2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Iterative learning control for nonlinear systems with input constraints and discontinuously changing dynamics

Marnix Volckaert* , Moritz Diehl** , Jan Swevers*

Abstract— **This paper discusses the implementation and application of an iterative learning control (ILC) algorithm for nonlinear systems with input constraints and discontinuously changing dynamics. The ILC approach consists of two steps: in the first step the nominal model of the plant is corrected based on the previous iteration's output, and in the second step the corrected model is inverted to track the reference. Both steps are formulated as nonlinear least squares problems with a sparse, banded structure, and solved using an efficient implementation of an interior point method called IPOPT. Due to this implementation high order systems and long data sets can be efficiently processed. The considered application is a trajectory tracking problem of a one degree-of-freedom robot arm carrying an object that is suddenly released during motion. Both the case where the exact time instant of release is known and unknown are considered.**

I. INTRODUCTION

Many industrial applications require a system to perform a given task repeatedly. The task is often represented by a reference output $\mathbf{y}_r$, that needs to be followed by the system's output $\mathbf{y}$. Iterative learning control (ILC) is an open loop control strategy that exploits the repetition of the task to iteratively improve the input signal $\mathbf{u}$ that is applied to the system.

Initial research into ILC resulted in a general formulation of the update law of the following form [1]:

$$\mathbf{u}^{i+1} = Q[\mathbf{u}^i + L(\mathbf{e}^i)], \tag{1}$$

with $Q(\cdot)$ and $L(\cdot)$ a robustness and learning operator respectively, and $\mathbf{e}^i$ the tracking error, defined as $\mathbf{e}^i = \mathbf{y}_r - \mathbf{y}^i$. Generally regarded as the first ILC algorithm is the work by Arimoto *et al.* [2], for which $L(\mathbf{e}^i) = \Gamma \mathbf{e}^i$ and without robustness operator. More elaborate (model based) algorithms were developed by extending the learning operator to the inverse of a linear model of the system, and choosing $Q(\cdot)$ as an operator to guarantee convergence of the algorithm, such as a zero-phase low pass filter [1]. These methods are based on linear models, and are often formulated in the frequency domain, which makes it hard to adapt them in their current form to nonlinear systems. [3] shows an extension of the general form by introducing a nonlinear learning operator. Other examples of nonlinear ILC algorithms are the Newton-type and secant-type ILC schemes described in [4].

The majority of ILC research focuses on time invariant systems. Efforts to extend the general ILC approach to LTV systems include [5], [6] and [7]. These control methods can be applied to systems with continuously changing dynamic behavior. However, many applications involve systems that have discontinuously changing dynamics, such as pick and place machines that pick up and release objects of a certain mass. Furthermore, an important drawback of these methods is that they cannot take actuator constraints into account.

This paper considers an ILC algorithm that can overcome the shortcomings of the presented existing algorithms. In other words an algorithm that can be applied to nonlinear systems with discontinuously changing dynamics, and that can account for input constraints. The authors have introduced such an algorithm in [8]. The algorithm consists of two steps: a model correction and a model inversion, which are both formulated as a nonlinear least squares problem. It was shown in [8] that a particular case of the introduced algorithm is equivalent to the existing linear quadratically optimal ILC algorithm (LQOILC), as described in [1].

Since [8] focused more on the theoretical background of the algorithm, and on its equivalence with LQOILC under certain conditions, this paper focuses on the practical implementation and validation of the algorithm. More specifically, this paper discusses how an interior point method algorithm for large scale sparse nonlinear programs called IPOPT [9] is used such that large data records can be efficiently processed. The validation case study is a one degree-of-freedom robot arm carrying an object that is suddenly released during motion.

Section II briefly introduces the ILC algorithm described in [8]. Section III discusses the practical implementation of the algorithm using IPOPT. Section IV describes the numerical validation case study and demonstrates the performance of the algorithm, followed by conclusions in section V.

II. NONLINEAR ILC WITH CONSTRAINTS

This section briefly summarizes the two step ILC algorithm that was introduced in [8]. Let us assume that an approximate input-output model $\hat{P}$ of the considered plant P is available. Let $\mathbf{u} = [u_0, u_1, \cdots, u_{N-1}]^T$ and $\mathbf{y} = [y_0, y_1, \cdots, y_{N-1}]^T$, with N the number of samples, denote the input and output signal respectively. Since the purpose of ILC is to find the input signal that tracks a reference output $\mathbf{y}_r$, the problem is essentially to solve for $\mathbf{u}^* = P^{-1}(\mathbf{y}_r)$.

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

** Moritz Diehl is with the Department of Electrical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium.

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).

Since the plant is not know exactly, this problem cannot be solved directly.

The presented algorithm transforms this plant inversion problem into a model inversion problem, using a corrected model $\hat{P}_c$. This corrected model is a function of the nominal model $\hat{P}$ and a correction vector $\boldsymbol{\alpha}$, written as $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \cdots, \alpha_{N-1}]^T$. There are several alternatives to calculate $\hat{P}_c$, such as:

$$\hat{P}_c(\mathbf{u}, \boldsymbol{\alpha}) = \quad \hat{P}(\mathbf{u})+\boldsymbol{\alpha}, \ \ \hat{P}(\mathbf{u}+\boldsymbol{\alpha}), \ \ \mathrm{diag}(\boldsymbol{\alpha})\cdot\hat{P}(\mathbf{u}). \quad (2)$$

The model correction vector $\boldsymbol{\alpha}$ is updated after each iteration based on the input and output data vectors $\mathbf{u}^i$ and $\mathbf{y}^i$ measured during trial i. This model correction step is the first step of the ILC algorithm and consists of calculating the optimal value of $\boldsymbol{\alpha}^i$, such that $\hat{P}_c(\mathbf{u^i}, \boldsymbol{\alpha}^i)$ describes $\mathbf{y^i}$ better than $\hat{P}(\mathbf{u^i})$. It is described in detail in [8] that this corresponds to minimizing the predicted next iteration tracking error, $\mathbf{e}^{i+1}$. The optimization problem that constitutes this first step can be formulated as:

$$\begin{aligned}\boldsymbol{\alpha}^i = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{y}^i - \hat{P}_c(\mathbf{u}^i, \boldsymbol{\alpha})\|^2_{\mathbf{Q}_\alpha} + \|\boldsymbol{\alpha}\|^2_{\mathbf{R}_\alpha} + \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^{i-1}\|^2_{\mathbf{S}_\alpha} \\ \text{s.t.} \qquad\qquad (3) \\ g_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \leq 0,\end{aligned}$$

where $\|\cdot\|$ denotes the weighted 2-norm, $\mathbf{Q}_{\boldsymbol{\alpha}}$ is an $N \times N$ positive-definite diagonal matrix, and $\mathbf{R}_{\boldsymbol{\alpha}}$ and $\mathbf{S}_{\boldsymbol{\alpha}}$ are $N \times N$ positive-semidefinite diagonal matrices. The function $g_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$ is a (non)linear function to constrain the vector $\boldsymbol{\alpha}$.

Regularization with $\mathbf{R}_{\boldsymbol{\alpha}} \neq 0^{N\times N}$ controls the magnitude of $\boldsymbol{\alpha}^i$ along the data record. For example, increasing the weighting of this term at specific samples reduces the model correction only at this part of the signal, which can avoid local instabilities of the ILC algorithm. $\mathbf{S}_{\boldsymbol{\alpha}} \neq 0^{N\times N}$ introduces a memory in the learning dynamics, similar to higher order ILC algorithms, and reduces the sensitivity of $\boldsymbol{\alpha}$ to measurement noise. Knowledge of the model quality can be incorporated through the application of inequality constraints.

The second step of the two step approach is then to calculate the optimal next iteration input signal $\mathbf{u}^{i+1}$. The second step can be formally written as:

$$\begin{aligned}\mathbf{u}^{i+1} = \arg\min_{\mathbf{u}} \ \|\mathbf{y}_r - \hat{P}_c(\mathbf{u}, \boldsymbol{\alpha}^i)\|^2_{\mathbf{Q}_\mathbf{u}} + \|\mathbf{u}\|^2_{\mathbf{R}_\mathbf{u}} + \|\delta\mathbf{u}\|^2_{\mathbf{R}_{\delta\mathbf{u}}} \\ \text{s.t.} \qquad\qquad (4) \\ g_{\mathbf{u}}(\mathbf{u}) \leq 0,\end{aligned}$$

with $\mathbf{Q}_\mathbf{u}$ an $N \times N$ positive-definite diagonal matrix, and $\mathbf{R}_\mathbf{u}$ and $\mathbf{R}_{\delta\mathbf{u}}$ $N\times N$ positive-semidefinite diagonal matrices. The vector $\delta\mathbf{u}$ is defined as $\delta\mathbf{u} = [\delta u_0, \delta u_1, \ldots, \delta u_{N-1}]^T$, with $\delta u_k = u_{k+1} - u_k$. Regularization with $\mathbf{R}_\mathbf{u} \neq 0^{N\times N}$ can be necessary to apply the algorithm to non-minimum phase systems [10]. Through the regularization on $\delta\mathbf{u}$, it is possible to smooth the input $\mathbf{u}^{i+1}$, while $g_\mathbf{u}(\mathbf{u}) \leq 0$ can be used to impose input constraints. The initial input signal that is applied to the system, $\mathbf{u}^0$, is the solution of the model inversion for the nominal model $\hat{P}$, using $\boldsymbol{\alpha} = 0$.

The nominal model $\hat{P}$ can be linear or nonlinear. Furthermore, $\hat{P}$ is not necessarily a fixed model along the data record. E.g. if the system dynamics suddenly change at the discrete time instant $k = k_c$, $\hat{P}$ can be of the following form:

$$\hat{P}(\mathbf{u}) = \begin{cases} \hat{P}_1(\mathbf{u}) & \text{for } k < k_c \\ \hat{P}_2(\mathbf{u}) & \text{for } k \geq k_c, \end{cases} \qquad (5)$$

with $\hat{P}_1(\mathbf{u})$ and $\hat{P}_2(\mathbf{u})$ two different models.

The optimization problems of both step one and two are nonlinear least squares problems, which can be large if long data and model correction vectors are considered. Service load simulation for durability tests in automotive applications is an example of such a large ILC problem [11]. However, both problems have a favorable sparse structure, and hence can be solved efficiently using e.g. IPOPT [9]. This is discussed in the following section.

III. PRACTICAL IMPLEMENTATION

This section discusses the practical implementation of the presented algorithm. The solution of the second step (4) will be described in detail. The solution of the first step (3) is similar.

The optimization problem (4) can be solved efficiently if the matrices that are used in the calculation of the optimal solution have a sparse, banded structure that is exploited by solvers such as IPOPT to speed up the calculations. In order to obtain this favorable structure, it is assumed that the model $\hat{P}$ is a nonlinear discrete-time state space model:

$$\begin{aligned}\mathbf{x}_{k+1} &= f[\mathbf{x}_k, u_k] \\ y_k &= h[\mathbf{x}_k, u_k],\end{aligned} \qquad (6)$$

with $\mathbf{x}_k \in \mathbb{R}^n$ the state vector at time instant k and n the model order. Another important step to introduce sparsity and structure in the optimization problem is to use an augmented variable vector $\mathbf{w}$, that contains the state vectors $\mathbf{x}_k$ at all considered time instances k, the input $\mathbf{u}$, and the input difference vector, $\delta\mathbf{u}$. Simultaneously optimizing on both inputs and states increases the problem size, since in this case $\mathbf{w}$ contains $(n+2)*N$ elements, but also increases sparsity and provides more degrees of freedom to the optimization algorithm. This makes this simultaneous approach more robust compared to the conventional sequential approach to dynamic optimization. This in particular allows to much better treat unstable or even chaotic systems [12].

In this case equation (4) can be written as the standard nonlinear program (NLP):

$$\begin{aligned}\min_{\mathbf{w}} \ f(\mathbf{w}) \quad \text{s.t.} \\ g(\mathbf{w}) \leq 0 \\ h(\mathbf{w}) = 0\end{aligned} \qquad (7)$$

with $f(\mathbf{w})$ the objective function, $g(\mathbf{w})$ the inequality constraints and $h(\mathbf{w})$ the equality constraints. The inequality constraint function $g(\mathbf{w})$ is used to constrain the input signal and contains the following elements:

$$g(\mathbf{w}) = \begin{cases} u_{min} - u_k \\ u_k - u_{max} \end{cases} \text{for } k = 0, \ldots, N-1 \qquad (8)$$

with u_{min} and u_{max} the minimum and maximum input.

The equality constraint function $h(\mathbf{w})$ forces the states to correspond to the model dynamics, and imposes the definition of $\delta\mathbf{u}$, and therefore contains the following elements:

$$h(\mathbf{w}) = \begin{cases} \mathbf{x}_{k+1} - f[x_k, u_k] & \text{for } k = 0, \ldots, N-2 \\ \delta\mathbf{u}_k - \mathbf{u}_{k+1} + \mathbf{u}_k & \text{for } k = 0, \ldots, N-2 \\ \mathbf{x}_0 - f[x_{N-1}, u_{N-1}] \\ \delta\mathbf{u}_{N-1} - \mathbf{u}_0 + \mathbf{u}_{N-1}. \end{cases} \tag{9}$$

The last two equations are included to force the input signal to be periodic, which is a requirement for some applications [11]. An alternative implementation of the algorithm replaces these equations by initial and end state constraints.

A necessary condition for optimality of an optimizer $\mathbf{w}^*$ to problem (7) is the following Karush-Kuhn-Tucker equation [13]:

$$\nabla f(\mathbf{w}^*) - J_g^T(\mathbf{w}^*)\lambda_g - J_h^T(\mathbf{w}^*)\lambda_h = 0, \tag{10}$$

with $\nabla f(\mathbf{w})$ the gradient of $f(\mathbf{w})$, J_g and J_h the Jacobian matrices of $g(\mathbf{w})$ and $h(\mathbf{w})$ respectively, and $\lambda_g, \lambda_h \in \mathbb{R}^N$ the corresponding Lagrange multipliers. Applying Newton's method to (10) provides search directions $\Delta\mathbf{w}$, $\Delta\lambda_g$ and $\Delta\lambda_h$, using the solution of the following equation:

$$\begin{aligned} \nabla^2_{ww}\mathcal{L}\Delta\mathbf{w} - J_g^T(\mathbf{w})\Delta\lambda_g - J_h^T(\mathbf{w})\Delta\lambda_h = \\ \nabla f(\mathbf{w}) - J_g^T(\mathbf{w})\lambda_g - J_h^T(\mathbf{w})\lambda_h \end{aligned} \tag{11}$$

with $\nabla^2_{ww}\mathcal{L}$ the Hessian of the Lagrangian function, defined as $\mathcal{L} = f(\mathbf{w}) - \lambda_g^T g(\mathbf{w}) - \lambda_h^T h(\mathbf{w})$. For nonlinear least squares problems this term can be approximated by $J_f^T J_f$, with J_f the Jacobian of the gradient of the objective function $\nabla f(\mathbf{w})$ [13]. The solution $\Delta\mathbf{w}$ of (11) is used to make iterative steps towards an optimizer $\mathbf{w}^*$ of (7).

The number of elements of the functions $\nabla f(\mathbf{w})$, $g(\mathbf{w})$ and $h(\mathbf{w})$, and of the variable vector $\mathbf{w}$ depends linearly on N, so the dimensions of the Jacobians J_f, J_g and J_h depend quadratically on N. However, each of these matrices has a block diagonal structure, with a limited number of nonzero elements per block, and one block per sample. Therefore the number of nonzero elements only increases linearly with increasing N. The number of elements and the number of nonzeros in $J_f^T J_f$, J_g and J_h is given in table I.

	Nr. of elements	Nr. of nonzeros
$J_f^T J_f$	$(n+2)^2 N^2$	$\left[1 + \frac{(n+1)(n+2)}{2}\right]N$
J_g	$2(n+2)N^2$	$2N$
J_h	$(n+1)(n+2)N^2$	$(n^2+2n+3)N$

TABLE I

TOTAL NUMBER OF ELEMENTS AND NUMBER OF NONZEROS OF THE JACOBIAN MATRICES

Due to this structure and sparsity, equation (11) and hence the NLP (7) can be efficiently solved by a specialized solver such as IPOPT [9], an open source implementation of an interior point algorithm for large scale sparse NLPs, in which case the calculation time to solve problem (11) increases linearly with increasing N.

IV. NUMERICAL VALIDATION

This section discusses the numerical validation of the developed two step ILC algorithm, which is another contribution of this work compared to [8]. The considered case is a robot arm with one rotational degree of freedom which is schematically represented in figure 1. The robot arm moves in the vertical plane. The input is the torque T applied to the arm at the joint, which is limited to the range of ± 12 Nm. The output is the angle θ of the arm measured as shown in figure 1. Joint viscous friction is considered. The dynamics

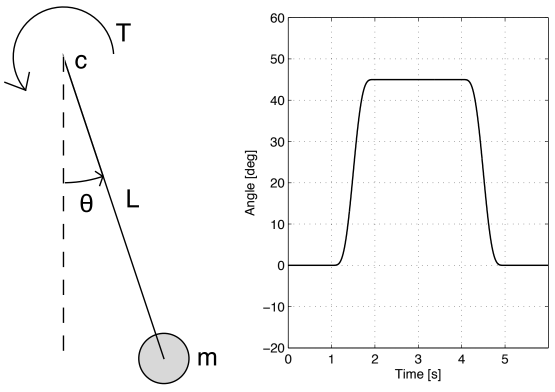

Fig. 1. Schematic drawing of the considered application (left), and desired output signal (right)

of this system can be described by a second order nonlinear state space model with state variables $x_1 = \theta, x_2 = \dot{\theta}, u = T$ and $y = \theta$:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{g}{L}\sin(x_1) - \frac{c}{mL^2}x_2 + \frac{1}{mL^2}u \\ y &= x_1, \end{aligned} \tag{12}$$

with length $L = 1.0$m, mass $m = 1.0$kg and viscous friction coefficient $c = 2.0$Nms/rad, assumed to be known, and $g = 9.81$m/s^2 the gravitational acceleration. m is the equivalent mass of the arm at its end, assuming a massless rod, so the arm's inertia is mL^2. The desired motion of the robot arm is shown in figure 1 on the right, and consists of a smooth forward and backward motion of 45 degrees.

It is now assumed that the robot arm is carrying an object of which the mass is not known exactly, introducing a model-plant mismatch. Furthermore it is assumed that during each repetition of the motion, the mass is suddenly released, so the dynamics of the plant change instantaneously. The equivalent mass m is assumed to be 1.5kg when the object is attached to the arm. Zero mean Gaussian noise with a variance of 4.8×10^{-6} rad^2/s^2 is added to the angular position data θ.

The applied ILC algorithm uses a discrete-time state space model of this system which is obtained by discretizing eq. (12) using the Euler discretization method $\dot{x} = \frac{x(t+1)-x(t)}{T_s}$

with a sampling period $T_s = 0.002$s. The resulting discrete-time model has the following structure:

$$\begin{aligned} x_1(t+1) &= x_1(t) + T_s x_2(t) \\ x_2(t+1) &= -\frac{gT_s}{L}\sin(x_1(t)) + \big[1 - \frac{cT_s}{m_m L^2}\big]x_2(t) \\ &\quad + \frac{T_s}{m_m L^2}u(t) \\ y(t) &= x_1(t), \end{aligned} \tag{13}$$

with m_m the modeled mass. Two different models are used by the ILC algorithm in the experiments described below, for the two distinct configurations of the robot arm: model M1 describes the robot arm with the object, while model M2 is valid after the object is released. For M1, the modeled mass m_m is assumed to be 1.4kg, to introduce a model-plant mismatch. Model M2 is assumed to be exact, using $m_m = 1.0$kg.

Four different experiments are simulated, as summarized in table II. During the first experiment the object is not released, so there is no change in the dynamics of the system. Only model M1 is used, so this experiment serves as a benchmark for the other experiments. During experiment two the object is released but the ILC algorithm uses only model M2, that is, the change in dynamics is not modeled. In experiments three and four both models are used. For experiment three the load release time is assumed to be known exactly, while experiment four examines the performance of the ILC algorithm in case there is a difference in the modeled and actual load release time. For each experiment, 10 iterations are considered.

Nr.	Plant	Model
1	object not released	M1
2	release at 3s	M2
3	release at 3s	M1 $\rightarrow$ M2, Δt = 0s
4	release at 3s	M1 $\rightarrow$ M2, Δt = 0.4s

TABLE II
OVERVIEW OF SIMULATION EXPERIMENTS

The applied ILC algorithm uses the first alternative model correction of eq. (2), that is, $\hat{P}_c(\mathbf{u}, \boldsymbol{\alpha}) = \hat{P}(\mathbf{u}) + \boldsymbol{\alpha}$. No regularization or inequality constraints are used in the first step. In the second step a regularization on $\delta\mathbf{u}$ is used, with $\mathbf{R}_{\delta\mathbf{u}} = 0.001\ I$, and the input is constrained to the interval ± 12 Nm. The reference signal contains 3000 samples. This means that the optimization variable vector contains 9000 elements for the first step, and 12000 elements for the second step. However, due to the efficient implementation, the ILC algorithm needs only 1 second per iteration, running on a 2.4 GHz Intel Core 2 Duo.

Figure 2 shows the results of the first experiment. The introduced model plant mismatch leads to a tracking error in the first iteration, as seen from the bottom left graph. On the right of the figure, it is clear that the tracking error quickly decreases from iteration to iteration, and the ILC algorithm needs four iterations to converge. The tracking error remaining after convergence corresponds to the measurement noise level except during the forward motion, where the input hits the 12 Nm input constraint, as can be seen on the top left graph.

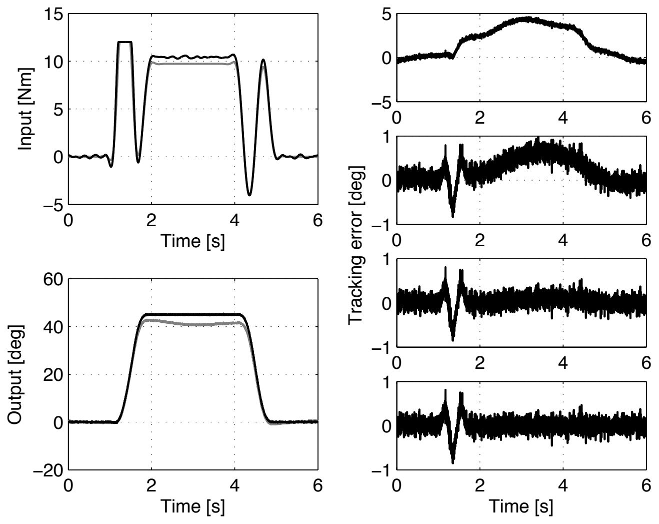

Fig. 2. Experiment 1. TOP LEFT: initial input signal (gray) and converged input signal (black), BOTTOM LEFT: initial output signal (gray) and converged output signal (black), RIGHT (top to bottom): tracking error of the first, second, third and tenth iteration

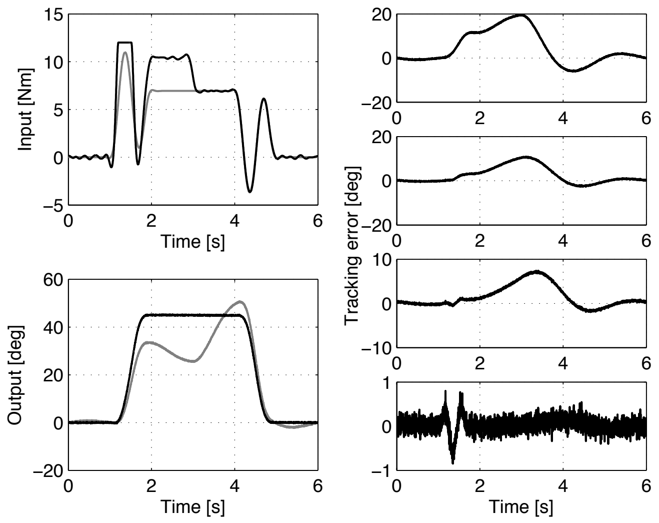

Fig. 3. Experiment 2. TOP LEFT: initial input signal (gray) and converged input signal (black), BOTTOM LEFT: initial output signal (gray) and converged output signal (black), RIGHT (top to bottom): tracking error of the first, second, third and tenth iteration

Figure 3 shows the results of the second experiment. The initial input signal deviates much from the converged input signal during the forward motion, because the model used in the ILC algorithm does not take the extra mass of the object into account: the torque required to accelerate the arm is too small yielding a large initial tracking error. The ILC algorithm has nearly converged after 10 iterations, so the convergence speed is much slower than during the first experiment. After convergence, the input signal during forward motion is similar to that of the first experiment. During the backward motion a smaller input signal is obtained because of the reduced load.

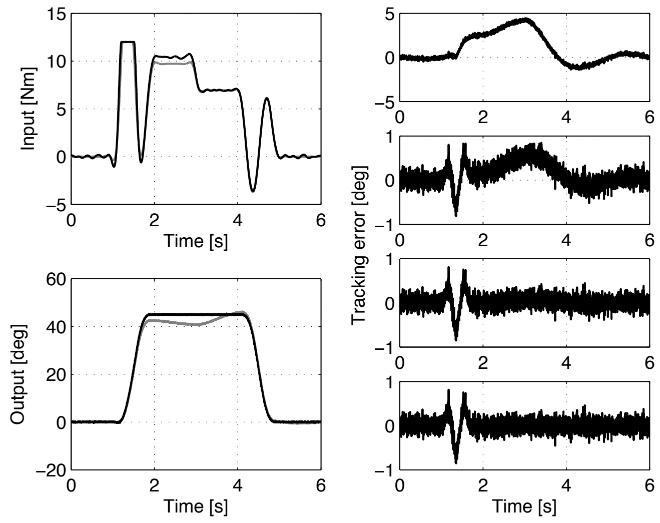

Fig. 4. Experiment 3. TOP LEFT: initial input signal (gray) and converged input signal (black), BOTTOM LEFT: initial output signal (gray) and converged output signal (black), RIGHT (top to bottom): tracking error of the first, second, third and tenth iteration

Figure 4 shows the results of the third experiment. It is clear from the graph of the initial input signal that the ILC algorithm immediately takes the changing dynamics into account. The initial tracking error during the forward motion of this experiment is comparable to that of the first experiment. During the backward motion the initial tracking error is much smaller in this experiment because the applied model M2 is exact. The convergence of the ILC algorithm is comparable to the convergence in the first experiment, so the sudden change in dynamics does not influence the convergence of the ILC algorithm.

The results of the fourth experiment are shown in figure 5. It is clear that the initial input signal is reduced too early, since the time of the release is not modeled accurately. This leads to an initial tracking error that is higher than for experiment three, where the release time is known exactly, but not as high as for experiment two, where no change in dynamics is taken into account by the ILC algorithm. However, the algorithm converges after five iterations. The conclusion is that an error in the estimation of the load release time only results in slower convergence, so the ILC algorithm is robust with respect to this estimation.

Figure 6 shows the evolution of the relative tracking error for all the experiments. It is clear that the converged tracking error is similar during all experiments, at 1%. However, the number of iterations to reach convergence and the initial tracking errors are different for the four experiments. For the first experiment, the initial tracking error is the smallest since the dynamics do not change. The RMS value of the tracking error starts at 8 % and the algorithm converges in three iterations. A similar initial tracking error and convergence

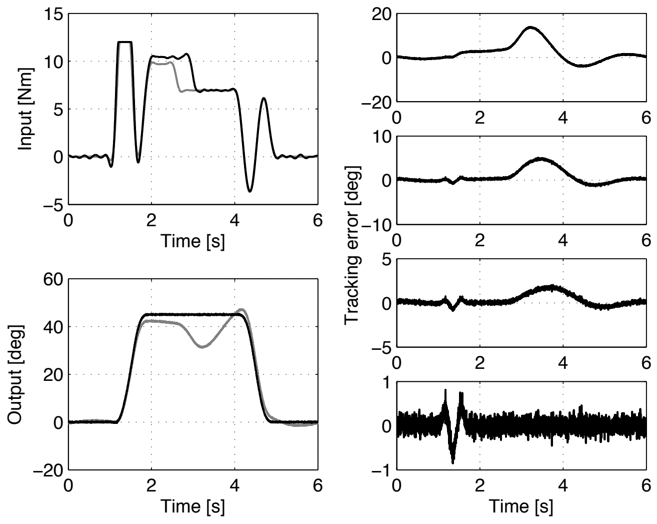

Fig. 5. Experiment 4. TOP LEFT: initial input signal (gray) and converged input signal (black), BOTTOM LEFT: initial output signal (gray) and converged output signal (black), RIGHT (top to bottom): tracking error of the first, second, third and tenth iteration

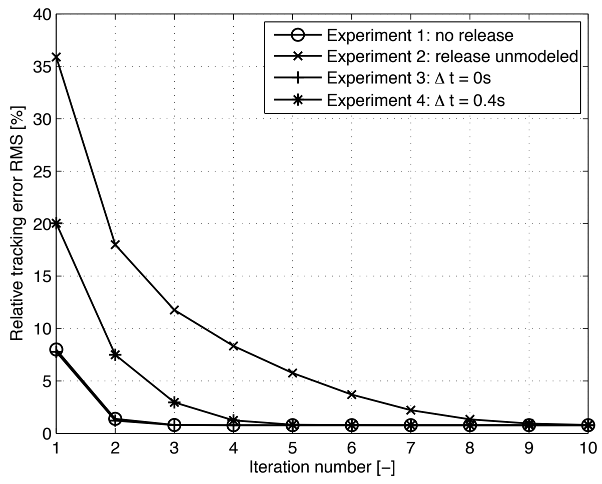

Fig. 6. Evolution of the relative tracking error of the ILC algorithm for all experiments over 10 iterations

speed are obtained during the third experiment. This means that the ILC algorithm can effectively deal with the changing dynamics, without a decrease in convergence speed, if the time of dynamic change is known exactly. An error in the time estimation leads to slower convergence of the ILC algorithm, as can be seen from the graph in figure 6.

Figure 7 shows the model correction vector α for the first three iterations, during both the third and the fourth experiment. It is clear that during the fourth experiment, the model correction vector is larger than during the third experiment, because α must compensate for the error in the load release time estimation. It is also clear that the algorithm needs more iterations to reach the optimal value of α during the fourth experiment, compared to the third experiment.

It can be noted that the change in dynamics for the

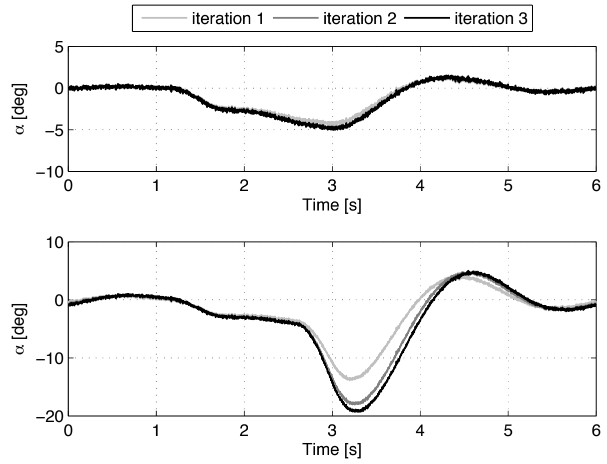

Fig. 7. Model correction vector $\boldsymbol{\alpha}$ for 3 iterations during the third experiment (TOP) and the fourth experiment (BOTTOM)

presented example does not lead to a divergence of the ILC algorithm, even if the change in dynamics is not modeled, such as in the second experiment. However, if the difference between the load mass and arm mass is larger, the change in dynamics becomes larger and the ILC algorithm fails to converge if only model M2 is used. This is clear from figure 8, which shows the evolution of the RMS value of

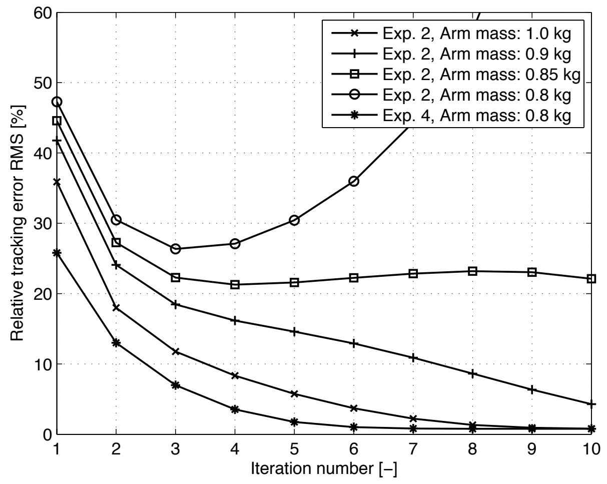

Fig. 8. Evolution of the relative tracking error of the ILC algorithm for different values of the robot arm mass, under the conditions of experiments two and four

the tracking error for different values of the arm mass and a fixed load mass of 1.5kg, under the same conditions as during experiments two and four. For the case when the arm mass is 0.8kg the ILC algorithm diverges if the change in dynamics is not modeled. This means that the ability of the model correction vector to correct this kind of modeling error is limited. However, under the conditions of experiment four, that is, if two models are used, the algorithm converges in seven iterations, even though the load release time is not known accurately.

V. CONCLUSIONS

This paper shows that the two step iterative learning control algorithm presented by the authors in [8] can be solved efficiently, e.g. using IPOPT [9], by formulating both steps such that sparse, block structured NLPs are obtained. Due to this implementation, high order systems and long data sets can be efficiently processed. Numerical validation on a nonlinear system with discontinuously changing dynamics shows (i) that the algorithm can account for changing dynamics by using one or several different models, and (ii) that the convergence speed depends on the accuracy of the available models and the accuracy of the estimated time instant at which the change in dynamics occurs.

REFERENCES

[1] 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.

[2] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operations of robots by learning," *Journal of Robotic Systems*, vol. 1, pp. 123–140, 1984.

[3] M. Heertjes and T. Tso, "Nonlinear iterative learning control with applications to lithographic machinery," *Control Eng Pract*, vol. 15, no. 12, pp. 1545–1555, Jan 2007.

[4] J.-X. Xu and Y. Tan, *Linear and Nonlinear Iterative Learning Control.* Springer, 2003.

[5] W. B. J. Hakvoort, "Iterative learning control for ltv systems with applications to an industrial robot," Ph.D. dissertation, Universiteit Twente, Enschede, May 2009.

[6] 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.

[7] 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.

[8] M. Volckaert, J. Swevers, and M. Diehl, "A two step optimization based iterative learning control algorithm," *ASME Dynamic Systems and Control Conference*, 2010.

[9] 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.

[10] S. Gunnarsson and M. Norrlof, "On the design of ilc algorithms using optimization," *Automatica*, vol. 37, no. 12, pp. 2011–2016, Jan 2001.

[11] J. D. Cuyper, M. Verhaegen, and J. Swevers, "Off-line feed-forward and feedback control on a vibration rig," *Control Engineering Practice*, vol. 11, no. 2, pp. 129 – 140, 2003.

[12] E. Baake, M. Baake, H. Bock, and K. Briggs, "Fitting ordinary differential equations to chaotic data," *Phys. Rev. A*, vol. 45, pp. 5524–5529, 1992.

[13] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. Springer, 2006.