\hat{P}$ is a discrete time state space model of the following form:

$$\hat{P} = \begin{cases} \mathbf{x}(k+1) = f[\mathbf{x}(k), u(k)] \\ \hat{y}(k) = h[\mathbf{x}(k), u(k)], \end{cases} \quad (18)$$

with $\mathbf{x}$ the state vector of size n , $\mathbf{u}$ the input and $\hat{y}$ the model output. Note that the algorithm is presented for SISO systems, but it is equally applicable to MIMO systems.

The aim of the ILC algorithm is to make the output of the system P track the reference signal $\mathbf{y}_r$, which can be represented by a vector of size N . The algorithm can be summarized in the following steps:

- 1) Assume an initial guess of the input signal, $\mathbf{u}_1$. This initial guess can be for example the solution of an optimal control problem using the nominal model $\hat{P}$.
- 2) Apply the input signal to the system, and measure and record the output, $\mathbf{y}_i$, with i the trial index.
- 3) Use $\mathbf{u}_i$ and $\mathbf{y}_i$ to solve a large parameter estimation problem of size N , to improve $\hat{P}$.
- 4) Calculate a new optimal input signal, based on the improved model.

- 5) Advance the trial index i and return to step 2.

First the model correction will be discussed in detail. It will be shown that the quadratically optimal ILC algorithm for linear systems implicitly makes such a correction, and that the general ILC approach for nonlinear systems can be developed by making this step explicit.

It will then be shown how steps 3 and 4 can be regarded as two dynamic optimization problems that can be efficiently solved as discussed in section II.

A. Explicit model correction

The quadratically optimal ILC algorithm approximates the next iteration tracking error as $\hat{\mathbf{e}}_{i+1} = \mathbf{y}_r - \mathbf{y}_i - \hat{\mathbf{P}}\delta_i$. Using $\delta_i = \mathbf{u}_{i+1} - \mathbf{u}_i$ and rearranging leads to:

$$\hat{\mathbf{e}}_{i+1} = \mathbf{y}_r - [\hat{\mathbf{P}}\mathbf{u}_{i+1} + (\mathbf{y}_i - \hat{\mathbf{P}}\mathbf{u}_i)]. \quad (19)$$

The term $\hat{\mathbf{P}}\mathbf{u}_{i+1}$ is the expected next iteration output based on the nominal model. Therefore it is clear that for the quadratically optimal ILC algorithm, a correction term $(\mathbf{y}_i - \hat{\mathbf{P}}\mathbf{u}_i)$ is implicitly added to the modeled output, in order to estimate the next iteration tracking error better than using the nominal model only. To construct the matrices $\mathbf{Q}$ and $\mathbf{L}$ the model should be linear, such that it can be written in lifted form by the matrix $\hat{\mathbf{P}}$.

The same principle can however be applied

to nonlinear models if the correction term is made explicit. For example, the next iteration tracking error can be approximated as follows:

$$\hat{\mathbf{e}}_{i+1} = \mathbf{y}_r - [\hat{P}(\mathbf{u}_{i+1}) + \boldsymbol{\alpha}_i] \quad (20)$$

with $\boldsymbol{\alpha}_i$ a correction signal to be estimated after the previous iteration. The sum of the nominal model and the correction signal can be considered as a corrected model $\hat{P}_c(\mathbf{u}_{i+1})$. It is clear that if $\hat{P}(\mathbf{u}_{i+1})$ is a linear model, and $\boldsymbol{\alpha}_i$ is estimated as $\mathbf{y}_i - \hat{\mathbf{P}}\mathbf{u}_i$, then the approximation of $\hat{\mathbf{e}}_{i+1}$ is the same as for the quadratically optimal linear ILC. However, explicitly estimating $\boldsymbol{\alpha}_i$ makes it possible to manipulate this signal, for example to constrain it, or to change its features in time- or frequency domain before it is used to approximate $\hat{\mathbf{e}}_{i+1}$. Another advantage is that in this case the correction signal is not necessarily additive to the model's output, as it is for the linear ILC approach.

Assume the nominal model $\hat{P}(\mathbf{u})$ is given in the form of (18). Then $\boldsymbol{\alpha}$ can enter the corrected model $\hat{P}_c(\mathbf{u})$ in several ways. For example $\boldsymbol{\alpha}$ can be additive to the model output, $\hat{P}_c(\mathbf{u}) = \hat{P}(\mathbf{u}) + \boldsymbol{\alpha}$, or additive to the input, $\hat{P}_c(\mathbf{u}) = \hat{P}(\mathbf{u} + \boldsymbol{\alpha})$, or in even more complex forms. Regardless of the choice of how to correct the nominal model, $\hat{P}_c(\mathbf{u})$ can be written

as

$$\hat{P}_c : \begin{cases} x(k+1) = f[x(k), u(k), \alpha(k)] \\ y(k) = h[x(k), u(k), \alpha(k)] \end{cases},$$

and the signal α can be considered to be an additional input. This means that it is possible to formulate an optimal control problem for the estimation of α , analogous to the problem of finding an optimal input signal. The solution of these 2 problems constitute steps 3) and 4) of the generalized ILC approach.

B. Two step procedure

The goal of model correction is to make the corrected model $\hat{P}_c(\mathbf{u})$ correspond better with the real system than the nominal model $\hat{P}(\mathbf{u})$. Therefore the aim of the ILC algorithm is to find a pair of vectors $(\mathbf{u}^*, \alpha^*)$ that correspond to $\hat{P}_c(\mathbf{u}^*, \alpha^*) = P(\mathbf{u}^*) = \mathbf{y}_r$. However, $\mathbf{u}^*$ and α^* cannot be found simultaneously because it is unknown beforehand how well $\hat{P}$ corresponds to the real system P , and therefore it is impossible to differentiate between both signals. The solution lies in solving the problem alternatively for α and for $\mathbf{u}$, using the other signal as a known exogenous input, and repeating these two steps for each trial in an iterative manner. In this way α will drive the corrected model $\hat{P}_c$ closer to the true system P since $\alpha_k \approx \alpha^*$ for increasing k , while $\mathbf{u}$ will drive the model output $\hat{\mathbf{y}}$ closer

to the reference $\mathbf{y}_r$, so $\mathbf{u}_k \approx \mathbf{u}^*$ for increasing k .

Note that both steps are separate instances of the same problem. Therefore both steps can be written in the general problem formulation of section II-A. The formulation requires three signals to be defined:

- $\bar{\mathbf{e}}$: the signal to be minimized in the objective function
- $\bar{\mathbf{u}}_c$: the control input, which is the signal to be calculated
- $\bar{\mathbf{w}}$: the exogenous input, known beforehand

1) *Step 1: estimation*: The aim of this step, after trial i , is to find a value for the model correction signal α_i , such that the modeled output $\hat{\mathbf{y}}$ follows the actual measurement $\mathbf{y}_i$ more closely. Therefore the following signals are defined:

$$\text{Estimation step: } \begin{cases} \bar{\mathbf{e}} = \mathbf{y}_i - \hat{\mathbf{y}} \\ \bar{\mathbf{u}}_c = \alpha \\ \bar{\mathbf{w}} = \mathbf{u}_i \end{cases}, \quad (21)$$

and the weighting matrices in (10) are written as $\mathbf{Q}_e, \mathbf{R}_e$ and $\mathbf{S}_e$. Since α is the control input for this step, the regularization term $\bar{\mathbf{u}}_c^T \mathbf{R}_e \bar{\mathbf{u}}_c$ can be used to tune and shape α at this stage. Possible applications include a frequency weighting on the correction, to limit

the learning behavior to a certain bandwidth, or manipulations in time domain, such as enabling and disabling learning at specific time intervals along the trial.

It is also possible to define one or more algebraic states $\bar{z}$, and to penalize them in the objective function. This can be used for example to penalize large deviations of α from trial to trial, in order to increase the robustness of the algorithm to non repeating disturbances such as measurement noise.

The solution of the estimation step after trial i is α_i .

2) *Step 2: control*: The aim of this step is to use the estimated α_i in order to find the optimal input signal for the next trial, u_{i+1} . For this step the signals are defined as follows:

$$\text{Control step: } \begin{cases} \bar{e} = y_r - \hat{y} \\ \bar{u}_c = u \\ \bar{w} = \alpha_i \end{cases}, \quad (22)$$

and the weighting matrices are written as Q_c , R_c and S_c . During this step it is possible to use the regularization term $\bar{u}_c^T R_c \bar{u}_c$ to tune the input signal directly. Such a regularization can be necessary in the application of the ILC algorithm to non-minimum phase systems [12], or it can be used to penalize large control efforts. An algebraic state $\bar{z}$ can for example be defined as the time difference of u , such that

$z(k) = u(k) - u(k-1)$, in order to smooth u . Another candidate for $\bar{z}$ is the control signal of a feedback controller, in case the ILC algorithm is applied to a closed loop system with actuator constraints.

During both steps the weight on $\bar{e}$ can be varied over the trajectory. This makes it easy to apply the algorithm to point to point motions, by eliminating the tracking at the time instants during the motion.

The presented algorithm can be used with a broad class of systems and applications. Its strength lies in the flexibility with which the user can adapt the algorithm, by choosing appropriate weighting matrices, constraints, and defining appropriate algebraic states. All these manipulations require no changes to the underlying optimization problems on the side of the user.

C. Linear case: quadratically optimal ILC

The presented algorithm is a generalization of iterative learning control, which means that many common ILC algorithms are special cases of this general approach, that can be found by a suitable choice of model correction and of the weight terms in the objective functions of both steps. For example, consider a linear system and model in matrix form, P and $\hat{P}$. Assume the correction term α is additive to the output, such that the corrected model can

be written as:

$$\hat{P}_c(\mathbf{u}, \boldsymbol{\alpha}) = \hat{\mathbf{P}}\mathbf{u} + \boldsymbol{\alpha} \quad (23)$$

1) *Estimation*: Consider the estimation step with weighting terms $\mathbf{Q}_e = \mathbf{I}$, and $\mathbf{R}_e = 0$, without algebraic states and with $\pm\infty$ as bounds on $\mathbf{v}$. In this case the objective function for this step is:

$$J_\alpha(\boldsymbol{\alpha}_i) = [\mathbf{y}_i - \hat{P}_c(\mathbf{u}_i, \boldsymbol{\alpha}_i)]^T [\mathbf{y}_i - \hat{P}_c(\mathbf{u}_i, \boldsymbol{\alpha}_i)] \quad (24)$$

Taking the derivative of (24) and using (23), the stationary condition for optimality becomes:

$$\frac{\partial J_\alpha}{\partial \boldsymbol{\alpha}_i} = \mathbf{I}(\mathbf{y}_i - \hat{\mathbf{P}}\mathbf{u}_i + \boldsymbol{\alpha}_i) = 0 \quad (25)$$

$$\boldsymbol{\alpha}_i = \mathbf{y}_i - \hat{\mathbf{P}}\mathbf{u}_i. \quad (26)$$

It was shown in section I that the quadratically optimal ILC algorithm implicitly makes a correction equal to (24).

2) *Control*: Consider the control step with arbitrary weighting terms $\mathbf{Q}_c$ and $\mathbf{R}_c$, without algebraic states and with $\pm\infty$ as bounds on $\mathbf{v}$. The objective function becomes:

$$J_{\mathbf{u}}(\mathbf{u}) = [\mathbf{y}_r - \hat{P}_c(\mathbf{u}, \boldsymbol{\alpha}_i)]^T \mathbf{Q}_c [\mathbf{y}_r - \hat{P}_c(\mathbf{u}, \boldsymbol{\alpha}_i)] + \mathbf{u}^T \mathbf{R}_c \mathbf{u} \quad (27)$$

Taking the derivative of (27) and using (23), the stationary condition for optimality becomes:

$$\frac{\partial J_{\mathbf{u}}}{\partial \mathbf{u}_{i+1}} = -\hat{\mathbf{P}}^T \mathbf{Q}_c (\mathbf{y}_r - \hat{\mathbf{P}}\mathbf{u}_{i+1} - \mathbf{y}_i + \hat{\mathbf{P}}\mathbf{u}_i) + \mathbf{R}_c \mathbf{u}_{i+1} = 0,$$

and after rearranging

$$\mathbf{u}_{i+1} = \underbrace{(\hat{\mathbf{P}}^T \mathbf{Q}_c \hat{\mathbf{P}} + \mathbf{R}_c)^{-1} \hat{\mathbf{P}}^T \mathbf{Q}_c \hat{\mathbf{P}}}_{\mathbf{Q}} [\mathbf{u}_i + \underbrace{(\hat{\mathbf{P}}^T \mathbf{Q}_c \hat{\mathbf{P}})^{-1} \hat{\mathbf{P}}^T \mathbf{Q}_c}_{\mathbf{L}} \mathbf{e}_i]. \quad (28)$$

Equation (28) gives the update law that follows from the general ILC approach under the given conditions, and it is clear that the same update law is found as for the quadratically optimal linear ILC algorithm of (1) and (7). The role of the robustness filter $\mathbf{Q}$ is played by a suitably chosen regularization term in the control step, $\mathbf{u}_{i+1}^T \mathbf{R}_c \mathbf{u}_{i+1}$, by penalizing the sample by sample difference in the input signal, denoted by $\boldsymbol{\delta} = [\delta(0), \delta(1), \dots, \delta(N-1)] = [u(0) - u_{init}, u(1) - u(0), \dots, u(N-1) - u(N-2)]$, with u_{init} the input that is required to keep the system at the initial state $\mathbf{x}_{init}$. This is clear in the following example:

Consider a discrete time second order system with transfer function $H_s(z)$ and a sampling time of 1. The resonance frequency of this system is denoted by ω_n . To simulate a model

System	Model
$\frac{0.2918z}{z^2 - 1.592z + 0.892}$	$\frac{0.4357z}{z^2 - 1.412z + 0.8669}$

TABLE I

SIMULATION DATA FOR THE COMPARISON OF THE GENERAL ILC APPROACH WITH A QUADRATICALLY OPTIMAL LINEAR APPROACH

plant mismatch, a model for this system is assumed to have a resonance frequency of $0.8\omega_n$ but equal damping, leading to the transfer function $H_m(z)$. The parameters of both transfer functions are listed in table III-C2. The model is used to design both a quadratically optimal ILC algorithm and an equivalent generalized ILC algorithm, using $\mathbf{Q}_e = \mathbf{I}$, $\mathbf{R}_e = 0$ such that (25) holds (α is assumed to be output additive), and with $\mathbf{Q}_c = \mathbf{I}$ in the control step. If the desired penalty term $\delta^T \mathbf{R}_\delta \delta$ is achieved with a matrix $\mathbf{R}_\delta = r_\delta \mathbf{I}^{N \times N}$, then this term can be written as $\mathbf{u}_{i+1}^T \mathbf{R}_c \mathbf{u}_{i+1}$ by using the following matrix $\mathbf{R}_c$:

$$\mathbf{R}_c = \begin{bmatrix} r_\delta & -r_\delta & 0 & \cdots & 0 \\ -r_\delta & 2r_\delta & -r_\delta & \cdots & 0 \\ 0 & -r_\delta & 2r_\delta & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & r_\delta \end{bmatrix}. \quad (29)$$

The value of r_δ is set to 0.1 for this simulation. Such a choice for the weighting matrix $\mathbf{R}_c$ results in a robustness operator $Q(\cdot)$ that acts as a zero phase low pass filter. This is clear

Fig. 2. System, model and .

from figure 2, which shows the magnitude and phase of the system, the model, and the $Q(\cdot)$ filter. The reference for both algorithms is a smoothed step, using a trapezoidal velocity profile. The result of both approaches is shown in figure 3. The initial input signal is assumed to be zero and is not shown in the figure. In order to slow down the convergence of the ILC algorithm, to better visualize its effect, only a fraction of the update to the input signal is applied for both approaches, and this fraction is chosen as 0.7. It is clear that the quadratically optimal linear ILC algorithm is a special case of the generalized approach. The same performance can be achieved by the latter approach, which has several advantages over the former approach, such as the ability to use nonlinear models, and the ability to take constraints into

Fig. 3. Comparison of the general ILC approach with a quadratically optimal approach for a linear, unconstrained problem.

Fig. 4. Photo (left) and schematic (right) of the lab scale overhead crane

account directly.

IV. EXPERIMENTAL AND NUMERICAL VALIDATION

The generalized ILC approach is validated experimentally and in simulation on a lab scale model of an overhead crane, see fig. 4. The position of the cart is denoted by x_c , the position of the load by x_L . The length of the

pendulum is fixed at l , and the angle of the pendulum with respect to the vertical is θ . x_c is measured with an encoder on the cart, while another encoder in the point where the pendulum is attached measures θ . The goal of the experiment is to control the position of the load in open loop, i.e. without feedback of θ . Therefore the output of the system is x_L . The input signal is the reference signal for a feedback position controller of the cart. Since the pendulum has very little damping, the load must hang perfectly still at the end of the motion. Other feedforward approaches, such as input shaping, rely on exact knowledge of the length of the pendulum to solve this problem. However, it is assumed that this knowledge is unavailable. The pendulum has a length of 45.7cm (and therefore a resonance frequency of 0.74Hz), but the length is modeled as 35cm. The feedback controller for the cart is designed as a PD controller with a bandwidth of 3Hz. The dynamics of the crane are governed by the following nonlinear equation [?]:

$$l \ddot{\theta} + \ddot{x}_c \cos \theta + g \sin \theta = 0. \quad (30)$$

This leads to a 4th order nonlinear state space model, with $\mathbf{x} = [x_c, \dot{x}_c, \theta, \dot{\theta}]^T$. This model is discretized by a forward Euler method, with a sample frequency of 200Hz. The model is augmented by a correction signal α , additive

to the model output. The corrected model $\hat{P}_c$ is given as:

$$\begin{aligned} \mathbf{x}(k+1) &= A(k)\mathbf{x}(k) + B(k)u(k) \quad (31) \\ &\quad + \begin{bmatrix} 0 & 0 & 0 & -0.14 \sin[\theta(k)] \end{bmatrix}^T \\ y(k) &= x_c + l \sin[\theta(k)] + \alpha(k) \end{aligned}$$

with

$$A = \begin{bmatrix} 1 & 0.005 & 0 & 0 \\ -14.25 & -0.073 & 0 & 0 \\ 0 & 0 & 1 & 0.005 \\ 40.99c & 3.07c & 0 & 1 \end{bmatrix} \quad (32)$$

$$B = \begin{bmatrix} 0 & 14.25 & 0 & -40.99c \end{bmatrix}^T \quad (33)$$

and $c = \cos[\theta(k)]$. Two experiments involving tracking control are carried out using the real setup, while an application of a point to point motion is tested in simulation.

A. Experiments

The aim of the experiments is to track a smoothed step of 15cm in 1 second, shown in figure 5, without residual swinging. The first input signal is found by solving the control step with the nominal model, so $\alpha = 0$. The estimation step is performed in the same way for both experiments, using $\mathbf{Q}_e = \mathbf{I}$, and $\mathbf{R}_e = \mathbf{S}_e = 0$. There are no algebraic states, and the bounds on the optimization variables are $\pm\infty$. For the control step, a penalty term is put on the time difference of the input signal,

Fig. 5. Reference load position: smoothed step.

in order to smooth $\mathbf{u}$. Therefore an algebraic state is defined as $\mathbf{z}(k) = \mathbf{u}(k) - \mathbf{u}(k-1)$ for $k = 1, \dots, N$. The weighting matrices are defined as $\mathbf{Q}_c = \mathbf{I}$, $\mathbf{R}_c = 0$, and $\mathbf{S}_c = 0.5\mathbf{I}$. The bounds on $\mathbf{x}$ and $\mathbf{u}$ are $\pm\infty$. The difference between both experiments is the choice of bounds on $\mathbf{z}$: for experiment 1 this bound is constant at ± 0.1 cm/sample, while for experiment 2 this bound is varied over the trajectory.

The result of experiment 1 is shown in figure 6. It is clear that the first motion leads to a large tracking error, due to the model error that was assumed. However, the tracking error reduces over multiple trials, and after 7 trials the tracking is near perfect. The correction signal α has converged to the value $\alpha_\infty = \mathbf{y}_r - \hat{P}(\mathbf{u}_\infty)$, and therefore captures the model error. However, it is clear that the ILC algorithm has also introduced an unwanted vibration in the input signal. The reason is most likely an unmodelled error

Fig. 6. Input signal (top), tracking error (middle) and correction signal (bottom) for the first experiment, for iterations 1, 2 and 7

in the feedback loop of the cart. However, because of the small effect of this vibration on the tracking error, the ILC algorithm cannot correct it, but only prevent its increase.

In order to remove this unwanted behavior in the second experiment, the bound on $\mathbf{z}$ is reduced to 1×10^{-7} before and after the actual motion, as shown in figure ???. The result of the second experiment is shown in figure 8. It is clear that the performance of the algorithm is as good as for the first experiment, but the unwanted vibration of the input signal is suppressed by the bound on $\mathbf{z}$.

Fig. 7. Bound on the sample by sample difference of the input signal during the second experiment (reference shown on top)

Fig. 8. Tracking error (top) and input signal (bottom) for the first experiment, for iterations 1, 2 and 7

B. Simulation

The crane is simulated by a system of the same structure as (31), but derived using the real length $l = 45.7\text{cm}$. This system is used to perform a simulation for a point to point motion of the crane. The objective for the load is to arrive at a point 20cm from the starting point,

Fig. 9. Model correction signal α for the first experiment (top) and the second experiment (bottom), for iterations 1, 2 and 7

after 0.75 second. For the estimation step, the weight on the residual is $\mathbf{Q}_e = \text{diag}(w_e)$, with $w_e = 0$ for $t < 0.75\text{s}$, and $w_e = 1$ for $t \geq 0.75$, and $\mathbf{R}_e = \mathbf{S}_e = 0$. There are no algebraic states, and all bounds are $\pm\infty$. The control step is performed with an extra algebraic state, namely the control signal that is sent by the position controller of the cart. The algebraic state is not penalized in the objective, so $\mathbf{R}_c = \mathbf{S}_c = 0$, but it is bounded in order to constrain the velocity of the cart. The weight on the residual is the same as during the estimation step, so $\mathbf{Q}_c = \mathbf{Q}_e$. The bounds on $\mathbf{x}$ and $\mathbf{u}$ are $\pm\infty$, while the bound on $\mathbf{z}$ is $\pm 0.8\text{V}$. The result of the simulation is shown in figure 9. It is clear that the ILC algorithm converges to a smooth point to point motion without residual

swinging of the pendulum, while satisfying the bound on the cart velocity. The optimal input signal, being the reference to the position controller, is discontinuous, since no smoothing was applied. Note that the estimation of α only depends on the error after 0.75s, while the largest updates to the input signal are before 0.75s, demonstrating the non causality of the ILC algorithm.

V. CONCLUSIONS

This paper discusses a generalized approach to ILC for nonlinear systems. The approach is based on the explicit estimation of a model correction signal. An optimal value of this signal is estimated after each trial, based on the known input and measured output of the past trial. The corrected model is then used to calculate an optimal input signal for the next trial, based on the reference. Both the estimation and the control step are formulated as an NLP problem. A general formulation and solution strategy for this class of problems is also discussed in this paper. It is shown that the formulation leads to sparse and block structured matrices, that can be efficiently processed by IPOPT, a sparse implementation of an interior point method.

The generalized ILC approach is versatile and can be easily adapted by selecting appropriate weights and bounds of the optimization problems, in order to add multiple objectives or

constraints. It can use a broad class of nonlinear models, including black box models, and it is applicable both to tracking control problems and point to point motion control problems. It is shown that a common optimization based linear ILC algorithm is a special case of the generalized approach under certain conditions.

The presented algorithm was validated on a lab scale model of an overhead crane, both experimentally and in simulation. These experiments have demonstrated the ability of the algorithm to compensate for a considerable model plant mismatch by means of model correction, leading to accurate tracking control and smooth point to point motions.

REFERENCES

- [1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operations of robots by learning," *Journal of Robotic Systems*, vol. 1, pp. 123–140, 1984.
- [2] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control: A learning-based method for high-performance tracking control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.
- [3] W. B. J. Hakvoort, "Iterative learning control for ltv systems with applications to an industrial robot," Ph.D. dissertation, Universiteit Twente, Enschede, May 2009.
- [4] C. Chien, "A discrete iterative learning control for a class of nonlinear time-varying systems," *Ieee T Automat Contr*, vol. 43, no. 5, pp. 748–752, Jan 1998.
- [5] M. Arif, T. Ishihara, and H. Inooka, "A learning control for a class of linear time varying systems using double differential of error," *J Intell Robot Syst*, vol. 36, no. 2, pp. 223–234, Jan 2003.
- [6] N. Amann, D. Owens, and E. Rogers, "Iterative learning control for discrete time systems with exponential rates of rate convergence," *Proceeding of The Institution of Electrical Engineers on Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, 1996, accepted for publication. [Online]. Available: <http://eprints.ecs.soton.ac.uk/133/>
- [7] J. Ratcliffe, P. Lewin, E. Rogers, J. Hatonen, and D. Owens, "Norm-Optimal Iterative Learning Control Applied to Gantry Robots for Automation Applications," *Robotics, IEEE Transactions on*, vol. 22, no. 6, pp. 1303–1307, 2006.
- [8] N. Amann, D. H. Owens, and E. Rogers, "Predictive optimal iterative learning control," *International Journal of Control*, vol. 69, no. 2, pp. 203–226, 1998. [Online]. Available: <http://eprints.ecs.soton.ac.uk/673/>
- [9] A. Schöllig, "Optimization-based iterative learning control for trajectory tracking," in *Proceedings of the European Control . . .*, 2009.
- [10] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. Springer, 2006.
- [11] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large- scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [12] S. Gunnarsson and M. Norrlof, "On the design of ilc algorithms using optimization," *Automatica*, vol. 37, no. 12, pp. 2011–2016, Jan 2001.