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

Semi-Active Iterative Learning Control

Sandipan Mishra* and Andrew Alleyne**

Abstract— **This paper presents an Iterative Learning Control (ILC) algorithm for iterative parameter update in a semi-active system. The ILC law is designed to minimize a cost function, for example, the mean squared tracking error. First, a parametrized lifted domain representation of a linear parameter-varying system is developed explicitly. Based on this lifted domain representation and a cost function, gradient-based laws for the parameter update from iteration to iteration are proposed. Stability, monotonicity, steady state error, and robustness properties of these algorithms are presented. Finally, an application of the proposed algorithm is illustrated through the simulation of a plastic blow molding system.**

Index Terms— **Iterative Learning Control, Semi-Active Systems**

I. Introduction

Iterative Learning Control (ILC) is based on the idea that the performance of a system which executes the same task multiple times can be improved by learning from the previous executions (trials) of the same task. Such a strategy has the advantage of overcoming imperfect knowledge of the dynamics of the structure to achieve near perfect tracking through repetition. In this technique, the error and input signals from previous trials are stored in memory and used to modify the current input signal. ILC has been widely used in control of repetitive processes because of its simplicity of design, analysis and ease of implementation. The first rigorous formulation of ILC was developed by Arimoto [1] for robotic manipulators. Since then, ILC has found application in a wide variety of manufacturing processes. An interesting area that has seen little research is the use of ILC for update of parameters in a system. This is particularly because during the learning transient, the system may become unstable *within* a trial. However, in the case of semi-active systems, this problem is alleviated since we restrict ourselves to modulation of parameters associated with rate of dissipation of energy in the system.

The use of semi-active systems for vibration control has been well known to members of the dynamics and vibration communities. In particular, it has been used successfully for disturbance rejection. In a fully active system, energy can be supplied to the overall system from a controlled actuator. A semi-active system is one in which energy can only be removed but with a controlled rate of removal. There have been several investigations into the control of systems using semi-active means. In the civil engineering community, first Electrorheological (ER) dampers [2] and subsequently Magnetorheological (MR) dampers [3] were examined for use in seismic response reduction. The MR dampers are currently favored because of their greater actuation capabilities. In the vehicle dynamics community, semi-active suspensions were intensely scrutinized as attractive options to expensive and complex fully active suspensions [4]. The types of semi-active suspensions included ER and MR fluid actuators as well as variable orifice dampers. An example of such a system is a blow molding system, where the plunger follows a desired profile. This profile depends on the part to be molded and the resistance to motion is also dependent on the same. By adjusting the damping coefficient dynamically, the plunger motion can be controlled in a semi-active regime. We use this system to demonstrate the semi-active parameter learning algorithm.

Traditionally ILC algorithms are designed to iteratively update *input control signals* to achieve or optimize some performance objective [5]. On the other hand, in semi-active ILC the performance objective is optimized by iteratively updating *controllable parameters* in the system, such as variable-damping dashpots, variable-stiffness springs, and potentiometers. The parameter update law is derived from a gradient-based optimization scheme. In order to obtain the gradient, closed form expressions of the cost function in terms of the controllable parameters are necessary. To this end, we first propose a lifted domain input-output implicit expression involving the controllable parameters. The gradient-based parameter update laws from iteration to iteration can then be obtained to guarantee cost function decrease. We present three alternate learning laws: (a) P-type, (b) Inverse-type, and (c) Adjoint-type. The choice of learning law depends on the availability of model information. We also present stability, monotonicity, and steady state performance results for these learning laws.

This paper is organized as follows. Section II presents the set up and framework for lifted domain analysis of the semi-active ILC problem. Section III introduces gradient-based ILC algorithms for the semi-active ILC problem. Section IV discusses an example application of the proposed method to blow molding. Finally, in Section V conclusions and discussion of the semi-active ILC algorithm are presented.

II. Problem Definition

Consider a single input single output (SISO) linear time-invariant (LTI) continuous time (CT) system described by the transfer function $G(s) = \frac{B(s)}{A(s)} = \frac{\sum_{i=0}^{m} b_i s^i}{s^n + \sum_{j=1}^{n-1} a_j s^j}$.The correspond-

*Mechanical Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY, e-mail: mishrs2@rpi.edu.

**Mechanical Science and Engineering, University of Illinois Urbana, IL

ing differential equation is therefore given by

$$\sum_{i=0}^{m} b_i \frac{d^i u}{dt^i} = \frac{d^n y}{dt^n} + \sum_{j=1}^{n-1} a_j \frac{d^j y}{dt^j} \tag{1}$$

Henceforth, for notational simplicity we will refer to the n-th derivative of a signal v as $v^{(n)}$. If we sample the input and output signals at a sampling rate of T_s seconds, we have for the time instant j :

$$\begin{aligned} b_m u^{(m)}(j) + b_{m-1} u^{(m-1)}(j) + \ldots + b_1 u^{(1)}(j) + b_0 u(j) = \\ y^{(n)}(j) + a_{n-1} y^{(m-1)}(j) + \ldots + a_1 y^{(1)}(j) + a_0 y(j) \end{aligned} \tag{2}$$

The lifted system formulation provides a method for analyzing the behavior of *linear sampled time* repetitive processes, such as ILC. This formulation exploits the finiteness of the length of each trial to reduce the ILC problem into a finite-dimensional design problem. The lifted form of the output (input) vector is obtained by stacking all the sampled outputs (inputs) over the entire (k-th) cycle, as shown below.

$$\mathbf{y}_k = \begin{bmatrix} y_k(0) & y_k(1) & \ldots & y_k(N-1) \end{bmatrix}^T \tag{3}$$

$$\mathbf{u}_k = \begin{bmatrix} u_k(0) & u_k(1) & \ldots & u_k(N-1) \end{bmatrix}^T \tag{4}$$

where N is the length (number of sampled time steps) of each trial. Next, we will find an approximation of the lifted vector forms for the derivatives of the output and input.

The Lifted Derivative Matrix $\mathbf{D}$

The derivative of a signal v at step j can be determined (in discrete time) by several alternate approximations (forward difference (subscript fd), backward difference (bd), central difference (cd), etc.):

$$\begin{aligned} v_{fd}^{(1)}(j) &= \frac{1}{T_s}\left(v(j+1) - v(j)\right) \\ v_{bd}^{(1)}(j) &= \frac{1}{T_s}\left(v(j) - v(j-1)\right) \\ v_{cd}^{(1)}(j) &= \frac{1}{2T_s}\left(v(j+1) - v(j-1)\right) \end{aligned}$$

Assuming zero-initial and final conditions, these expressions are *linear* combinations of the signal (v). Therefore, in lifted form the derivative of the signal v is:

$$\mathbf{v}^{(1)} = \mathbf{D}\mathbf{v} \tag{5}$$

$$\begin{aligned} \mathbf{D}_{fd} &= \frac{1}{T_s}\begin{bmatrix} -1 & 1 & 0 & \ldots & 0 \\ 0 & -1 & 1 & \ldots & 0 \\ 0 & 0 & -1 & 1 & \ldots \\ 0 & \ldots & 0 & -1 & 1 \\ 0 & \ldots & 0 & 0 & -1 \end{bmatrix} \\ \mathbf{D}_{bd} &= -\mathbf{D}_{fd}^T \\ \mathbf{D}_{cd} &= \frac{1}{2T_s}\begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ -1 & 0 & -1 & \ldots & 0 \\ 0 & 1 & 0 & -1 & \ldots \\ 0 & \ldots & 1 & 0 & -1 \\ 0 & \ldots & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

The second derivative can be determined by combining the finite, backward, or central difference formulae twice:

$$v^{(2)}(j) = \frac{1}{T_s^2}\left(v(j+1) - 2v(j) + v(j-1)\right) \tag{6}$$

In terms of the lifted vectors, these equations can be written as $\mathbf{v}^{(2)} = \mathbf{D}_i\mathbf{D}_j\mathbf{v}$. The subscripts i and j refer to the type of approximation used. If we use the same approximation type twice, $\mathbf{v}^{(2)} = \mathbf{D}_i^2\mathbf{v}$. This reasoning can be extended to obtain approximations for lifted forms of higher derivatives of the signal. If we choose the same approximation for all the derivatives, then the approximation of an n^{th} derivative in lifted domain is $\mathbf{D}^n$. It is important to note that these approximations have the underlying assumption that the continuous time signals are approximated by zero-order holds.

Lifted System Description

Based on the discussion above, we can now describe the lifted formulation of the system shown in Equation 2 as:

$$\begin{aligned} b_m\mathbf{u}^{(m)} + b_{m-1}\mathbf{u}^{(m-1)} + \ldots + b_1\mathbf{u}^{(1)} + b_0\mathbf{u} = \\ a_n\mathbf{y}^{(n)} + a_{n-1}\mathbf{y}^{(m-1)} + \ldots + a_1\mathbf{y}^{(1)} + a_0\mathbf{y} \end{aligned} \tag{7}$$

$$\begin{aligned} \Rightarrow b_m\mathbf{D}^m\mathbf{u} + b_{m-1}\mathbf{D}^{m-1}\mathbf{u} + \ldots + b_1\mathbf{D}\mathbf{u} + b_0\mathbf{u} = \\ a_n\mathbf{D}^n\mathbf{y} + a_{n-1}\mathbf{D}^{n-1}\mathbf{y} + \ldots + a_1\mathbf{D}\mathbf{y} + a_0\mathbf{y} \end{aligned} \tag{8}$$

The lifted-system representation for the system in Equation 2 is therefore

$$\begin{aligned} \mathbf{y} = \left(a_n\mathbf{D}^n + a_{n-1}\mathbf{D}^{n-1} + \ldots + a_1\mathbf{D} + a_0\mathbf{I}\right)^{-1} \\ \left(b_m\mathbf{D}^m + b_{m-1}\mathbf{D}^{m-1} + \ldots + b_1\mathbf{D} + b_0\mathbf{I}\right)\mathbf{u} \end{aligned} \tag{9}$$

It is important to note that this lifted system representation uses the transfer function instead of the impulse response coefficients of the system [6].

Now, consider a linear differential equation with time-varying coefficients, shown in Equation 10.

$$\begin{aligned} b_m(j)u^{(m)}(j) + b_{m-1}(j)u^{(m-1)}(j) + \ldots + b_0(j)u(j) = \\ a_n(j)y^{(n)}(j) + a_{n-1}(j)y^{(m-1)}(j) + \ldots + a_0(j)y(j) \end{aligned} \tag{10}$$

The corresponding lifted system representation for the system (with zero initial and final conditions) is

$$\begin{aligned} \mathbf{y} = \left(diag(\mathbf{a}_n)\mathbf{D}^n + diag(\mathbf{a}_{n-1})\mathbf{D}^{n-1} + \ldots + diag(\mathbf{a}_0)\mathbf{I}\right)^{-1} \\ \left(diag(\mathbf{b}_m)\mathbf{D}^m + diag(\mathbf{b}_{m-1})\mathbf{D}^{m-1} + \ldots + diag(\mathbf{b}_0)\mathbf{I}\right)\mathbf{u} \end{aligned} \tag{11}$$

where

$$\mathbf{a}_i = \begin{bmatrix} a_i(0) & a_i(1) & \ldots & a_i(N-1) \end{bmatrix}^T \tag{12}$$

$$\mathbf{b}_j = \begin{bmatrix} b_j(0) & b_j(1) & \ldots & b_j(N-1) \end{bmatrix}^T \tag{13}$$

Note: In the above development we have dropped the subscript k corresponding to the trial number for clarity. Subsequently, we will add the subscript k to all trial-varying lifted vectors (or matrices).

III. Gradient-based semi-active ILC

Consider the lifted system shown in Equation 11. We assume that all parameters are constant other than the controllable parameters (1) a_p (associated with the p^{th} derivative of y) and (2) b_q (associated with the q^{th} derivative of u). The lifted system equation can then be obtained from Equation 11 as

$$\mathbf{y}_k = \left(a_n\mathbf{D}^n + \ldots + diag(\mathbf{a}_{p,k})\mathbf{D}^p + \ldots + a_0\mathbf{I}\right)^{-1} \left(b_m\mathbf{D}^m + b_{m-1}\mathbf{D}^{m-1} + diag(\mathbf{b}_{q,k})\mathbf{D}^q + \ldots + b_0\mathbf{I}\right)\mathbf{u} \quad (14)$$

The goal of the learning law is to iteratively adjust the controlled parameters $\mathbf{a}_{p,k}$ and $\mathbf{b}_{q,k}$ after each iteration so that a cost function J is minimized.

$$\mathbf{a}_{p,k+1} = \mathbf{a}_{p,k} + \gamma_{p,k}\mathbf{s}_{p,k} \quad (15)$$

$$\mathbf{b}_{q,k+1} = \mathbf{b}_{q,k} + \gamma_{q,k}\mathbf{s}_{q,k} \quad (16)$$

where $\mathbf{s}_{p,k}$ and $\mathbf{s}_{q,k}$ are the search directions. For a gradient-based search, these must be chosen such that $\mathbf{s}_{p,k}^T\nabla_{a_p}(J) + \mathbf{s}_{q,k}^T\nabla_{b_q}(J) < 0$ at each step k.

For tracking a reference trajectory $\mathbf{y}_d$, a typical quadratic cost function is

$$J = \frac{1}{2}(\mathbf{y}_d - \mathbf{y})^T(\mathbf{y}_d - \mathbf{y}) = \frac{1}{2}\mathbf{e}^T\mathbf{e} \quad (17)$$

where $\mathbf{y}_d$ is the desired trajectory to be tracked. The gradients of the cost function with respect to the parameters $\mathbf{a}_p$ and $\mathbf{b}_q$ at the k^{th} iteration are

$$\begin{bmatrix} \nabla_{a_p}J \\ \nabla_{b_q}J \end{bmatrix}_k = \begin{bmatrix} -\mathbf{A}_k\ diag(\mathbf{D}^p\mathbf{y}_{k+1}) \\ \mathbf{A}_k\ diag(\mathbf{D}^q\mathbf{u}) \end{bmatrix} \quad (18)$$

where $\mathbf{A}_k = \left(a_n\mathbf{D}^n + \ldots + diag(\mathbf{a}_{p,k})\mathbf{D}^p + \ldots + a_0\mathbf{I}\right)^{-1}$. For additional adjustable parameters, the extension is simple (adding rows to the expression above). With an update law of the form $\mathbf{a}_{p,k+1} = \mathbf{a}_{p,k} + \delta\mathbf{a}_{p,k}$ and $\mathbf{b}_{q,k+1} = \mathbf{b}_{q,k} + \delta\mathbf{b}_{q,k}$; the error evolution equation becomes

$$\mathbf{e}_{k+1} = \mathbf{e}_k + \mathbf{A}_k diag\left(\mathbf{D}^p\mathbf{y}_{k+1}\right)\delta\mathbf{a}_{p,k} - \mathbf{A}_k diag\left(\mathbf{D}^q\mathbf{u}\right)\delta\mathbf{b}_{q,k} \quad (19)$$

The following observations are important:

1) The error evolution equation is linearly dependent on $\delta\mathbf{a}_{p,k}$ and $\delta\mathbf{b}_{q,k}$.
2) The error evolution equation depends on $\mathbf{y}_{k+1}$. While this is not realizable before implementation, this expression can be used for convergence and robustness analysis.

Consider a general learning law of the form

$$\begin{bmatrix} \mathbf{a}_{p,k+1} \\ \mathbf{b}_{q,k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{p,k} \\ \mathbf{b}_{q,k} \end{bmatrix} + \begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix}\mathbf{e}_k \quad (20)$$

Claim: The learning law in Eq. 20 is monotonically stable in the 2-norm sense [5] (i.e. $\mathbf{e}_k \to 0 as k \to \infty$ and $\|\mathbf{e}_{k+1}\|_2 < \|\mathbf{e}_k\|_2$) if

$$\left\|\mathbf{I} - \mathbf{A}_k\begin{bmatrix} -diag(\mathbf{D}^p\mathbf{y}_{k+1}) & diag(\mathbf{D}^q\mathbf{u}) \end{bmatrix}\begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix}\right\|_2 < 1 \quad (21)$$

Proof: Using the update law (Eq. 20) and the error evolution equation (Eq. 19), we get

$$\mathbf{e}_{k+1} = \left[\mathbf{I} - \mathbf{A}_k\begin{bmatrix} -diag(\mathbf{D}^p\mathbf{y}_{k+1}) & diag(\mathbf{D}^q\mathbf{u}) \end{bmatrix}\begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix}\right]\mathbf{e}_k \quad (22)$$

Therefore, for monotonic convergence in the 2-norm sense, we need

$$\left\|\mathbf{I} - \mathbf{A}_k\begin{bmatrix} -diag(\mathbf{D}^p\mathbf{y}_{k+1}) & diag(\mathbf{D}^q\mathbf{u}) \end{bmatrix}\begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix}\right\|_2 < 1 \quad (23)$$

Typical Learning Laws

We now present some typical semi-active learning laws inspired by traditional ILC algorithms.

1. Proportional Learning Law

$$\begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix} = \begin{bmatrix} K_p \\ K_q \end{bmatrix} \quad (24)$$

The proportional (P-type) learning law is simple for implementation but not guaranteed to be *monotonically stable*.

2. Adjoint Learning Law

$$\begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix} = \frac{1}{2\sigma(\bar{\mathbf{A}}_k)^2}\begin{bmatrix} -\frac{1}{max(\mathbf{D}^p\mathbf{y}_{k+1})}diag(sign(\mathbf{D}^p\mathbf{y}_{k+1})) \\ \frac{1}{max(\mathbf{D}^q\mathbf{u})}diag(sign(\mathbf{D}^q\mathbf{u})) \end{bmatrix}\mathbf{A}_k^T \quad (25)$$

This learning law does not require explicit knowledge of the exact value of $\mathbf{D}^p\mathbf{y}_{k+1}$ or $\mathbf{D}^q\mathbf{u}$, rather we need to know only the sign and the maximum possible value of these signals, which are more reasonable to obtain.

3. Inverse Learning Law

$$\begin{bmatrix} \mathbf{L}_{p,k} \\ \mathbf{L}_{q,k} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2max(\mathbf{D}^p\mathbf{y}_{k+1})}diag(sign(\mathbf{D}^p\mathbf{y}_{k+1})) \\ \frac{1}{2max(\mathbf{D}^q\mathbf{u})}diag(sign(\mathbf{D}^q\mathbf{u})) \end{bmatrix}\mathbf{A}_k \quad (26)$$

The inverse learning law can be implemented only when we have knowledge of $\mathbf{A}_k$. Further, typically $\mathbf{A}_k$ has very large singular values because of the derivative action of $\mathbf{D}_m$ and may cause amplification of high-frequency noise. However, it guarantees monotonic stability of the error.

Convergence Rates and Steady State Performance

We now investigate convergence and steady state performance of the semi-active parameter update law with a single controllable parameter $\mathbf{a}_p$. Extensions to multiple parameters are fairly straightforward but are omitted for brevity. With a single controllable parameter, we have,

$$\mathbf{e}_{k+1} = \left[\mathbf{I} + \mathbf{A}_k diag(\mathbf{D}^p\mathbf{y}_{k+1})\mathbf{L}_{p,k}\right]\mathbf{e}_k = \mathbf{F}\mathbf{e}_k \quad (27)$$

If $\mathbf{F}$ is a strict contraction in the 2-norm sense, i.e., $\bar{\sigma}(\mathbf{F}) < 1$, then $\mathbf{e}_k \to 0$. This condition is standard for monotonic stability in ILC [5]. We now present conditions under which the adjoint and inverse laws for parameter update are monotonically stable.

Adjoint Learning Law For the adjoint learning case

$$\mathbf{F} = \mathbf{I} - \frac{1}{\bar{\sigma}(\mathbf{A}_k)^2}\mathbf{A}_k diag\left(\frac{|\mathbf{D}^p\mathbf{y}_k(j)|}{max(\mathbf{D}^p\mathbf{y}_k(j))}\right)\mathbf{A}_k^T \tag{28}$$

If $\frac{min(|\mathbf{D}^p\mathbf{y}_k(j)|)}{max(\mathbf{D}^p\mathbf{y}_k(j))} > \lambda > 0$ and the condition number of $\mathbf{A}_k = \frac{\underline{\sigma}(\mathbf{A}_k)}{\bar{\sigma}(\mathbf{A}_k)} = \kappa(\mathbf{A}_k)$, then, the convergence rate $\bar{\sigma}(\mathbf{F}) < 1 - \lambda\kappa(\mathbf{A}_k)^2 < 1$. Note that λ is scaled measure of the smallest value of the p^{th} derivative of y. In other words, if the term $|\mathbf{D}^p\mathbf{y}_k(j)| > 0$, then the contraction is strict and hence $\mathbf{e}_k \to 0$. Intuitively, this makes sense since the controllable parameter $\mathbf{a}_p$ affects the dynamics of the system through the corresponding derivative of the output.

Inverse Learning Law For the inverse learning case, we have

$$\mathbf{F} = \mathbf{I} - \mathbf{A}_k diag\left(\frac{|\mathbf{D}^p\mathbf{y}_k(j)|}{max(\mathbf{D}^p\mathbf{y}_k(j))}\right)\mathbf{A}_k^{-1} \tag{29}$$

Defining λ and $\kappa(\mathbf{A}_k)$ as in the adjoint case, we get the convergence rate $\bar{\sigma}(\mathbf{F}) < 1 - \lambda\kappa(\mathbf{A}_k) < 1$.

The key observation from the above analysis is that **the semi-active learning laws can provide monotonic convergence to zero steady state error if the output derivative corresponding to the controllable parameter stays bounded away from zero and has known sign.**

Robustness of Gradient-based ILC Laws

For the problem described in the previous section, one may argue that the optimal solution may be obtained through a one step computation

$$\mathbf{a}_{p,opt} = (diag(\mathbf{D}^p\mathbf{y}_d))^{-1}\left(\left(b_m\mathbf{D}^m + b_{m-1}\mathbf{D}^{m-1} + \ldots + b_0\mathbf{I}\right)\mathbf{u} - \left(a_n\mathbf{D}^n + \ldots + diag(\mathbf{a}_{p,k})\mathbf{D}^p + \ldots + a_0\mathbf{I}\right)\mathbf{y}_d\right)$$

However, this computation will yield suboptimal solutions in the presence of model uncertainty. On the other hand, the iterative nature of the gradient-based learning laws provide robustness to model uncertainty as long as the stability criterion established in Equation 21 is satisfied. Therefore, the repetitive refinement of the optimal variation profile of a controllable parameter results in enhanced robustness of the learning system over a one-shot purely model-based computation of the optimal profile.

IV. Application: Position Profile Control for Blow Molding

Figure 1 shows a schematic of a typical plastic blow molding system. The plunger follows a specified pressure-position profile $\mathbf{y}_d$ to generate the molded part. Typically, the blow molding system produces thousands of *identical* parts per hour. As a result of this repetitiveness of the molding process, ILC-type algorithms have been successfully implemented (see, for example [8]) for performance enhancement in these systems. These investigations have focused on *active* control methods.

In this section, we investigate the use of a semi-active learning algorithm for controlling the plunger position

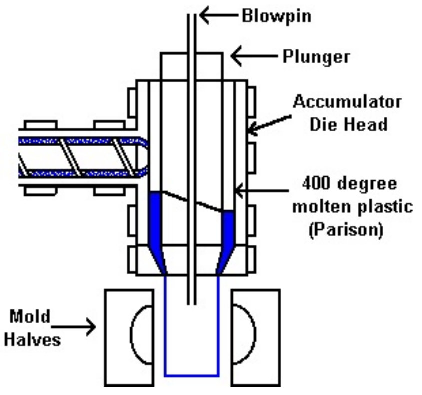

Fig. 1. SCHEMATIC OF BLOW MOLDING SYSTEM (FROM [7])

through adjustment of a system parameter. The plunger system in Figure 2 represents a simplified model of the blow molding system. The mass and spring constant are fixed, where as the damping may be controlled through a semi-active element (a controlled-orifice dashpot). The resistance to the plunger varies based on the part being blow molded. This resistance is denoted by f_r. The plunger moves down by

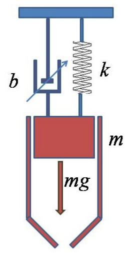

Fig. 2. SCHEMATIC OF VARIABLE-DAMPING SYSTEM.

gravity and its position profile is controlled by changing the damping coefficient dynamically. The objective is to follow the reference trajectory $\mathbf{y}_d$ without any closed loop control or actuation, just by using gravity and changing the damping coefficient as the plunger moves down.

We make the following assumptions about the system:

1) y is measurable: position output.
2) $v = \dot{y}$ is measurable: velocity output.
3) Known parameters: m, K, and $b \in [b_l, b_u]$. b can be controlled/ changed while m and K are fixed.
4) The sampling rate is T_s seconds.

The semi-active plunger system dynamics can be described through the differential equation

$$m\ddot{y}(t) + b(t)\dot{y}(t) + Ky(t) = mg - f_r(t) \tag{30}$$

, where $m = 30$ kg is the mass of the plunger, $K = 1$ N/m is the spring constant, and $f_r(t)$ is chosen to vary between 10 and 50 N [9]. The range for controllable damping coefficient was chosen to be 10 Ns/m to 10000 Ns/m.

The lifted plant system is represented by the equation:

$$\left(m\mathbf{D}^2 + diag(\mathbf{b})\mathbf{D} + K\mathbf{I}\right)^{-1}\mathbf{y} = mg\mathbf{1} - \mathbf{f}_r \tag{31}$$

where **1** is a vector of ones and g is gravitational acceleration. We define our cost function as

$$J = \frac{1}{2}(\mathbf{y}_d - \mathbf{y})^T(\mathbf{y}_d - \mathbf{y}) \tag{32}$$

We use the gradient-based semi-active learning law:

$$\mathbf{b}_{k+1} = \mathbf{b}_k + \gamma_k \mathbf{s}^k \tag{33}$$

We investigate and compare the performance of a P-type and an inverse-type learning law for this system. Figure 3 shows a plot of the reference trajectory $\mathbf{y}_d$ to be followed. Following a similar derivation to the one presented in Section III, we have

$$\nabla_{\mathbf{b}}(J) = \left[\left(m\mathbf{D}^2 + diag(\mathbf{b})\mathbf{D} + K\mathbf{I}\right)^{-1}(diag(\mathbf{D}\mathbf{y}))\right]^T(\mathbf{y}_d - \mathbf{y}) \tag{34}$$

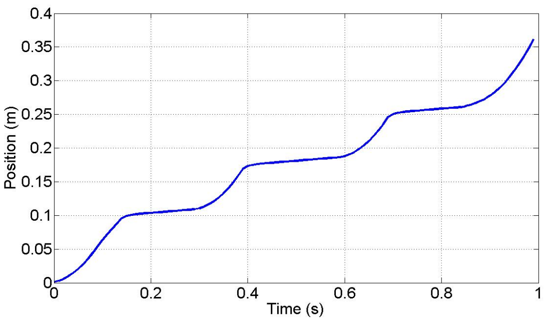

Fig. 3. PLOT OF REFERENCE TRAJECTORY (DESIRED PLUNGER POSITION PROFILE). THE PLUNGER FOLLOWS A TRAJECTORY OF ALTERNATE SEGMENTS FOR FILL AND PACK PHASES. DURING THE FILL PHASE, THE PLUNGER MOVES WITH CONSTANT VELOCITY WHILE DURING THE PACK PHASE, THE PLUNGER IS HELD TO OBTAIN CONSTANT PRESSURE.

First, we use the P-type search algorithm, i.e., the parameter update is proportional to the error; $\mathbf{s}_k = K_p\mathbf{e}_k$. The P-type learning law is given by

$$\mathbf{b}_{k+1} = \mathbf{b}_k + \mathbf{K}_p(\mathbf{y}_d - \mathbf{y}_k) \tag{35}$$

We run the semi-active ILC algorithm for 1000 iterations ($K_p < 0$), and the convergence of the cost function (2-norm of error norm) is shown in Figure 4. Note that the the position error norm decreases, although *non-monotonically*. Further, Figure 5 shows the error profile across a single run for iterations 10, 50, 100, 500, and 1000. We observe that the peak error is limited to less than 25 mm after 100 iterations and less than 8 mm after 1000 iterations. Though the algorithm requires 1000 iterations to converge to 1% error, most of the improvement (80%) is achieved within the first 100 iterations. This performance is acceptable for most blow molding applications.

Further, the damping coefficient converges to the ideal profile as iterations go by, as shown below in Figure 6. The ideal profile was obtained in simulation by assuming that a perfect model is available and using Equation 30.

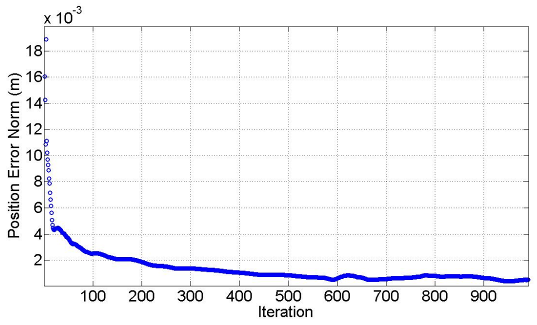

Fig. 4. EVOLUTION OF COST (ERROR 2-NORM) OVER ITERATIONS USING THE P-TYPE LEARNING LAW.

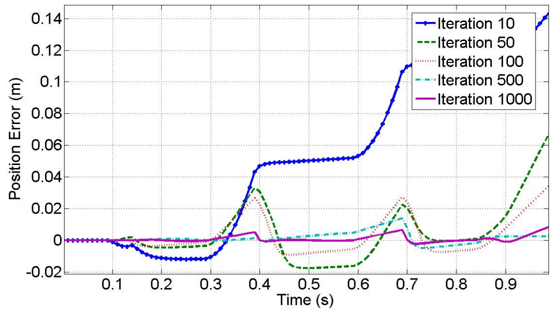

Fig. 5. POSITION ERROR PROFILE ACROSS ITERATIONS 10, 50, 100, 500, AND 1000 USING THE P-TYPE LEARNING LAW.

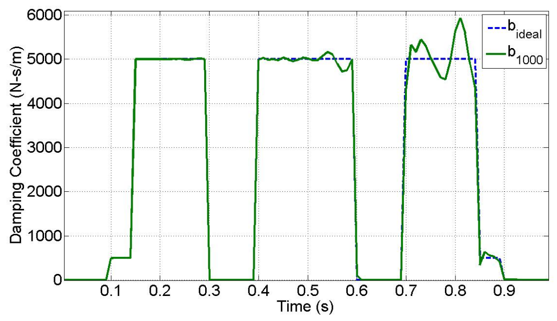

Fig. 6. PLOT OF DAMPING COEFFICIENT ACROSS ITERATION 1000. THE DOTTED LINE SHOWS THE *IDEAL* DAMPING COEFFICIENT PROFILE.

Next, we use an inversion-type semi-active ILC law

$$\mathbf{b}_{k+1} = \mathbf{b}_k - \frac{diag(sign(\mathbf{D}^p\mathbf{y}_k))}{max(\mathbf{D}^p\mathbf{y}_k)}\mathbf{A}_k(\mathbf{y}_d - \mathbf{y}_k) \tag{36}$$

Since we do not have access to $\mathbf{D}^p\mathbf{y}_{k+1}$, we use $\mathbf{D}^p\mathbf{y}_k$ in the learning law above. Note that we only need to know the *sign* of $\mathbf{D}^p\mathbf{y}_{k+1}$, which is always positive.

Figure 7 shows the decrease in error 2-norm over iterations. We notice that the error norm decreases *monotonically*. Further, the error profiles within iterations 10, 50, 100, 500 and 1000 in Figure 8 show that the peak error is reduced to 0.4 mm after 1000 iterations. The inverse-type law yields very fast and monotonic convergence at the cost of requiring accurate model parameter information.

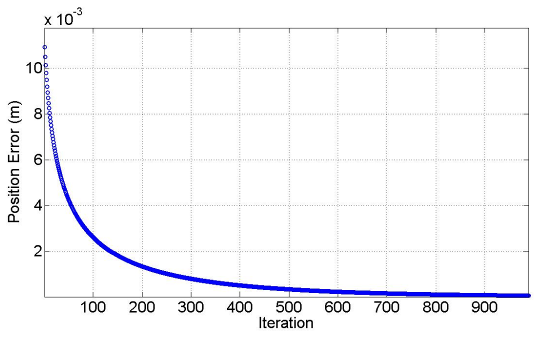

Fig. 7. EVOLUTION OF COST (ERROR 2-NORM) OVER ITERATIONS USING THE INVERSION-TYPE LEARNING LAW.

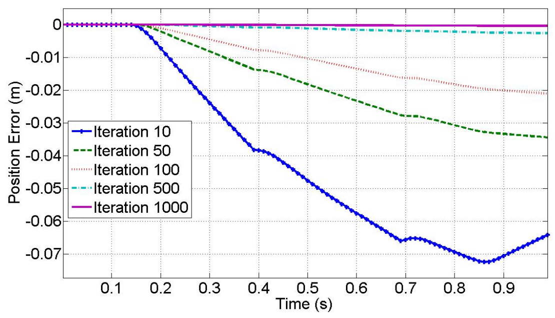

Fig. 8. POSITION ERROR PROFILE ACROSS ITERATIONS $10, 50, 100, 500$, AND 1000 USING THE INVERSION-TYPE LEARNING LAW.

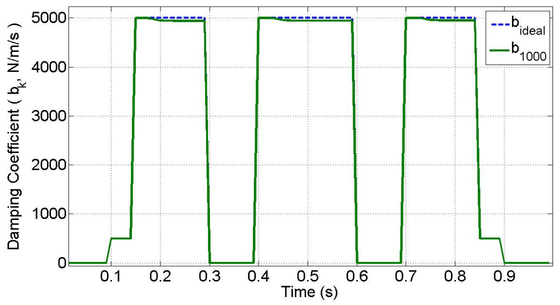

Fig. 9. PLOT OF DAMPING COEFFICIENT ACROSS THE ITERATION FOR ITERATION 1000. THE DOTTED LINE SHOWS THE *IDEAL* DAMPING COEFFICIENT PROFILE.

In order to evaluate the robustness of the method, the algorithm was tested through Monte Carlo simulations of parameter uncertainty of 30% in the mass and spring constants of the actual and nominal models. The sleeve of error decay over iterations for these Monte Carlo simulations is shown in Figure 10. Therefore, we note that the proposed algorithm is robust to *parametric uncertainty* in the model as long as the stability criterion is satisfied.

V. CONCLUSIONS AND FUTURE WORK

This paper investigated semi-active ILC algorithms. The semi-active learning law is different from traditional ILC laws in that it iteratively adjusts *system parameters* instead of control signals in a repetitive process. The parameters were adjusted from one cycle to the next for minimization of a cost function such as the 2-norm of tracking error. The stability criteria for the semi-active ILC update law was obtained based on a gradient-type search. The gradient-based law guarantees convergence to the local minimum of the cost function. With suitable step size monotonic decrease in the cost function is achieved. In order to illustrate an application of the semi-active learning law, a case study on plastic blow molding was presented. P-type and inverse-type semi-active learning laws were shown to provide performance enhancement for this system. The learning laws yielded stable convergence and were shown to be robust to parametric model uncertainty. The P-type learning law, while requiring less model information, resulted in non-monotonic convergence. In contrast, the inverse-type law was monotonically stable, but required model parameter knowledge. The choice of the learning law is therefore based on the availability of accurate model information.

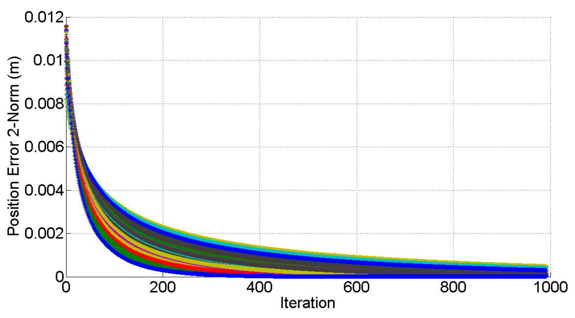

Fig. 10. SLEEVE OF ERROR NORM DECAY PROFILES FOR 1000 MONTE CARLO SIMULATIONS WITH 30% MODEL-PLANT PARAMETER MISMATCH (MASS AND SPRING CONSTANT).

REFERENCES

[1] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *J. of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.
[2] N. McClamroch and H. Gavin, "Electrorheological dampers and semi-active structural control," *Proc. of the 34rd Conf. On Decision and Control*, pp. 3528–3533, Dec. 1995.
[3] B. Spencer, S. Dyke, M. Sain, and J. Carlson, "Phenomenological models for magnetorheological dampers," *Journal of Engineering Mechanics*, vol. 123, no. 3, pp. 230–238, March 1997.
[4] H. Tsend and J. Hedrick, "Semi-active control laws: Optimal and sub-optimal," *Journal of Vehicle System Dynamics*, vol. 23, pp. 545–569, 1994.
[5] M. Norrlof and S. Gunnarsson, "Time and frequency domain convergence properties in iterative learning control," *International Journal of Control*, vol. 75, pp. 1114–1126, 2002.
[6] B. Dijkstra and O. Bosgra, "Extrapolation of optimal lifted system ILC solution, with application to a waferstage," *American Control Conference*, 2002.
[7] EngineersHandbook, "http://www.engineershandbook.com/mfgmethods," 2004.
[8] H. Havlicsek and A. Alleyne, "Nonlinear control of an electro- hydraulic injection molding machine via iterative adaptive learning," *IEEE/ASME Trans. on Mechatronics*, vol. 4, no. 3, p. 312323, 1999.
[9] Moog, *Private Communication with Moog Inc.*, 1998.