ADCHEM 2006

International Symposium on Advanced Control of Chemical Processes
Gramado, Brazil – April 2-5, 2006

ITERATIVE LEARNING CONTROL APPLIED TO BATCH PROCESSES: AN OVERVIEW

Jay H. Lee * **Kwang S. Lee** **

* *School of Chem. and Biomolec. Eng., Georgia Institute of Technology*
311 Ferst Dr. NW, Atlanta, GA 30332-0100, USA
** *Dept. of Chem. and Biomolec. Eng., Sogang University*
1 Shinsoodong, Mapogu, Seoul 121-742, Korea

Abstract: With the recent emphasis on batch processing by the emerging industries like the microelectronics and biotechnology, the interest in batch process control has been renewed. In this paper, we present an overview of the Iterative Learning Control (ILC) technique, which can be used to improve tracking control performance in batch processes. We present the fundamental concepts and review the various ILC algorithms, with a particular focus on a model-based algorithm called Q-ILC and an application involving a Rapid Thermal Processing (RTP) system. The study indicates that one can solve a seemingly very difficult multivariable nonlinear tracking problem with relative ease by combining the ILC technique with basic process insights and standard system identification techniques. We also bring forth some related techniques in the literature with the hope of unifying them and also suggest some remaining challenges.

Keywords: tracking control, iterative learning control, model based control, batch process control

1. INTRODUCTION

Batch processes have historically lagged continuous processes in terms of development and deployment of advanced optimization and control tools. Whereas significant developments have occurred during the past few decades in the industrial practice of continuous process control (Qin and Badgwell, 1996; Morari and Lee, 1999), the same has not been the case for batch processes, which have continued to rely on old techniques like ladder-logics and PID control. Part of this can be attributed to the comparatively lower production volume through batch processing. Another reason for this may be that batch processes present a set of challenges uncommon in continuous processes, including nonstationary operating recipes, the consequent exposure to process nonlinearity, and significant variations in the initial charge condition (Berber, 1996). These challenges are not easily met by the standard linear optimal control theories and tools, which are widely adopted for continuous industrial process control today.

However, the role of batch processing is ever-increasing in today's diversified manufacturing environment. Besides the fine or specialty chemicals, new industries that have emerged from the VLSI technology, bio-technology, and material science are mostly batch-processing-oriented. In accordance with its increased importance, its operation support tools need to be upgraded. Such a shift in the trend has already started taking place, as evidenced by the extensive use of run-to-run control and multivariate monitoring in some of the new industries. We believe, however, that much more can be done, even with the existing technologies today. For example, Iterative Learning Control

(ILC), the topic of this paper, has not enjoyed a serious look by the practitioners thus far despite its vast potentials for improving tracking control performance in batch processes.

This paper presents an overview of ILC in the context of trajectory tracking problems in batch processes. Although basic theories of ILC have been firmly laid out in the literature, it is not always straightforward to apply them to achieve success in practice. We present ILC in the context of a multiple point temperature tracking problem in a Rapid Thermal Processing (RTP) system. By doing so, our objective is to bring forth the unique capabilities of ILC for batch process control and at the same time some of the subtle challenges one may face in applying the technique. Fortunately, such challenges are not insurmountable and the standard linear ILC technique can provide an excellent performance for what appears to be a very difficult nonlinear trajectory tracking problem. We also point out some related techniques like Run-to-Run Control and Repetitive Control, highlighting the similarities and differences. Finally, we point to some of the open issues left for future research.

2. EXEMPLARY PROBLEM: TEMPERATURE TRACKING CONTROL FOR A RTP SYSTEM

In the operation of Rapid Thermal Processing (RTP) systems, one of the most important challenges is to achieve uniform temperature distribution across the wafer surface while tracking a reference trajectory with a temperature range of several hundred degrees. From the system theoretic viewpoint, it is a nonlinear, multivariable control problem involving a batch system with fast dynamics and noisy measurements. Added to this are the facts that a single RTP system can be used for different wafer fabrications demanding different temperature trajectories to be followed and the characteristics of a RTP system may vary significantly by reasons like contamination. All these factors combine to make reliable modeling of the system very difficult (Cho and Gyugyi, 1997).

In the *experimental* RTP equipment, the silicon wafer is heated by an array of 38 bar-type tungsten-halogen lamps, the maximum power of which is 1 Kw each. The lamps are assembled together by two, four or six to comprise a total of ten independent groups, as shown in the figure. The chamber wall is cooled with circulating cooling water. The electric power inputs to the ten groups, denoted by $u_1, \cdots, u_{10}$, are therefore the manipulated inputs. Wafer temperature was measured at eight points with K-type thermocouples (TC's) glued on the backside of the wafer surface. As a consequence, the experimental RTP equipment is configured as an 8×10 MIMO system. In the commercial RTP operation, however, such a wafer with embedded thermocouples would be available only for testing purposes. In the actual production runs, in-situ temperature measurements would have to be provided by pyrometers, and for economic reasons, the number of such sensors per equipment may be limited. Hence, in addition to the full 8×10 system, we investigate the possibility of limiting the temperature measurements to just three locations. In this case, selection of the measurement points becomes an important issue.

Radiative heat transfer equations can be used to construct a fundamental or semi-empirical model representing heat balances. Due to the space limitation, we refer the readers to the open literature for the details of such models (Lee *et al.*, 2001*b*; Lee *et al.*, 2003). Given the idiosyncratic designs of individual equipments, however, it is more realistic to try to develop a control model from system identification, which is the approach we adopt here.

3. ITERATIVE LEARNING CONTROL

ILC is a general technique for improving transient tracking performance of a system that executes a same operation repeatedly. In its basic formulation, a target system has the following characteristics: i) Each run lasts for a fixed length of time; ii) the reference trajectories (to be followed by the outputs) remain the same from run to run; iii) the process state is reset to a same value at the start of each operation. ILC techniques developed under such assumptions can be used effectively on a process with some disturbances and initialization errors as well as occasional changes in the reference trajectories, however. Temperature tracking problems in many chemical batch processes can be tailored to fit into this category and hence the relevance to batch process operations.

3.1 *Historical Account*

It seems that the first technical contribution on ILC was the patent work by Garden (1971) three decades ago. Although a few independent contributions followed after that (Miller and G. T. Mallick, 1978; Uchiyama, 1978), it seems to have gone unnoticed by the larger control community until Arimoto *et al.* (1984) proposed the so-called D-type learning algorithm as a teaching mechanism for robot manipulators. This seminal work launched ILC into the mainstream control community and established it as a new branch of control technology. Significant body of work followed after that, which we summarize below.

However, we do not claim the list of references to be complete or unbiased.

3.2 *Basic Formulation*

Let $\mathbf{e}_k = \mathbf{r} - \mathbf{y}_k$ and $\mathbf{u}_k$ represent the output error trajectory and the manipulated input trajectory for the k^{th} run. For sampled **data** systems, these trajectories can be represented by finite dimensional vectors as follows:

$$\begin{aligned} \mathbf{e}_k &\triangleq \left[\, e_k(1)^T \;\; e_k(2)^T \;\; \cdots \;\; e_k(N)^T \,\right]^T \\ \mathbf{u}_k &\triangleq \left[\, u_k(0)^T \;\; u_k(1)^T \;\; \cdots \;\; u_k(N-1)^T \,\right]^T \end{aligned} \tag{1}$$

where N represents the number of sample points in each run. Note that the sampling interval do not need to be same for the inputs and the outputs or even uniform throughout the batch interval. This assumption is made here just for the convenience of exposition.

The objective in the ILC design can be simply stated as:

$$\|\mathbf{e}_k\| \to \mathbf{0} \;\; \text{as} \;\; \mathrm{k} \to \infty \tag{2}$$

A basic rule for updating the input trajectory on a run-to-run basis is the so-called first-order learning algorithm, which is represented by

$$\mathbf{u}_k = \mathbf{u}_{k-1} + \mathbf{H}\mathbf{e}_{k-1} \tag{3}$$

Here, $\mathbf{H}$ is called the learning filter matrix, which in general can be any map that transforms the finite-length error trajectory to a trajectory whose length and dimension are equal to those of the input trajectory. The learning filter can be designed as a dynamic filter, $H(s)$ or $H(z)$, operating on the time signal $e(t)$, depending on the underlying time domain for the system representation. Since $\mathbf{H}$ operates on the error trajectory of the previous run, it is *not* limited to *causal* maps, however. This is what gives ILC the distinct ability to overcome hindrances from dynamic elements like time delays to provide perfect tracking.

Note that the above learning algorithm has an integral action over the run index k. Hence, one can intuitively argue that (2) can be fulfilled with an appropriately chosen $\mathbf{H}$, just as the integral action in a PID controller can remove offset in the time domain. Since batch processes have no dynamics carried over from one run to next, pure integral control such as in (3) is sufficient in achieving the convergence. Nevertheless, a high-order algorithm like

$$\mathbf{u}_k = \mathbf{u}_{k-1} + \mathbf{H}_1\mathbf{e}_{k-1} + \cdots + \mathbf{H}_p\mathbf{e}_{k-p} \tag{4}$$

has also been studied as a generalization of (3) (Bien and Huh, 1989). When all the states of a batch process are reset to same values at the start of each run and disturbances do not vary from batch to batch, there is no benefit to be gained from the high-order generalization. However, when errors do not carry over completely from one batch run to next due to run-specific disturbances, measurement noises, and model errors, the high-order algorithm can deliver a superior performance owing to its ability to filter the error trajectories by using the results of several runs.

3.3 *Model-Based Formulation*

With the above form of the learning algorithm, the problem of ILC design is reduced to the design of the learning filter. In the initial period of development, researchers focused on model-free approaches, where a certain generic structure is presupposed on $\mathbf{H}$ and the parameters are tuned to achieve the convergence. D-type(Arimoto *et al.*, 1984) and PID-type(Bondi *et al.*, 1988) algorithms are such examples.

Alternatively, model-based algorithms were introduced in order to address more complex problems (e.g., MIMO systems) and to bring more insights into the technique. Early approaches were based on direct model inversion, *i.e.*, $\mathbf{H} = \mathbf{G}^{-1}$, and its variants(Togai and Yamano, 1985; Oh *et al.*, 1988; Lucibello, 1992; Moore, 1993; Lee *et al.*, 1994), where $\mathbf{G}$ represents the input-output map of the concerned process (*i.e.*, $\mathbf{y}_k = \mathbf{G}\mathbf{u}_k$). Note that $\mathbf{G}$ contains information about the batch process *dynamics* and is similar to the concept of *dynamic matrix* used in model predictive control (MPC). Though we are using a linear map here, note that the underlying dynamics are not limited to be linear time-invariant; time-varying dynamics (approximating nonlinear dynamics for instance) may be easily incorporated into the $\mathbf{G}$ matrix. In the case that $\mathbf{G}$ is exactly known, this particular choice of $\mathbf{H}$ eliminates the error completely after one iteration, which can be easily verified by multiplying $\mathbf{G}$ on both sides of (3). Nonminimum-phase dynamics do not cause any problem in the inversion here as $\mathbf{H}$ is not restricted to be causal. In practice, however, the inverse-model-based learning filter can give many problems. For a typical over-damped system, of which the inverse has increasingly higher gains with the frequency, the filter can be very sensitive to high frequency components of $e_k(t)$ producing extremely spiky input profiles. Also, since high frequency dynamics typically carry large model errors, the high filter gains in high frequency region can cause divergence.

Furthermore, the objective in (2) cannot always be satisfied for general MIMO systems. When the number of output variables is larger than that of input variables or when constraints become active, the error may not be made zero in general. An alternative objective may be to try to converge to an input trajectory that minimizes the output error:

$$\|\mathbf{e}_k\| \rightarrow \min_{\mathbf{u}} \|\mathbf{e}\| \quad \text{as} \quad \mathrm{k} \rightarrow \infty \tag{5}$$

where $\|\cdot\|$ is some vector norm. The 2-norm is the typical choice.

Moore (1993) proposed to solve

$$\min_{\mathbf{u}_k} \|\mathbf{e}_k\|^2 \tag{6}$$

before the start of the k^{th} run, based on the error from the $k-1^{\text{th}}$ run. For a linear system, $\mathbf{y} = \mathbf{G}\mathbf{u}$, the error model can be written as

$$\mathbf{e}_k = \mathbf{e}_{k-1} - \mathbf{G}(\mathbf{u}_k - \mathbf{u}_{k-1}) \tag{7}$$

The least squares solution is

$$\mathbf{u}_k = \mathbf{u}_{k-1} + \mathbf{G}^{+}\mathbf{e}_{k-1} \tag{8}$$

where the superscript $^+$ represents the pseudo-inverse. Alternatively, Amann *et al.* (1996) and Lee *et al.* (1996) independently proposed to solve

$$\min_{\Delta\mathbf{u}_k} \left\{\|\mathbf{e}_k\|^2_{\mathbf{Q}} + \|\Delta\mathbf{u}_k\|^2_{\mathbf{R}}\right\} \tag{9}$$

In the above, $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$ and the notation of $\|\mathbf{x}\|^2_{\mathbf{P}}$ denotes $\mathbf{x}^T\mathbf{P}\mathbf{x}$. The resulting input for the linear system, $\mathbf{y} = \mathbf{G}\mathbf{u}$, is given as

$$\mathbf{u}_k = \mathbf{u}_{k-1} + (\mathbf{G}^T\mathbf{Q}\mathbf{G} + \mathbf{R})^{-1}\mathbf{G}^T\mathbf{Q}\mathbf{e}_{k-1} \tag{10}$$

As $k \rightarrow \infty$, input profiles that result from (8) and (10) converge to the same limit $\mathbf{u}^*$ that satisfies $\|\mathbf{e}(\mathbf{u}^*)\| = \min_{\mathbf{u}} \|\mathbf{e}\|$. What happens with (10) is that the convergence is retarded by the input change penalty term. In particular, high-frequency type changes in the input profile, which tend to be large due to low system gains, are penalized heavily. In the course of learning, therefore, much smoother input profiles that drive the error to a near-minimum (*i.e.*, minimum except for the high frequency components for which the learning gains are much too low compared to the system gains due to the input penalty term) are obtained by (10). In practice, the learning can be stopped before the input profiles become very spiky or divergence behavior starts setting in. For simplicity, we call the algorithm based on (9) Q-ILC (Quadratic criterion-based ILC) hereafter.

Later, Lee *et al.* (2000) considered a batch process subject to stochastic disturbances and proposed an observer-based Q-ILC algorithm. This extension provided a more convenient and intuitive tuning knob to control the rate of convergence and the ability to take a more systematic account of disturbances and noises. A robustness study of Q-ILC indicated that convergence can be achieved in the presence of a fairly large model error (Lee *et al.*, 2000; Kim *et al.*, 2000). They also pointed out that, when the constraints are given as linear inequalities, the optimal input profile respecting the constraints can be obtained through a quadratic programming technique. Convergence of the constrained algorithm with an observer was proved in Lee and Lee (2000).

4. PROTOTYPICAL MODEL-BASED ILC TECHNIQUE

There are several similar versions of Q-ILC (Amann *et al.*, 1996; Lee *et al.*, 2000; Lee *et al.*, 1999; Chin *et al.*, 2004). The basic idea for all these versions is the same, but they differ in the way the filtering of error trajectories (or detuning of the learning gain) is accomplished, and whether and how real-time feedback control (RFC) signal is added to the feedforward ILC signal. Here we present a particular version of Q-ILC, which is based on the work by Chin *et al.* (2004) and was found to be particularly suited to the RTP application.

A general form of ILC combined with RFC is

$$\begin{aligned} u_k(t) = \; & u_{k-1}(t) + H_1(t)e_{k-1}(1:N) \\ & + H_2(t)e_k(1:t) \end{aligned} \tag{11}$$

where $(i:j)$ means data from $t = i$ to j and H_1 and H_2 represent the gains for the ILC and RFC, respectively. The role of RFC is to further modify $u_k(t)$ based on the real-time error feedback of $e_k(1:t)$, which contains information on a new disturbance occurring during the on-going k^{th} run. However, under this scenario, the updated input for the next run would also be affected by the new disturbance, which may not show up in the next run. The following technique attempts to separate $u(t)$ into the ILC- and RFC-related terms so that disturbances specific to a current run would have minimal effect on the next run.

4.1 *Model Formulation*

Suppose that the system dynamics are represented by

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + Kv(t) \\ y(t) &= Cx(t) + v(t) \end{aligned} \tag{12}$$

In (12), $v(t)$ is a zero-mean independent, identically distributed (i.i.d.) sequence in time but $v_k(t)$

for different k's may show correlation. In fact, $v_k(t)$ may even exhibit drifting behavior along k and not just random fluctuations around a stationary mean. Such behavior can be reasonably described by the equation

$$\begin{aligned} \bar{v}_k &= \bar{v}_{k-1}(t) + n_k(t) \\ v_k(t) &= \bar{v}_k(t) + \hat{v}_k(t) \end{aligned} \tag{13}$$

where $\hat{v}_k(t)$ and $n_k(t)$ are zero-mean i.i.d. sequences with respect to both k and t. Such a model can be obtained through system identification techniques like N4SID (Overschee and Moor, 1994), as will be demonstrated later.

Now, using the superposition principle, we decompose $u_k(t)$ into $\bar{u}_k(t)$ (the ILC input) and $\hat{u}_k(t)$ (the RFC input), and also separate (12) into two parts, one that is driven by $\bar{u}_k(t)$ and $\bar{v}_k(t)$, and the other by $\hat{u}_k(t)$ and $\hat{v}_k(t)$ as follows:

$$\begin{aligned} \bar{x}_k(t+1) &= A\bar{x}_k(t) + B\Delta\bar{u}_k(t) + Kn_k(t) \\ \bar{y}_k(t) &= C\bar{x}_k(t) + \bar{y}_{k-1}(t) + n_k(t) \end{aligned} \tag{14}$$

$$\begin{aligned} \hat{x}_k(t+1) &= A\hat{x}_k(t) + B\hat{u}_k(t) + K\hat{v}_k(t) \\ \hat{y}_k(t) &= C\hat{x}_k(t) + \hat{v}_k(t) \end{aligned} \tag{15}$$

where $\Delta\bar{u}_k \triangleq \bar{u}_k - \bar{u}_{k-1}$. Naturally, we have

$$y_k(t) = \bar{y}_k(t) + \hat{y}_k(t) \tag{16}$$

4.2 *Algorithm*

The Q-ILC algorithm follows the following steps:

(1) *Information Gathering*: After the $k-1^{\text{th}}$ run, $\{e_{k-1}(t),\ \bar{u}_{k-1}(t), \hat{u}_{k-1}(t)\}$ are available.
(2) *ILC Signal Computation*: Before starting the k^{th} run, compute $\bar{u}_k(t),\ t \in [0, \cdots, N-1]$ off-line according to a chosen cost function involving $\bar{e}_{k|k-1}(t)$.
(3) *RFC Signal Computation*: At each t during the k^{th} run, calculate $\hat{u}_k(t)$ such that the predicted error for the current run is minimized. Then, apply $u_k(t) = \bar{u}_k(t) + \hat{u}_k(t)$ to the process.

4.2.1. *Details of the ILC Signal Computation*

Define

$$\begin{aligned} \bar{\mathbf{e}} &\triangleq \left[\, \bar{e}(1)^T\ \bar{e}(2)^T\ \cdots\ \bar{e}(N)^T \,\right]^T \\ \Delta\bar{\mathbf{u}} &\triangleq \left[\, \Delta\bar{u}(0)^T\ \Delta\bar{u}(1)^T\ \cdots\ \Delta\bar{u}(N-1)^T \,\right]^T \end{aligned} \tag{17}$$

and similarly for other variables where $\bar{e} \triangleq r - \bar{y}$. Expanding equations (14) and (15) gives

$$\begin{aligned} \bar{\mathbf{e}}_k &= \bar{\mathbf{e}}_{k-1} - \mathbf{G}\Delta\bar{\mathbf{u}}_k + \mathbf{w}_k \\ \mathbf{e}_k &= \bar{\mathbf{e}}_k - \mathbf{G}\hat{\mathbf{u}}_k + \mathbf{m}_k \end{aligned} \tag{18}$$

where

$$\mathbf{G} \triangleq \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B & CA^{N-2}B & \cdots & CB \end{bmatrix} \tag{19}$$

$\mathbf{w}_{k+1}$ and $\mathbf{m}_k$ are appropriately defined signals that are independent in k.

$\Delta\bar{\mathbf{u}}_k$ is determined according to

$$\min_{\Delta\bar{\mathbf{u}}_k}\ \frac{1}{2}\left\{\|\bar{\mathbf{e}}_{k|k-1}\|^2_{\mathbf{Q}} + \|\Delta\bar{\mathbf{u}}_k\|^2_{\mathbf{R}}\right\} \tag{20}$$

where $\bar{\mathbf{e}}_{k|k-1}$ is the optimal prediction of $\bar{\mathbf{e}}_k$ based on the information available at the start of the k^{th} run, which is given by the Kalman filter applied to (18). The unconstrained solution to (20) is

$$\begin{aligned} \Delta\bar{\mathbf{u}}_k &= \mathbf{H}\bar{\mathbf{e}}_{k-1|k-1} \\ &= (\mathbf{G}^T\mathbf{Q}\mathbf{G} + \mathbf{R})^{-1}\mathbf{G}^T\mathbf{Q}\bar{\mathbf{e}}_{k-1|k-1} \end{aligned} \tag{21}$$

The Kalman filter estimates $\bar{\mathbf{e}}_{k-1|k-1}$ is computed as

$$\begin{aligned} \bar{\mathbf{e}}_{k-1|k-1} &= \bar{\mathbf{e}}_{k-1|k-2} + \mathbf{K}(\mathbf{e}_{k-1} + \mathbf{G}\hat{\mathbf{u}}_{k-1} - \bar{\mathbf{e}}_{k-1|k-2}) \\ \bar{\mathbf{e}}_{k|k-1} &= \bar{\mathbf{e}}_{k-1|k-1} - \mathbf{G}\Delta\bar{\mathbf{u}}_k \end{aligned} \tag{22}$$

where $\mathbf{K}$ is the optimal gain matrix computed using the model of (18).

4.2.2. *Details of RFC Signal Calculation*

For the calculation of RFC signal, it is tempting to think that one should use (15) to control just the $\hat{y}$ component of y. However, it turns out to be beneficial to try to reduce the entire error in y as it helps to achieve faster convergence. It is convenient to consider a model with the input $\Delta\bar{u}_k(t)$ term taken out from the model. For this, we consider the following deterministic model that represents the output by $\Delta\bar{u}_k(t)$

$$\begin{aligned} a_k(t+1) &= Aa_k(t) + B\Delta\bar{u}_k(t) \\ y_{a,k}(t) &= Ca_k(t) \end{aligned} \tag{23}$$

Subtracting (23) from (14) eliminates $\Delta\bar{u}_k(t)$ from the equation:

$$\begin{aligned} \bar{x}_{a,k}(t+1) &= A\bar{x}_{a,k}(t) + Kn_k(t) \\ \bar{y}_k(t) &= C\bar{x}_{a,k}(t) - Ca_k(t) + \bar{y}_{k-1}(t) + n_k(t) \end{aligned} \tag{24}$$

We replace $\bar{y}_{k-1}(t)$ with $\bar{y}_{k-1|k-1}(t)$ which is given by (22). Combining (15) and (24) yields

$$\begin{aligned} &\begin{bmatrix} \hat{x}_k(t+1) \\ \bar{x}_{a,k}(t+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} \hat{x}_k(t) \\ \bar{x}_{a,k}(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \hat{u}_k(t) \\ &\qquad + \begin{bmatrix} K\hat{v}_k(t) \\ Kn_k(t) \end{bmatrix} \\ &y_k(t) + Ca_k(t) - \bar{y}_{k-1|k-1}(t) \\ &\qquad = \begin{bmatrix} C & C \end{bmatrix} \begin{bmatrix} \hat{x}_k(t) \\ \bar{x}_{a,k}(t) \end{bmatrix} + \hat{v}_k(t) + n_k(t) \end{aligned} \tag{25}$$

Denote the above as

$$z_k(t+1) = \Phi z_k(t) + \Gamma \hat{u}_k(t) + \zeta_k(t) \quad (26)$$
$$y_k(t) + Ca_k(t) - \bar{y}_{k-1|k-1}(t) = \Sigma z_k(t) + \eta_k(t)$$

Then, $\hat{u}_k(t)$ is determined based on the quadratic criterion

$$\min_{\hat{u}_k(\cdot)} E\Big\{ \|e_k(N)\|_M^2 + \sum_{t=0}^{N-1} \|e_k(t)\|_Q^2 + \|\hat{u}_k(t)\|_R^2 \Big\}$$
$$\text{subject to (26)} \quad (27)$$

Enforcing $e_k(t) \rightarrow 0$ is equivalent to steering $y_k(t) + Ca_k(t) - \bar{y}_{k-1|k-1}(t)$ to $r(t) + Ca_k(t) - \bar{y}_{k-1|k-1}(t)$. (27) is a standard LQ servo problem for the output of (26) to follow $r(t) + Ca_k(t) - \bar{y}_{k-1|k-1}(t)$. The solution is standard and given in the following form:

$$\begin{aligned} & \hat{u}_k(t) = -L_{\text{fb}}(t) z_k(t|t) + L_{\text{ff}}(t) b_k(t) \\ \rightarrow \quad & u_k(t) = \bar{u}_k(t) + \hat{u}_k(t) \end{aligned} \quad (28)$$

Detailed forms of $L_{\text{fb}}(t)$, $L_{\text{ff}}(t)$ and $b_k(t)$ can be found in textbooks like (Lewis and Syrmos, 1995), or in (Lee *et al.*, 2001*b*).

5. RESULTS OF APPLICATION TO AN EXPERIMENTAL RTP SYSTEM

The previously described Q-ILC algorithm was applied to the experimental RTP system introduced earlier. The length for each experimental run was 40 seconds and the sampling time was chosen as 0.5 second. The reference temperature trajectory was comprised of a holding zone at 400°C for 7 seconds and a ramping zone with 30°C/sec followed by another holding zone at 700°C for 25 seconds. Control of the wafer temperature for the first run was carried out using the regular MPC and then Q-ILC was applied from the 2nd run on.

5.1 *Model Identification*

The identification experiments were conducted while maintaining the wafer temperatures at around 650°C. Independent PRBSs were simultaneously added for 2,000 seconds to the respective steady state input values of the 10 lamp groups and the resulting temperature responses at the eight locations were taken. Suitable choice for the minimum clock period of the PRBS signals was found to be 15 seconds.

Examining the radiative heat transfer terms leads us to consider that an RTP system would be better represented by a linear dynamic model that relates $u(t)$ to $T_w^4(t)$, rather to $T_w(t)$ as previously done in (Lee *et al.*, 2001*b*) and (Lee *et al.*, 2003). Here T_w represents a vector containing the wafer temperatures at the eight locations. A linear dynamic model between $u(t)$ and $y(t) \triangleq T_w^4(t)$ can be obtained using a standard identification method, *e.g.*, N4SID (Overschee and Moor, 1994). To remove the bias effect ('trend'), we pretreated the input and output data using a difference filter of $F(q^{-1}) = \frac{1-q^{-1}}{1-fq^{-1}}$, $0 \le f < 1$. We chose $f = 0.974$. Let the filtered variables be

$$u_f(t) \triangleq F(q^{-1})u(t), \quad y_f(t) \triangleq F(q^{-1})y(t) \quad (29)$$

Processing $\{u_f(t)\}$ and $\{y_f(t)\}$ using N4SID yields a linear stochastic state space model in the following innovation form:

$$\begin{aligned} x_f^r(t+1) &= \bar{A}x_f^r(t) + \bar{B}u_f(t) + \bar{K}v(t) \quad (30) \\ y_f(t) &= \bar{C}x_f^r(t) + v(t) \end{aligned}$$

where $\{v(t)\}$ is a zero-mean i.i.d. sequence referred to as the *innovations*. (30) can be rewritten in terms of the original input and output as follows:

$$\begin{bmatrix} x^r(t+1) \\ x^d(t+1) \end{bmatrix} = \begin{bmatrix} \bar{A} & \bar{K} \\ 0 & I \end{bmatrix} \begin{bmatrix} x^r(t) \\ x^d(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} \bar{K} \\ (1-f)I \end{bmatrix} v(t)$$
$$y(t) = \begin{bmatrix} \bar{C} & I \end{bmatrix} \begin{bmatrix} x^r(t) \\ x^d(t) \end{bmatrix} + v(t) \quad (31)$$

Note that this model is in the form of (12), which was the basis for the Q-ILC algorithm development.

5.2 *Results and Discussion*

The result of applying the Q-ILC algorithm to the 8×10 system with T_w^4 as the output is shown in Table 2. The tracking error is continuously decreased with the run number and converges approximately after 7 runs. The temperature gap given in the table is defined as the maximum difference among the eight temperatures. This was reduced down to 5°C in the constant temperature region.

One of the main feature of the observer based Q-ILC algorithm is that batch-specific disturbances are filtered out to have minimal effect on the learning for subsequent runs. To demonstrate this, we lowered the initial wafer temperature to 370°C in the 8^{th} run and then returned it to the nominal value in the following run. From the results in Table 1, we can see that the performance of the 7th run is almost completely restored in the 9th run. Though not shown, it was observed that the ILC input for the 9^{th} run was hardly affected by the disturbance that occurred during the 8^{th} run.

In Table 2, the temperature from T_w^4 model-based control is compared with that of the conventional

linear model-based control where $y(t) \triangleq T_w(t)$. We can see that the temperature uniformity could be remarkably improved by the use of T_w^4 as the output of the control model. Note that the temperature gap was reduced significantly on the average. This shows that such simple process knowledge can be a huge factor if used appropriately.

We also attempted to limit the measurements to three locations, leading to a 3×10 system. The three measurement locations were decided to minimize the estimation error covariance. For comparison, we show in Table 3. the results when only the three temperatures were controlled. The three controlled points could achieve excellent tracking and regulation, but the temperature gap among the eight points, which were just monitored, was observed to be quite large. As an alternative, we decided to estimate the temperatures of the remaining five points based on the three measured temperatures. Linear estimation was employed for simplicity. The estimated temperatures along with the measured ones were used as if they were all measured, as in the 8×10 system case. Performance similar to that for the full measurement case could be achieved, as can be seen from Table 4.

6. RELATED TECHNIQUES

There are several techniques in the literature that share strong similarities with ILC. For example, repetitive control is a technique applied to a continuous system with periodic characteristics, which are also seen in repeated batch runs and thus form a basis for ILC. Given such strong similarities, cross-breeding of the techniques from these fields should be possible. Here we briefly review the related techniques and discuss whether they provide any new insight into ILC and vice versa.

6.1 *Run-to-Run Control*

Run-to-run control is a popular method to control product qualities in processes where direct in-situ measurements of the quality variables are impractical and off-line product analysis results must be utilized instead. Such processes are common in semi-conductor and polymer manufacturing. The basic form of linear model used for run-to-run control is

$$q_k = Mp_k + \frac{1}{1-q^{-1}} v_k \tag{32}$$

where q_k is the vector containing the end quality variables, p_k is the vector containing the recipe parameters, and v_k is an i.i.d. sequence. Here, the disturbance is modeled as an integrated white noise to account for jumps and slow drifts. Other disturbance models such as $\frac{1-\alpha q^{-1}}{1-q^{-1}}$ or double integrators can be used if deemed more appropriate.

The optimal prediction model for (32) is expressed as

$$q_{k|k-1} = q_{k-1} + M\Delta p_k + v_k \tag{33}$$

and Δp_k representing the recipe adjustment can be computed by minimizing $\|r - q_k\|_P^2$ or $\|r - q_k\|_P^2 + \|\Delta p_k\|_Q^2$ as before where r represents the desired quality values. Of course, nonlinear recipe models can be used, which may or may not be combined with a nonlinear state observer.

Examining the above, we see that there is little difference between ILC and run-to-run control except that the former addresses trajectory tracking problems whereas the latter addresses end quality control problems. This similarity was noticed by Chin *et al.* (2000), who combined ILC and run-to-run control concepts into an integrated technique called QBMPC intended for simultaneous trajectory tracking and end quality control. An added feature of QBMPC was that the end quality variables for an on-going batch run could be inferred from the on-line process measurements, thereby giving the controller the capability to make more immediate adjustment with respect to disturbances.

6.2 *Batch-to-Batch Optimization*

Batch-to-batch optimization (BBO) refers to the use of nonlinear programming technique on a real batch process. The key is to evaluate the objective function with actual process measurements rather than a model. The idea was once called EVOP (EVolutionary OPtimization) (Wilde and Beightler, 1967) and popular in the 60's in optimizing operating conditions in the process industries. The technique was recently brought back to the attention of the process control community by a number of researchers, mostly in the context of optimizing semiconductor manufacturing processes(Zafiriou and Zhu, 1990; Zafiriou *et al.*, 1995).

Consider an optimization problem

$$\min_{\mathbf{u}} \quad \phi(\mathbf{y}, \mathbf{u}) \tag{34}$$

Suppose the batch process's dynamics are represented by the nonlinear input/output map, $\mathbf{y} = \mathcal{N}(\mathbf{u}, \mathbf{d})$, where $\mathbf{d}$ is a disturbance trajectory. Based on the model, the optimization may be recast as

$$\min_{\mathbf{u}} \; J(\mathbf{u}) \; (= \phi(\mathbf{y}, \mathbf{u})) \tag{35}$$

In the above, J can be evaluated through ϕ where $\mathbf{y}$ is provided by the prediction model

$$\mathbf{y}_{k+1|k} = \mathbf{y}_k + \left(\frac{\partial \mathcal{N}}{\partial \mathbf{u}}\right)_k \Delta \mathbf{u}_{k+1} \tag{36}$$

Note that the actual process measurements of $\mathbf{y}_k$ is used, which provides some robustness when the dynamic map $\mathcal{N}$ and the disturbance $\mathbf{d}$ are not perfectly known.

A general search algorithm for $\mathbf{u}$ can be written as

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{g}_k \tag{37}$$

where $\mathbf{g}$ represents a search direction. From

$$\begin{aligned} J(\mathbf{u}_{k+1}) &\approx J(\mathbf{u}_k) + \nabla J(\mathbf{u}_k)^T (\mathbf{u}_{k+1} - \mathbf{u}_k) \\ &= J(\mathbf{u}_k) + \alpha_k \nabla J(\mathbf{u}_k)^T \mathbf{g}_k \end{aligned} \tag{38}$$

it can be seen that $J(\mathbf{u}_{k+1}) < J(\mathbf{u}_k)$ for a sufficiently small α_k as long as $\mathbf{g}_k$ is given such that $\nabla J(\mathbf{u}_k)^T \mathbf{g}_k < 0$. In the steepest descent algorithm, $\mathbf{g}_k$ is given by $-\nabla \tilde{J}(\mathbf{u}_k)$, where $\tilde{J}$ represents ϕ evaluated with the prediction model (36). Alternatively, one can also solve for $\Delta \mathbf{u}$ minimizing the objective function ϕ. Note that a significant model error can be allowed in J (or equivalently in $\mathcal{N}$) before the convergence is violated.

From the use of (36), it is easy to see that the above described BBO is very closely related to ILC. In fact, the standard ILC can be interpreted as a special case of BBO where the objective function $\phi(\mathbf{y}, \mathbf{u})$ is quadratic in the tracking error and the input change and the underlying dynamic input-output map is linear. Hence, the BBO approach provides a natural extension of the standard ILC to the case of nonlinear model.

In Bonvin *et al.* (2001), BBO methods are classified into four categories according to how the model information is used. In their paper, the term BBO is used in a wider sense to include a model-based method with recursive identification, and reference-based and data-based model-free methods. The BBO method described in the above belongs to the (uncertain) fixed-model-based method according to the classification. In addition, they proposed a special BBO method called invariant-based optimization where some invariant structure of the input profile is determined first off-line using a coarse process model, the input is parameterized based on the invariant structure, and the true invariants are found using the process measurements.

6.3 *Repetitive Control*

Repetitive control (RC) is a control technique for canceling a periodic disturbance or tracking a periodic reference signal in a continuous process. Repetitive controllers are originally constructed using a delay loop based on the internal model principle(Hara *et al.*, 1988). However, the parallel with the ILC is clear that the error trajectory tends to repeat from cycle to cycle. The main difference is that, unlike in batch systems, the system state is not reset at the beginning. In the ILC literature, such a case is referred by the name of *no-reset ILC*, which is a new branch of study in ILC(Sison and Chong, 1996). In no-reset ILC, transient effect of previous cycles carried over is not taken into account and is simply left to die out with cycles.

Lee *et al.* (2001*a*) noted the similarity between the two problems and proposed an MPC technique called RMPC (Repetitive MPC), which is based on a periodically varying system description and has a strong parallel to Q-ILC. The RMPC technique was successfully applied to the control of a simulated bed chromatography system(Natarajan and Lee, 2000; Erdem *et al.*, 2004). More recently, Lee and Gupta (2005) proposed an extension of this to add robustness to mismatch in the period length. The robustness is achieved by penalizing the input change in a higher order difference form. Though the period mismatch is an important issue for batch processes as well, it is not clear how the idea would extend to the batch system case.

7. CONCLUSIONS

7.1 *Summary*

Iterative Learning Control (ILC) has great potentials for improving tracking control in batch processes. Though initially developed as a heuristic method for improving trajectory tracking performance of robot manipulators, two decades of research has laid solid theoretical foundations and generated insights needed for successful use in general tracking problems in batch processes. In particular, model-based algorithms like Q-ILC can address complex multivariable constrained systems and can be designed for significant robustness to model errors. We demonstrated the potentials and subtle challenges by presenting a case study involving an experimental RTP system. As turned out, one can effectively solve what appeared to be a very difficult multivariable, nonlinear tracking problem by combining the model-based ILC technique with some sound engineering judgment and creativity.

7.2 *Future Research Directions*

An important assumption behind all current ILC algorithms is that the run length is fixed and the

reference trajectory remains same. In many industrial batch processes, however, this assumption is oftentimes violated. Even though each batch run may slightly different, the basic pattern of the trajectory, such as hold-ramp-hold may not change. The main question is how one can translate the error trajectory from previous batch runs into an error trajectory for a new run, which may have a different length and reference trajectory.

Another important issue is more systematic accounting for model errors in the ILC design. In particular, when the model error can be described quantitatively such as polytopic bounds for the dynamic gain matrix, we would like to be able to use such information directly in the design. Because the batch system can be viewed as a simple integrating system along the batch index, derivation of robust ILC algorithms using the usual formulation like min-max optimization may prove to be more tractable. Some initial ideas along this direction can be found in Lee *et al.* (2000).

Finally, the use of a nonlinear model within the existing ILC algorithms has been studied extensively but it is beyond the scope of this paper.

8. REFERENCES

Amann, N., D. H. Owens and E. Rogers (1996). Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proc.-Control Theory and Applications* **143**, 217–224.

Arimoto, S., S. Kawamura and F. Miyazaki (1984). Bettering operation of robots by learning. *J. Robotic Syst.* **1**, 123–140.

Berber, R. (1996). Control of batch reactors: a review. *Trans IChemE.* **74**, 3–20.

Bien, Z. and K. M. Huh (1989). Higher-order iterative learning control algorithm. *IEE Proc.* **136**, 105–112.

Bondi, P., G. Casalino and L. Gambradella (1988). On the iterative learning control theory for robotic manipulators. *IEEE J. Robot Autom.* **4(1)**, 14–22.

Bonvin, D., B. Srinivasan and D. Ruppen (2001). Dynamic optimization in the batch chemical industry. *Proc. of CPC-VI, Tucson, AZ.*

Chin, I., S. J. Qin, K. S. Lee and M. Cho (2004). A two-stage ilc technique combined with real-time feedback for independent disturbance rejection. *Automatica* **40**, 1913–1922.

Chin, I. S., K. S. Lee and J. H. Lee (2000). A technique for integrated quality control, profile control, and constraint handling for batch processes. *Ind. Eng. Chem. Res.* **39**, 693–705.

Cho, Y. M. and P. J. Gyugyi (1997). Control of rapid thermal processing: A system theoretic approach. *IEEE Trans. Contr. Sys. Tech.* **5**, 644.

Erdem, G., S. Abel, M. Morari, M. Mazzotti, M. Morbidelli and J. H. Lee (2004). Automatic control of simulated moving bed. *Ind. Eng. Chem. Res.* **43**, 405–421.

Garden, M. (1971). Learning control of actuators in control systems. *US Patent 3555252.*

Hara, S., Y. Yamamoto, T. Omata and N. Nakano (1988). Repetitive control system: a new type servo system for periodic exogeneous signals. *IEEE Trans. on A.C.* **33**, 659–668.

Kim, W. C., I. S. Chin, K. S. Lee and J. Choi (2000). Analysis and reduced-order design of quadratic criterion-based iterative learning control using singular value decomposition. *Comp. and Chem. Eng.* **24**, 1781–2039.

Lee, J. H. and M. Gupta (2005). Period robust repetitive model predictive control. *Journal of Process Control.*

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

Lee, J. H., S. Natarajan and K. S. Lee (2001*a*). A model-based predictive control approach to repetitive control of continuous processes with periodic operations. *J. Process Control* **11**, 195–207.

Lee, K. S. and J. H. Lee (2000). Convergence of constrained model predictive control for batch processes. *IEEE Trans. on A.C.* **45**, 1928–1932.

Lee, K. S., H. J. Ahn, D. R. Yang and J. H. Lee (2003). Experimental application of a quadratic optimal iterative learning control method for control of wafer temperature uniformity in rapid thermal processing. *IEEE Trans. Semicond. Manuf.* **16**, 36.

Lee, K. S., J. H. Lee, I. S. Chin and H. J. Lee (1999). Model predictive control technique combined with iterative learning control for batch processes. *AIChE J.* **45**, 2175–2187.

Lee, K. S., J. Lee, I. S. Chin, J. Choi and J. H. Lee (2001*b*). Control of wafer temperature uniformity in rapid thermal processing using an optimal iterative learning control technique. *Ind. Eng. Chem. Res.* **40**, 1661.

Lee, K. S., S. H. Bang and K. S. Chang (1994). Feedback-assisted iterative learning control based on an inverse process model. *J. Process Control* **4**, 77–89.

Lee, K. S., W. C. Kim and J. H. Lee (1996). Model-based iterative learning control with quadratic criterion for linear batch processes. *J. Cont. Autom. Sys. Engng.* **2**, 148–157.

Lewis, F. L. and V. L. Syrmos (1995). *Optimal Control.* John Wiley and Sons. New York.

Lucibello, P. (1992). Learning control of linear systems. *Proc. of Amer. Control Conf., Chicago, IL* pp. 1888–1892.

Miller, R. C. and Jr. G. T. Mallick (1978). Method for controlling an automatic machine tool. *US Patent 4088899*.

Moore, K. L. (1993). *Iterative Learning Control for Deterministic System*. Springer-Verlag. New York.

Morari, M. and J. H. Lee (1999). Model predictive control: past, present, and future. *Comp. Chem. Eng.* **23**, 667–682.

Natarajan, S. and J. H. Lee (2000). Repetitive model predictive control applied to a simulated bed chromatography system. *Comp. Chem. Eng.* **24**, 1127–113.

Oh, S. R., Z. Bien and I. H. Suh (1988). An iterative learning control method with application for the robot manipulator. *IEEE J. Robot Autom.* **4(5)**, 508–514.

Overschee, P. Van and B. D. Moor (1994). 4sid: Subspace algorithms for the identification of combined deterministic-stochastic system. *Automatica* **30**, 75.

Qin, S. J. and T. A. Badgwell (1996). An overview of industrial model predictive technology. *Proc. CPC-V, Lake Tahoe, CA* pp. 232–256.

Sison, L. G. and E. K. P. Chong (1996). No-reset iterative learning control. *Proc. of IEEE Conf. on Decision and Control, Kobe, Japan* pp. 3062–3063.

Togai, M. and O. Yamano (1985). Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. *Proc. 24th IEEE Conf. on Decision and Control, Ft. Lauderdale, FL* pp. 1399–1404.

Uchiyama, M. (1978). Formation of high speed motion pattern of mechanical arm by trial. *Trans. the Society of Instrum. and Control Engineers* **19**, 706–712.

Wilde, D. J. and C. S. Beightler (1967). *Foundations of Optimization*. Prentice-Hall. Englewood-Cliffs.

Zafiriou, E. and J. M. Zhu (1990). Optimal control of semi-batch processes in the presence of modeling error. *Proc. of Amer. Control Conf., San Diego, CA* pp. 1644–1649.

Zafiriou, E., R. A. Adomaitis and G. Gattu (1995). Approach to run-to-run control for rapid thermal processing. *Proc. of Amer. Control Conf., Seattle, WA* pp. 1286–1288.

Table 1: Performance of full-order Q-ILC. After 7 runs, the profiles converged. The initial wafer temperature, T_0 was dropped to 370^oC in the 8^{th} Run and then returned to 400^oC in the next run. (Temperature gap is defined to be the largest difference among the 8 temperatures. Mean square error here means $\frac{1}{8}\sum_{i=1}^{8}(y_i(t)-r)^2$. For both, 'Max' and 'Min' entries represent the maximum and minimum values over the course of a run. 'Mean' entry represents the average over time.)

	Temperature Gap			Mean Square Error		
Run	Max.	Min.	Mean	Max.	Min.	Mean
1^{st}	41.35	4.357	22.35	8250	2.170	3136
3^{rd}	17.49	4.707	10.97	563.8	3.739	77.29
7^{th}	8.116	1.893	4.968	91.80	1.667	14.97
8^{th}	9.880	4.429	7.030	880.5	2.433	97.62
9^{th}	7.613	4.189	6.265	95.20	1.821	7.374

Table 2: Comparison of the gap temperature between T_ω-model-based and T_ω^4-model-based Q-ILC.

	Temperature Gap		
Model	Max.	Min.	Mean
T_ω	12.02	5.371	7.740
T_ω^4	8.116	1.893	4.968

Table 3: Performance of Q-ILC with 3 point measurements and 3 point control. Results for (a) three controlled points and (b) 8 monitored points.

	Temperature Gap			Mean Square Error		
$7^{th}Run$	Max.	Min.	Mean	Max.	Min.	Mean
(a)	7.840	0.110	1.625	197.9	0.032	7.890
(b)	27.53	14.90	20.29	437.8	39.99	102.3

Table 4:Performance of Q-ILC with 3 point measurements and 8 point control under explicit inference of the unmeasured temperatures with different initial temperature,T_0.

8^{th} Run	Temp Gap			Mean Square Error		
$T_0/^oC$	Max.	Min.	Mean	Max.	Min.	Mean
350	10.96	5.553	7.776	2463	4.354	256.5
400	12.87	6.282	8.903	106.2	5.450	22.97
450	15.86	6.702	9.509	2516	4.601	422.1