Model Predictive Control Technique Combined with Iterative Learning for Batch Processes

Kwang S. Lee, In-Shik Chin, and Hyuk J. Lee
Department of Chemical Engineering, Sogang Univ., Mapogu, Seoul 121-742, Korea

Jay H. Lee
School of Chemical Engineering, Purdue Univ., West Lafayette, IN 47907

A novel model predictive control technique geared specifically toward batch process applications is demonstrated in an experimental batch reactor system for temperature tracking control. The technique, called Batch-MPC (BMPC), is based on a time-varying linear system model (representing a nonlinear system along a fixed trajectory) and utilizes not only the incoming measurements from the ongoing batch, but also the information stored from the past batches. This particular feature is shown to be essential for achieving effective tracking control performance despite model errors and disturbances. In a series of experiments performed on a bench-scale batch reactor system, the technique was found to deliver satisfactory tracking performance, as expected, overcoming a large amount of model uncertainty and various process disturbances.

Introduction

Model predictive control (MPC), despite its widely publicized industrial success, has been applied almost exclusively to continuous processes. For batch processes, there have been very few reported industrial applications of MPC. This is in spite of growing importance of batch operation in a large number of industrial sectors, ranging from traditional chemical and polymerization industries to emerging pharmaceutical, material processing, and semiconductor industries. In part, this can be attributed to the fact that the nature of batch process operations and control problems that arise from them are very different from those for the continuous processes. Whereas continuous process control tends to emphasize regulatory control around a fixed operating point—or within a narrow operating region—batch process operations are by nature much more dynamic, involving several transitions and large transient phases that cover large operating envelopes.

Most batch operations are described as hybrid systems involving discrete and continuous decision variables. Theories for modeling, optimization, and control of such systems are still at infancy and a satisfactory framework for "complete" model predictive control—including (real-time) sequence/trajectory optimization and abnormal situation management—is yet to emerge. Within the current mode of batch operation, once the recipe gets fixed and the operating condition optimized, control problems are generally formulated as tracking of given time-varying reference trajectories over a fixed time period, which may represent the whole batch or a particular phase of batch. The same time-varying trajectories tend to get used batch after batch until a different product needs to be made or some abnormal situation occurs which necessitates a change in the operating procedure. This article will address this particular problem of tracking given time-varying reference trajectories repeated over many batches. The problem of optimizing operating procedures for product/feed transitions, their real-time refinement, and abnormal situation management is important, but is beyond our present scope.

During the course of a typical batch operation, process variables swing over a wide range, exposing the nonlinearities inherent in the process dynamics. Owing to this, time-invariant linear models often fail to describe process dynamics adequately, rendering traditional linear control techniques ineffective. Most research efforts in batch process

Correspondence concerning this article should be addressed to K. S. Lee or J. H. Lee.

control have therefore been dedicated to the problem of designing a feedback controller that can compensate for strong effects of nonlinearity. Nonlinear cascade control (Marroquin and Luyben, 1972), generic model control and its variants (Lee and Sullivan, 1988; Cott and Macchietto, 1989), successive-linearization-based control (Lee and Datta, 1994), feedback linearization (Soroush and Kravaris, 1992), adaptive control (Cabassud et al., 1989; Fileti and Pereira, 1997; Wang et al., 1995), adaptive MPC (Jarupintusophon et al., 1994), and nonlinear MPC (Nagy and Agachi, 1997) are some of the approaches that have been pursued in academia, whereas most industrial problems have been solved by gain-scheduling the PID control parameters, sometimes with appropriate feedforward compensations, such as those for the heat of reaction inferred from a calorimetric balance (Juba and Hamer, 1986).

One of the traits of batch processes—left unexplored in control thus far—is that they involve repetitive operations. This characteristic endows the operator (and/or the controller) with a unique opportunity to make compensations based on errors from the previous batches. For instance, errors due to a bias in the input trajectory will repeat themselves in the subsequent batches. Traditional feedback control can remove these errors only to a certain extent, limited by system dynamics and model uncertainty. The same can be said for any disturbance that repeats itself or exhibits strong batch-wise correlation. Indeed, this very idea has been pursued in the robotics community, under the generic name of iterative learning control (ILC) (Arimoto et al., 1984; Chen, 1998). In the chemical process control side, however, there have been only a few activities of this kind (Katoh et al., 1988; Lee et al., 1994). Unfortunately, the ILC techniques developed for robot-arm training do not take into account the unique characteristics of chemical processes—like the typical overdamped nonlinear dynamics, significant interactions, large model errors, active constraints, and so on—and therefore, lead to problems when applied to these types of processes. Motivated by this, Lee and Lee (1996) proposed a model-based ILC framework called Q-ILC, as it is based on a quadratic optimization criterion, tailored specifically for chemical process control applications. Independent of this work, Zafiriou et al. (1995) has proposed a run-to-run optimization technique, a variant of ILC, where the result from a previous run is used along with the model to improve the operating trajectories for a new run in the presence of model errors.

The pure learning algorithms lack real-time feedback action, however, and cannot handle disturbances as they occur; it must wait until the next cycle to make any compensation. Various ways to combine feedback control with ILC have been proposed. Examples include: direct addition of a PID-type controller to an existing ILC algorithm (Lee et al., 1994); conversion of the ILC equation into a state feedback form (Amann et al., 1996); cascade configuration with feedback control for the inner loop and learning control for the outer loop (Kuc and Lee, 1996), and so on. However, these methods have not addressed the unique aspects of chemical or related batch processes. In addition, many of them have considered the feedback design as a separate task—rather than as an integral aspect of the learning controller design.

These considerations motivated us to develop a novel MPC technique for batch processes, which we present in this article. The proposed control technique, called *Batch-MPC* (BMPC), is based on a time-varying MIMO linear model (representing a nonlinear system along a fixed trajectory) and has been derived by integrating the concept of iterative learning into the conventional MPC technique. As a consequence, BMPC is capable of eliminating persisting errors from previous runs in addition to responding to new disturbances as they occur during a run. We evaluate the performance of BMPC in an experimental batch reactor system involving a highly exothermic reaction where a pre-specified trajectory for the reaction temperature has to be tracked. Rather than carrying out the actual reaction experimentally, we simulate the heat of reaction using an electric heater. The process was identified as a linear time-varying model by combining two linear time-invariant models with time-dependent weighting functions. The BMPC algorithm leaves the user with several parameters to choose. Some of them such as weighting factors in the cost function can be chosen based on their physical significance, but others such as the covariance matrices need some discretion in tuning. We discuss their effects on the performance and offer some guidelines for tuning.

The BMPC algorithm is derived and its properties are investigated—the mathematical results, as well as their implications for practical applications. The experimental system along with the procedure for model identification and controller tuning are described. Experimental results and discussions are presented and conclusions are drawn.

Batch-MPC Algorithm

Mathematical description of batch processes for BMPC

We consider a MIMO discrete-time batch process where the run length is fixed and consists of N sample steps. The problem is to manipulate the given inputs within given constraints so that the outputs follow specific reference trajectories. In batch operations, it is frequently encountered that some inputs are manipulated from the midst of a run while others are changed from the very start. Initiator addition in polymerization reactors usually begins at a specific time instant. The same is true for the outputs. Some outputs are controlled over an entire run while others are taken care of only over a limited duration. In order to accommodate such aspects in a single model structure, we allow the input and output dimensions to vary with time, say $n_u(t)$ and $n_y(t)$. For the future use, we define

$$N_y(t) \triangleq \sum_{l=1}^{t} n_y(l), \quad N_u(t), \triangleq \sum_{l=0}^{t-1} n_u(l) \tag{1}$$

A batch operation is typically modeled with a dynamical system, but it will prove to be convenient to consider a static map relating the input sequence to the output sequence over the whole batch horizon. Let us define the input, output and disturbance sequences as

$$\begin{aligned} \boldsymbol{u} &\triangleq \quad \left[u^T(0)\ u^T(1)\ \cdots\ u^T(N-1)\right]^T \in R^{N_u(N)} \\ \boldsymbol{y} &\triangleq \quad \left[y^T(1)\ y^T(2)\ \cdots\ y^T(N)\right]^T \in R^{N_y(N)} \\ \boldsymbol{d} &\triangleq \quad \left[d^T(1)\ d^T(2)\ \cdots\ d^T(N)\right]^T \end{aligned} \tag{2}$$

The input-output relationship for a general nonlinear batch system can be written in the form of

$$y = \mathfrak{N}(u, d) \tag{3}$$

where $\mathfrak{N}$ is some nonlinear map.

Let $\bar{y}$ and $\bar{u}$ represent the specified output reference trajectory and the corresponding nominal input trajectory. Hence,

$$\bar{y} = \mathfrak{N}(\bar{u}, \mathbf{0}) \tag{4}$$

Combining Eq. 4 with Eq. 3, we can write the equation for the error trajectory e as

$$e \overset{\Delta}{=} \bar{y} - y = \mathfrak{N}(\bar{u}, \mathbf{0}) - \mathfrak{N}(u, d) \tag{5}$$

Since the map $\mathfrak{N}$ is not known accurately in general and some of the parameters could change with time, it is unrealistic to assume exact knowledge of the nominal input trajectory. Hence, we assume that some coarse estimate of $\bar{u}$ (denoted as $\hat{\bar{u}}$) is available to us *a priori*. Then, linearizing the above error trajectory equation at $u = \hat{\bar{u}}$ and $d = \mathbf{0}$, we obtain

$$e = \left.\frac{\partial \mathfrak{N}}{\partial u}\right|_{(\hat{\bar{u}}, \mathbf{0})} (\bar{u} - \hat{\bar{u}}) - \left.\frac{\partial \mathfrak{N}}{\partial u}\right|_{(\hat{\bar{u}}, \mathbf{0})} (u - \hat{\bar{u}}) - \left.\frac{\partial \mathfrak{N}}{\partial d}\right|_{(\hat{\bar{u}}, \mathbf{0})} d + \text{higher order terms} \tag{6}$$

Denoting $\partial \mathfrak{N}/\partial u$ and $\partial \mathfrak{N}/\partial d$ as G and G_d, respectively, and rearranging the above, we obtain the following linear equation for the error trajectory

$$e = G(\hat{\bar{u}} - u) + \underbrace{G(\bar{u} - \hat{\bar{u}}) - G_d d + \text{higher order terms}}_{e^d} \tag{7}$$

Hence, e^d represents the error due to the bias in the nominal input trajectory ($\bar{u} - \hat{\bar{u}}$), as well as the effect of disturbances and model error. In general, e^d has both deterministic and stochastic components. For estimator design, it is convenient to model e^d as output of a linear system driven by random inputs

$$\begin{aligned} x_{k+1}^e &= A x_k^e + B w_k \\ e_k^d &= C x_k^e + v_k \end{aligned} \tag{8}$$

Note that subscript k represents the batch index. w_k and v_k are zero-mean (batch-index-wise) independent and identically distributed (i.i.d.) random vector variable sequences. For the sake of simplicity, we restrict our discussion to the particular choice of $A = I$, $B = I$, $C = I$. Then, we have

$$\begin{aligned} \bar{e}_{k+1}^d &= \bar{e}_k^d + w_k \\ e_k^d &= \bar{e}_k^d + v_k \end{aligned} \tag{9}$$

Note that, with this model choice, $\bar{e}^d$ can be interpreted as the part of e^d that will repeat itself in the subsequent batches (such as the error due to the bias in the nominal input trajectory and batch-index-wise constant disturbances) while v can be thought of as the part that will disappear in the subsequent batches.

Similarly, we can define $\bar{e}$ as $[\bar{e}^d + G(\hat{\bar{u}} - u)]$, which represents *the part of the error trajectory* e *that will repeat itself in the next batch*, assuming input u remains the same. Writing the expression for $\bar{e}$ for two consecutive batch indices and differencing yields the following transition model for the tracking error trajectory

$$\begin{aligned} \bar{e}_{k+1} &= G(\hat{\bar{u}} - u_{k+1}) + \bar{e}_{k+1}^d \\ \bar{e}_k &= G(\hat{\bar{u}} - u_k) + \bar{e}_k^d \\ &\Downarrow \\ \bar{e}_{k+1} &= \bar{e}_k - G \Delta u_{k+1} + w_k \\ e_k &= \bar{e}_k + v_k \end{aligned} \tag{10}$$

where $\Delta u_{k+1} = u_{k+1} - u_k$. Note that Δ here denotes a difference operator with respect to the batch index and not with respect to the time index.

Remarks

(1) By causality, the structure of G is restricted to the following lower-block triangular form

$$G = \begin{bmatrix} g_{1,0} & 0 & \cdots & 0 \\ g_{2,0} & g_{2,1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ g_{N,0} & \cdots & \cdots & g_{N,N-1} \end{bmatrix} \in R^{N_y(N) \times N_u(N)} \tag{11}$$

where $g_{i,j} \in R^{n_y(i) \times n_u(j)}$ is the impulse response coefficient matrix composed of output responses at time i to independently applied unit pulse inputs at time j.

(2) G can be found by linearizing a nonlinear model along the nominal trajectories or through direct identification.

(3) Both v_k and w_k are assumed zero-mean i.i.d. sequences with respect to the batch index. The elements of these vectors may be correlated, however, due to the *temporal* correlation in the disturbance effects. When exact statistics are not available, a reasonable model for most chemical batch processes is a filter integrated white noise sequence along t. This forces the covariance matrices for v and w to have a specific structure. This will be discussed in more detail later.

Derivation of batch MPC algorithm

Formulation of a State-Space Model along the Time Index for Real-Time Control. For real-time prediction and control, we now convert the batch-index transition model Eq. 10 into a time-index transition model. For this, let us partition G according to

$$G \overset{\Delta}{=} [G(0)\; G(1)\; \cdots\; G(N-1)], \quad G(j) \in R^{N_y(N) \times n_u(j)} \tag{12}$$

Also, we define the state $\boldsymbol{e}_k(t)$ as the error sequence for the kth batch when the same input as the $k-$first batch is applied *starting at time t*, that is,

$$\boldsymbol{e}_k(t) \overset{\Delta}{=} \boldsymbol{e}_k \text{ with } \Delta u_k(t) = \cdots = \Delta u_k(N-1) = 0$$
$$= \bar{\boldsymbol{e}}_{k-1} - G(0)\Delta u_k(0) - \cdots - G(t-1)\Delta u_k(t-1) + \boldsymbol{w}_{k-1} + \boldsymbol{v}_k \tag{13}$$

Note that, with respect to the previous notation, $\boldsymbol{e}_k(N) = \boldsymbol{e}_k$.

Using this definition, the transition from the end of a batch to the start of the next batch can be written as

$$\boldsymbol{e}_k(0) = \bar{\boldsymbol{e}}_{k-1}(N) + \boldsymbol{w}_{k-1} + \boldsymbol{v}_k \tag{14}$$

In addition, if we write Eq. 13 for $t+1$ and take the difference, the following equation for the time transition of $\boldsymbol{e}_k(t)$ within a batch is obtained

$$\boldsymbol{e}_k(t+1) = \boldsymbol{e}_k(t) - G(t)\Delta u_k(t), \quad t = 1, \ldots, N \tag{15}$$

Since the initial condition for $\boldsymbol{e}_k(t)$ is determined by $\bar{\boldsymbol{e}}_{k-1}$, we need a state transition equation for $\bar{\boldsymbol{e}}_k(t+1)$, too. $\bar{\boldsymbol{e}}_k(t)$ can be defined in the same manner to yield

$$\bar{\boldsymbol{e}}_k(0) = \bar{\boldsymbol{e}}_{k-1}(N) + \boldsymbol{w}_{k-1}$$
$$\bar{\boldsymbol{e}}_k(t+1) = \bar{\boldsymbol{e}}_k(t) - G(t)\Delta u_k(t) \tag{16}$$

By combining Eqs. 15 and 16, we have the following periodically time-varying state-space model (with period of N) for the controller design

$$\begin{bmatrix} \bar{\boldsymbol{e}}_k(t+1) \\ \boldsymbol{e}_k(t+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{e}}_k(t) \\ \boldsymbol{e}_k(t) \end{bmatrix} - \begin{bmatrix} G(t) \\ G(t) \end{bmatrix} \Delta u_k(t)$$
$$e_k(t) = [0 \; H(t)] \begin{bmatrix} \bar{\boldsymbol{e}}_k(t) \\ \boldsymbol{e}_k(t) \end{bmatrix} \quad t = 0, \ldots, N-1 \tag{17}$$

with

$$\begin{bmatrix} \bar{\boldsymbol{e}}_k(0) \\ \boldsymbol{e}_k(0) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{e}}_{k-1}(N) \\ \boldsymbol{e}_{k-1}(N) \end{bmatrix} + \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{w}_{k-1} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{v}_k \tag{18}$$

where

$$H(t) = \Big[\underbrace{\boldsymbol{0}}_{n_y(t) \times N_y(t-1)} \quad \underbrace{\boldsymbol{I}}_{n_y(t) \times n_y(t)} \quad \underbrace{\boldsymbol{0}}_{n_y(t) \times (N_y(N) - N_y(t))} \Big] \tag{19}$$

We note that the error sequence over the entire batch horizon is defined as the state. As a result, the measurement matrix has a time-varying structure. Also, for the estimation of $\bar{\boldsymbol{e}}$, the states $\boldsymbol{e}$ may be viewed as redundant. However, for real-time predictive control, it is indeed $\boldsymbol{e}$ we are interested in predicting. $\bar{\boldsymbol{e}}$ can be viewed as the information that needs to be stored for the next batch.

Predictor Construction. Once the state-space model is formulated, the subsequent steps for prediction and control calculation are rather straightforward. For prediction, it is convenient to use the standard optimal state estimator for the state space system in Eqs. 17, 18 and 19. It is a standard result that the optimal (linear) estimator for the system of Eqs. 17–19 is the Kalman filter (Åström and Wittenmark, 1990), which is described by

$$\begin{bmatrix} \bar{\boldsymbol{e}}_k(t|t-1) \\ \boldsymbol{e}_k(t|t-1) \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{e}}_k(t-1|t-1) \\ \boldsymbol{e}_k(t-1|t-1) \end{bmatrix} - \begin{bmatrix} G(t-1) \\ G(t-1) \end{bmatrix} \Delta u_k(t-1)$$
$$\begin{bmatrix} \bar{\boldsymbol{e}}_k(t|t) \\ \boldsymbol{e}_k(t|t) \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{e}}_k(t|t-1) \\ \boldsymbol{e}_k(t|t-1) \end{bmatrix}$$
$$+ \boldsymbol{K}_k(t)\big[e_k(t) - H(t)\boldsymbol{e}_k(t|t-1)\big], \quad \text{for } t = 1, \ldots, N \tag{20}$$

and

$$\begin{bmatrix} \bar{\boldsymbol{e}}_k(0|0) \\ \boldsymbol{e}_k(0|0) \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{e}}_{k-1}(N|N) \\ \bar{\boldsymbol{e}}_{k-1}(N|N) \end{bmatrix} \tag{21}$$

The dynamic Kalman gain matrix is calculated by

$$\boldsymbol{K}_k(t) = \begin{bmatrix} \hat{\boldsymbol{P}}_k(t) \\ \boldsymbol{P}_k(t) \end{bmatrix} H^T(t)\big[H(t)\boldsymbol{P}_k(t)H^T(t)\big]^{-1} \tag{22}$$

where the covariance matrices are updated according to

$$\begin{bmatrix} \bar{\boldsymbol{P}}_k(t+1) & \hat{\boldsymbol{P}}_k(t+1) \\ \hat{\boldsymbol{P}}_k(t+1) & \boldsymbol{P}_k(t+1) \end{bmatrix}$$
$$= \begin{bmatrix} \bar{\boldsymbol{P}}_k(t) & \hat{\boldsymbol{P}}_k(t) \\ \boldsymbol{P}_k(t) & \boldsymbol{P}_k(t) \end{bmatrix} - \boldsymbol{K}_k(t)H(t)\big[\hat{\boldsymbol{P}}_k(t) \; \boldsymbol{P}_k(t)\big] \tag{23}$$

for $t = 1, \ldots, N$ and reset at the beginning of each new cycle according to

$$\begin{bmatrix} \bar{\boldsymbol{P}}_k(1) & \hat{\boldsymbol{P}}_k(1) \\ \hat{\boldsymbol{P}}_k(1) & \boldsymbol{P}_k(1) \end{bmatrix}$$
$$= \begin{bmatrix} \bar{\boldsymbol{P}}_{k-1}(N+1) + \boldsymbol{R}_w & \bar{\boldsymbol{P}}_{k-1}(N+1) + \boldsymbol{R}_w \\ \bar{\boldsymbol{P}}_{k-1}(N+1) + \boldsymbol{R}_w & \bar{\boldsymbol{P}}_{k-1}(N+1) + \boldsymbol{R}_w + \boldsymbol{R}_v \end{bmatrix} \tag{24}$$

Now, let $\boldsymbol{e}_k(t+m|t)$ represent the prediction of the error sequence for the kth batch, made at time t. To construct $\boldsymbol{e}_k(t+m|t)$, we must add to $\boldsymbol{e}_k(t|t)$ the effect of *m future* control moves

$$\boldsymbol{e}_k(t+m|t) = \boldsymbol{e}_k(t|t) - \boldsymbol{G}^m(t)\Delta \boldsymbol{u}_k^m(t) \tag{25}$$

where

$$G^m(t) = [G(t), \ldots, G(t+m-1)] \quad \text{and}$$

$$\Delta \boldsymbol{u}_k^m(t) = \begin{bmatrix} \Delta u_k(t) \\ \vdots \\ \Delta u_k(t+m-1) \end{bmatrix} \tag{26}$$

The control horizon m needs to be shrunk as t approaches toward N, the batch end time.

The above state estimator looks computationally demanding, mainly due to the need for gain update through the Riccati iteration at each sample time. However, the periodically varying Riccati difference equations defined by Eqs. 22, 23 and 24 will eventually converge (with respect to the batch index) to a periodic solution, irrespective of the initial value. Hence, one may pre-calculate the periodic steady-state solution and the corresponding Kalman gain matrices ($\boldsymbol{K}_\infty(1), \ldots, \boldsymbol{K}_\infty(N)$) off-line and avoid the on-line Riccati iteration.

Input Calculation. Unconstrained Case. Now we compute the control input sequence that minimizes $\boldsymbol{e}_k(t+m|t)$, the tracking error over the whole batch horizon, while penalizing excessive input movements. Employing the conventional quadratic objective function, the control input can be calculated according to

$$\min_{\Delta \boldsymbol{u}_k^m(t)} \frac{1}{2}\left\{\boldsymbol{e}_k^T(t+m|t)\boldsymbol{Q}\boldsymbol{e}_k(t+m|t) + \Delta\boldsymbol{u}_k^{mT}(t)\boldsymbol{R}\Delta\boldsymbol{u}_k^m(t)\right\} \tag{27}$$

For the unconstrained case, an analytical solution to the above minimization can be obtained as

$$\Delta\boldsymbol{u}_k^m(t) = \left[\boldsymbol{G}^{mT}(t)\boldsymbol{Q}\boldsymbol{G}^m(t) + \boldsymbol{R}\right]^{-1}\boldsymbol{G}^{mT}(t)\boldsymbol{Q}\boldsymbol{e}_k(t|t) \tag{28}$$

Only $\Delta u_k(t)$ from $\Delta\boldsymbol{u}_k^m(t)$ is applied to the process at each sampling time.

Constrained Case. Constraints can be imposed on the input magnitude, the input changes (both in terms of batch and time indices), and the predicted outputs. Generally, they are given as the following linear inequalities

$$\begin{aligned}
\boldsymbol{u}^{\text{low}} \le \quad & \boldsymbol{u}_k && \le \boldsymbol{u}^{\text{hi}} \\
\Delta\boldsymbol{u}^{\text{low}} \le \quad & \Delta\boldsymbol{u}_k && \le \Delta\boldsymbol{u}^{\text{hi}} \\
\delta u^{\text{low}} \le \quad & u_k(t) - u_k(t-1) && \le \delta u^{\text{hi}}, \text{ with } u_k(-1) = u_k(0) \\
\boldsymbol{y}^{\text{low}} - \epsilon_k(t) \le \quad & y_k(t+m|t) && \le \boldsymbol{y}^{\text{hi}} + \epsilon_k(t), \quad \epsilon_k(t) > 0
\end{aligned} \tag{29}$$

At time t of the kth batch, these constraints can be put in the standard form

$$\mathcal{C}^m(t)\Delta\boldsymbol{u}_k^m(t) \ge \mathcal{B}_k^m(\epsilon_k(t), t) \quad \text{and} \quad \epsilon_k(t) \ge \boldsymbol{0} \tag{30}$$

The role of $\epsilon_k(t)$ in the output constraint equation deserves some comments. $\epsilon_k(t)$ is a *slack variable* introduced to make sure that the constraints are always feasible (Zafiriou and Chiou, 1993). This is referred to as *constraint softening* and is a standard practice in MPC (Morari and Lee, 1997). Without the slack variables, the constraints can become infeasible since there may not exist any input sequence—within the given constraints—that can keep the output within the specified bounds. This would obviously create problems in on-line applications. Note that, by choosing a sufficiently large ϵ, the output constraints can always be satisfied. However, we do not want to "slacken" the constraints any more than we need to. Hence, the objective function is modified to include a quadratic term of $\epsilon_k(t)$ so that the slack variable is made as small as feasible. The corresponding optimization problem is

$$\min_{\Delta\boldsymbol{u}_k^m(t),\, \epsilon_k(t)} \frac{1}{2}\left\{\boldsymbol{e}_k^T(t+m|t)\boldsymbol{Q}\boldsymbol{e}_k(t+m|t) + \Delta\boldsymbol{u}_k^{mT}(t)\boldsymbol{R}\Delta\boldsymbol{u}_k^m(t) + \epsilon_k^T(t)\boldsymbol{S}\epsilon_k(t)\right\} \tag{31}$$

subject to Eqs. 25 and 30.

The above objective with Eq. 30 constitute a quadratic program (QP) (Garcia and Morshedi, 1984). At each time, the QP is solved and the first element (the optimal input move corresponding to the current time) is implemented.

Summary of the algorithm

The procedure for BMPC implementation is summarized as follows

Step 1. Obtain dynamic matrix $\boldsymbol{G}$ using an appropriate method.

Step 2. Initialize the state estimator. We can choose $\bar{\boldsymbol{e}}_0(N|N)$ ($= \boldsymbol{e}_0(N|N)$) and $\boldsymbol{u}_0$ as the error trajectory and the input trajectory obtained from a previous batch run. If there is no previous run, one can use the zero vector for $\bar{\boldsymbol{e}}_0(N|N)$ and the best available estimate for $\boldsymbol{u}_0$. Also, let

$$\begin{bmatrix} \bar{\boldsymbol{P}}_0(N+1) & \hat{\boldsymbol{P}}_0(N+1) \\ \hat{\boldsymbol{P}}_0(N+1) & \boldsymbol{P}_0(N+1) \end{bmatrix} = \begin{bmatrix} \gamma I & \gamma I \\ \gamma I & \gamma I \end{bmatrix} \tag{32}$$

where $\gamma > 0$ is chosen to be "fairly large." Set $k = 1$.

Step 3. Before the start of each fresh batch, initialize the state and the covariance matrix using Eqs. 21 and 24. Set $t = 1$.

Step 4. At time t, obtain the state estimate using Eqs. 20 and 22. Also, update the covariance matrix using Eq. 32.

Step 5. Calculate $\Delta u_k(t)$ using Eq. 28 or by solving Eq. 31. Implement $u_k(t)$.

Step 6. If $t < N$, set $t \leftarrow t+1$ and go back to Step 4. If $t = N$, set $k \leftarrow k+1$ and go to Step 3.

Comments on stability and robustness

In the context of batch systems, the notion of closed-loop stability should be considered over the batch index. It is because a batch operation is defined over a finite time interval and the system is reset at every initial time. Note that, since the model underlying the BMPC algorithm (Eq. 17) is a set of pure integrators, any nonzero $\boldsymbol{e}(0)$ will persist if no adjustment is made to the input. Therefore, the convergence, $\boldsymbol{e}_k \to \boldsymbol{0}$ as $k \to \infty$, from a nonzero initial condition means that, with BMPC, any initial bias in the error trajectory, low or high frequency, can be removed completely.

In order to show the stability, one needs to introduce the following assumptions:

- $\boldsymbol{G}$ is known and has full row-rank.
- $\boldsymbol{Q}$ is a positive definite matrix.
- There exists some input trajectory that zeros the error trajectory without constraint violation.

The first requirement mandates that there be at least as many inputs as the outputs. This is reasonable since we cannot expect to zero the outputs with a less number of inputs—even at steady state. If this requirement is not satisfied, we can ask whether the BMPC algorithm will drive the error trajectory vector to the minimum achievable—in the given 2-norm sense. The second requirement says that all the errors must be independently weighed in the objective function. The third requirement is natural in discussing the convergence. In addition to the above, we may consider the deterministic case (for which $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are uniformly zero) for the purpose of showing the convergence.

Under the above assumptions, it is possible to prove the convergence of the error trajectory to zero. The main idea is to show that the optimal cost is a nonincreasing function with respect to time and must therefore converge. With this, we can argue that the input trajectory must converge under the optimal control and that the convergence of the input trajectory implies convergence of the error trajectory to zero due to the integral action inherent in the controller. This basic idea is similar to that used in proving the closed-loop stability of conventional MPC (Zheng and Morari, 1995), but rigorous proof in the general case can become quite complex and technical since the effects of state estimation error and constraints should be taken into account together. Hence, this will be addressed in a future article.

Drawing an analogy to the fact that a feedback controller with integral action can provide a zero-offset regulation against a constant disturbance despite model errors, we can expect the convergence result to hold up in the presence of some model errors as well. Of course, too large a model error can also lead to divergence due to the feedback. The allowable model error bound depends on both the plant and tuning parameters in general. Theoretical investigation of model error effects and development of robust BMPC algorithms and/or tuning strategies is currently under way.

Tuning of BMPC

The BMPC algorithm has five major tuning factors: $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{R}_w$, $\boldsymbol{R}_v$, and the control horizon m. In this section, we discuss tuning of these parameters and propose some practical guidelines.

The control horizon m should be chosen as large as the computational resource allows. Ideally, one would like to compute the optimal trajectory for the entire remainder of the batch by choosing m at time t as $N-t$. This is called *shrinking horizon control* (Russell et al., 1998). Normally, however, such a choice would be computationally prohibitive, especially when the sample time is short in relation to the total batch time. However, since the prediction horizon is set as the entire batch by default, too small an m can cause sluggish performance, especially at the beginning. (Although not discussed, one always has the freedom of choosing the prediction horizon other than the entire remaining batch length within the BMPC framework.)

The weighting parameters are relatively easy to choose. First $\boldsymbol{Q}^{1/2}$ should be chosen to scale the outputs appropriately and assign relative importance to each output. $\boldsymbol{R}$ can be chosen as a scalar-times-identity matrix—as typically done in the conventional MPC—and the scalar can be used to adjust the relative weight of the input penalty to the tracking error suppression. Increasing the scalar slows down both the batch-to-batch learning and real-time feedback control feature, and lowers the sensitivity to batch-to-batch changes and the risk of divergence due to model errors. If one desires to have the flexibility of controlling the two features independently, it is best that the input penalty be kept to a minimum and tune using the noise covariance matrices.

Choosing $\boldsymbol{R}_w$, $\boldsymbol{R}_v$ requires some discretion. Physically, they represent covariance matrices of disturbance profiles, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$; however, the exact covariance matrices are not easy to obtain in practice. In general, it suffices to parameterize the stochastic disturbances with a small number of parameters and tune these parameters. For example, for most chemical processes, disturbances are reasonably modeled by summing integrated white noise (representing disturbances with strong time correlation) with white noise (representing disturbances with no time correlation) (Lee and Morari, 1994). In that case, we can express

$$v_k(t) = \frac{1}{1-q^{-1}} n_{v,1}(t) + n_{v,2}(t) \tag{33}$$

where $n_{v,1}(t)$ and $n_{v,2}(t)$ are vector-valued white noises of covariances $\sigma_{v,1}^2 I$ and $\sigma_{v,2}^2 I$, respectively. (Given that the outputs are scaled appropriately, the same noise parameter can be used for all the outputs. If not, one can scale the variance parameters for each output correspondingly.) These choices dictate that the covariance matrices take the following structure

$$\boldsymbol{R}_v = \sigma_{v,1}^2 \boldsymbol{J} + \sigma_{v,2}^2 \boldsymbol{I} \quad \text{where}$$
$$\boldsymbol{J} = \begin{bmatrix} I & I & I & \cdots & I \\ I & 2I & 2I & \cdots & 2I \\ I & 2I & 3I & \cdots & 3I \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ I & 2I & 3I & \cdots & NI \end{bmatrix} \quad \text{and} \quad \boldsymbol{I} = \text{identity matrix} \tag{34}$$

Modeling $w_k(\cdot)$ in the same way would give two additional variance parameters $-\sigma_{w,1}^2$ and $\sigma_{w,2}^2$. Hence, under the specific disturbance structure, one gets four parameters to tune.

In general, putting more weight on the white noise part in relation to the integrated white noise part for both $\boldsymbol{w}$ and $\boldsymbol{v}$ makes the real-time control less aggressive and, therefore, more robust. In addition, giving more weight to $\boldsymbol{v}$ in relation to $\boldsymbol{w}$ makes the trajectory change from one batch to the next smoother and increases the robustness to model errors—but at the expense of slowed convergence rate. Based on these general principles, we can offer the following guidelines for tuning:

• In most cases, we recommend that $\sigma_{w,2}^2$ be set to zero, since it is not desirable for the learning control part of the algorithm to attempt to control uncorrelated noises. The real-time control part automatically filters out white noises ($n_{v,2}$) before the control computation, because of the causality requirement. On the other hand, the iterative learning control part does not have this restriction and, therefore, will attempt to compensate for $n_{w,2}$ (occurred during a previous batch) if $\sigma_{w,2}^2$ is set significant in relation to $\sigma_{v,2}^2$.

• The input trajectories are judged to be too aggressive and oscillatory—with respect to t—causing a bad tracking performance. $\Rightarrow$ Lower $\sigma_{w,1}^2$ and $\sigma_{v,1}^2$ in relation to $\sigma_{v,2}^2$. Which of the two parameters to lower depends on whether one wants to slow down the learning rate as well as the real-time control. To keep the same learning rate, adjust only $\sigma_{v,1}^2$.

• There appears to be too large shifts in the input trajectories from batch to batch, causing a bad performance or giving the appearance of divergence—particularly with oscillation, in the error trajectory with respect to the batch index. $\Rightarrow$ Lower $\sigma_{w,1}^2$ in relation to $\sigma_{v,1}^2$ and $\sigma_{v,2}^2$.

Comparison with existing batch control techniques

The current industrial state of the art for the type of problems we are addressing in this article is PID control, possibly with some gain-scheduling feature and a feedforward compensation. The effectiveness of *any* feedback design is *fundamentally* limited by system dynamics and model accuracy. Hence, even in theory, perfect tracking of time-varying reference trajectories is not possible with feedback control alone—regardless of design methodology. In practice, PID control typically leaves large offsets, especially during transient periods (such as during and after a ramp-up or a ramp-down). In the case of temperature control of highly sensitive chemical reactors, a calorimetric balance can be used to infer the reaction heat and feedforward compensation can be added based on it (Juba and Hamer, 1986). However, this does require a detailed heat balance model and the performance improvement depends on its accuracy.

As a next step, one could conceivably employ the strategy where the input trajectory from a previous run—under conventional feedback control—is used as the nominal trajectory for the next run. This strategy can be generically described by

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{H}\boldsymbol{e}_{k+1} \tag{35}$$

where $\boldsymbol{H}$ represents any linear—possibly time-varying—feedback design. Note that because of this causality requirement, $\boldsymbol{H}$ must be a lower-triangular matrix. In addition, because of the particular way the vectors $\boldsymbol{u}$ and $\boldsymbol{e}$ are indexed, all the diagonal elements must be zero.

Now, if we substitute this into Eq. 10 with the assumption that $\boldsymbol{w}_k = \boldsymbol{v}_k = 0$, we obtain the following recursion equation for the tracking error

$$\boldsymbol{e}_{k+1} = (\boldsymbol{I} + \boldsymbol{G}\boldsymbol{H})^{-1}\boldsymbol{e}_k \tag{36}$$

For $\|\boldsymbol{e}_k\| \to 0$, it is obvious that $\boldsymbol{H}$ must be designed such that all the eigenvalues of $(\boldsymbol{I} + \boldsymbol{G}\boldsymbol{H})^{-1}$ lie strictly inside the unit circle. However, this is not possible; all the eigenvalues necessarily lie at $(1,0)$ as $\boldsymbol{G}$ and $\boldsymbol{H}$ are both lower triangular matrices and all the diagonal elements of $\boldsymbol{H}$ are zero. This means that offsets cannot be removed through this strategy.

Insteady of using the previous input trajectory as the nominal trajectory, one can use an iterative learning control algorithm and determine the nominal trajectory for the next run. This essentially combines the best of ILC and feedback control. There are several articles proposing this idea but very few in the MPC context. The work of Amann et al. (1996) discusses an implementation of a model-based ILC algorithm (based on a quadratic performance index) as a real-time state feedback. However, this algorithm is developed for linear a *linear-time-invariant* system with no constraints (which would not be suitable for representing chemical batch processes). In addition, their formation is completely deterministic. When interpreted in our combined deterministic/stochastic framework, this becomes equivalent to assuming $\boldsymbol{v} = \boldsymbol{0}$ uniformly. Of course, a potential problem of this is that, when there are significant batch-to-batch changes, the learning part of the algorithm can "overcorrect." Adjusting the sensitivity through the input weighting matrix $\boldsymbol{R}$ is not an attractive option, as this will also slow down the real-time control. The proposed BMPC algorithm achieves the essence of the idea of combining the best of ILC and conventional MPC, but it does not separate the iterative learning from the feedback control. Instead, it combines them as one integrated MPC algorithm. In addition, it affords enough flexibility to address the issues of concern in practical process control applications (such as constraint-handling, random disturbances of different types, use of time-varying models to address nonlinearity, and so on).

Limitations of the proposed BMPC technique

The BMPC technique, despite its merits over the available alternatives, does not fit into every existing batch process control problem. First, even though it is possible to include inputs and outputs that are active only during particular time windows of a batch, manipulating the time windows themselves is not possible within our formulation. Hence, it eliminates the possibility of contracting or dilating the operating trajectories by shortening or lengthening various phases. Such a decision must be relegated to the higher optimization layer. As long as these changes are not made on a too frequent basis (that iterative learning cannot take place), the BMPC can still be used by re-initializing every time a change is made. In fact, BMPC would be a valuable tool in this context as designing feedback controllers that can track different trajectories is a great challenge.

Second, the fact that we are using a linearized model means that operating trajectories from run to run cannot differ by very much. If the fundamental model is available, one could conceivably relinearize the model to update the dynamic matrix after each batch run.

Third, it is noteworthy that the ultimate purpose of temperature control lies not in itself, but in the regulation of final product quality. Considering the fact that quality is determined by factors besides the temperature trajectory (such

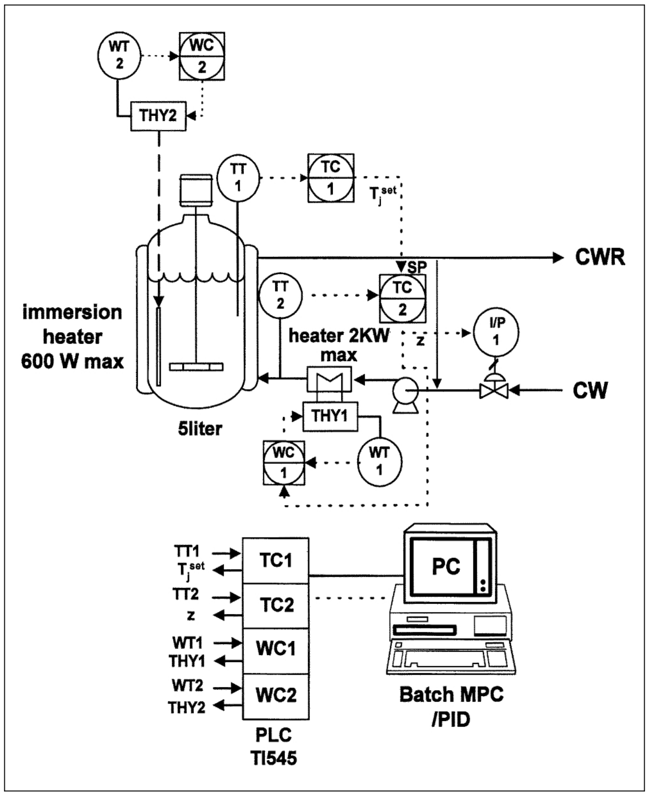

Figure 1. Experimental batch reactor system.

as the feedstock condition), maintaining a fixed temperature trajectory may not always be the best mode of operation. It should be possible to treat the issues of quality control (Russell et al., 1998) and temperature tracking all within a single control framework. Development of this technique is currently being carried out.

Experimental Studies

Batch reactor system and experimental procedure

In order to test the performance as well the implementation tips of the proposed BMPC technique, a series of experiments in a bench-scale batch reactor system has been conducted. Figure 1 shows the experimental batch system. The reactor vessel contains 5 L of water and has a jacket for heat exchange. The jacket temperature is adjusted by an electric heater installed in the heat-transfer liquid (water) circulation loop and by a cooling water control valve. The cooling water is supplied from a constant temperature bath. A special feature of this experimental system is an electric heater (600 Watt max.) immersed in the reactor content. This heater is used to simulate the heat that would be generated from any assumed reaction. For the purpose of temperature control experiments, this feature eliminates the necessity to actually carry out the reaction in the laboratory. Similar techniques have been used by Kershenbaum and Kittisupakorn (1994) with internal steam coil and also by Juba and Hamer (1986) with direct steam injection for large-scale pilot reactors. In order to eliminate the potential disturbance introduced by the fluctuation in the AC power source, the power from each electric heater is measured and independently controlled by high-gain P-controllers at every 100 ms.

Temperature control was conducted using a cascade control strategy. The BMPC algorithm (TC1) implemented in the PC calculates the jacket temperature set point, T_j^{set} and sends it to the slave controller (TC2) which is programmed in the PLC. The slave controller steers the jacket temperature to its set point by manipulating either the electric heater or the cooling water control valve. PI control was used for the slave controller.

The heat of reaction is calculated in the PC according to the following first-order exothermic reaction mechanism and sent to the Watt controller (WC2) as its set point

$$\frac{dC}{dt} = -kC$$

$$Q(t) = -\Delta H_r VkC$$

$$k = k_0 \exp\left(-\frac{E}{R(T_r + 273)}\right) \tag{37}$$

with

$$k_0 = 12.15\ (\mathrm{s}^{-1}) \qquad E = 22{,}000\ (\mathrm{J/mol \cdot K})$$

$$\Delta H_r = -50{,}000\ (\mathrm{J/mol}) \qquad C(0) = 1\ (\mathrm{mol/L})$$

Generally speaking, as the volume of a batch reactor decreases, the amount of heat of reaction per unit heat-transfer area decreases. This usually renders temperature control in a small batch reactor an easy task. To avoid such triviality, the amount of heat evolution at the onset of the reaction was designed to reach near the maximum cooling capacity of the system.

The batch operation was designed to proceed with five hypothetical stages as shown in Figure 2a. The first and second stages are for charge and heatup and last up to 2,200 s. At this point, the reaction is assumed to be initiated and continues for 1,500 s. After the reaction is completed, cooling and discharge stages follow. The heat generation pattern when the reaction temperature follows the nominal trajectory is shown in Figure 2b.

Two sets of experiments were carried out: one for the model identification and the other for testing the performance of BMPC. Details on the model identification experiment will be described in the following subsection. In the control experiments, the dual functions of BMPC—rejection of batch-wise correlated errors (learning) and real-time rejection of disturbances (real-time feedback control)—were tested separately. More specifically, in order to initialize BMPC, temperature control was conducted using a PI controller in the initial batch and then BMPC was repeated from there on with no additional disturbances. During this period, the only disturbances that could conceivably have entered the system were minor random perturbations such as measurement noises, fluctuations in the heat loss, and so on. Note, however, that the trajectories that we started out with were significantly biased. These errors would persist through the subsequent batches if no corrections were made. In addition, since the correction was to be calculated based on a model, model errors would inevitably introduce additional perturba-

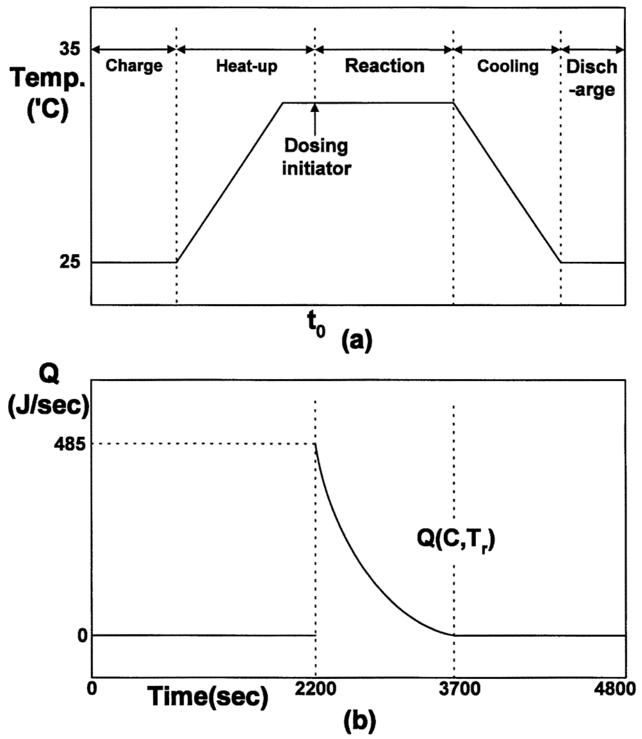

Figure 2. Reference temperature trajectory with operation steps (a) and a typical pattern of the heat of reaction (b).

tions that would also persist through the subsequent batches without further corrections. After the learning was completed, that is, after both the input and output trajectories had converged, additional disturbances were introduced to the system to investigate the real-time control performance of BMPC. Two different types of disturbances were entered: one in the initial reactor temperature and the other in the initial concentration. Both of them are commonly observed disturbances in industrial reactors. For comparison, we also tried Q-ILC (a pure learning algorithm) for the same disturbances.

The sampling interval was chosen to be 20 s, and the following input constraints were imposed throughout the experiments

$$\begin{aligned} 10°\text{C} \le & \quad T_{j,k}^{\text{set}}(t) & \le 45°\text{C} \\ -3°\text{C} \le \Delta T_{j,k}^{\text{set}}(t) & \overset{\Delta}{=} T_{j,k}^{\text{set}}(t) - T_{j,k-1}^{\text{set}}(t) & \le 3°\text{C} \end{aligned} \tag{38}$$

Model identification

For the design of BMPC, the system's dynamic matrix $\boldsymbol{G}$ should be available. Since direct identification of the time-varying impulse response coefficients is very difficult, we instead identified the following model consisting of two linear ARX models combined using time-varying weighting functions

$$A(q^{-1})y(t) = \mu_1(t)B(q^{-1})u(t) + \mu_2(t)C(q^{-1})u(t) + n(t) \tag{39}$$

where $n(\cdot)$ represents the residual term and

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na} \\ B(q^{-1}) &= b_1 q^{-1} + \cdots + b_{nb} q^{-nb} \\ C(q^{-1}) &= c_1 q^{-1} + \cdots + c_{nc} q^{-nc} \end{aligned} \tag{40}$$

In the above, B/A and C/A typically represent the reactor dynamics for the charge/heatup/cooling stage and the reaction stage, respectively; $u(\cdot)$ and $y(\cdot)$ stand for the input and output variables as defined in Eq. 41. Since no reaction takes place for the first 2,200 s and the reaction fades out as it approaches the cooling stage, the weighting functions were designed as shown in Figure 3 for smooth continuation from the reaction to cooling stages. In the figure, t_0 was *a priori* fixed at 2,200 s but t_1 and t_2 were optimized through an iterative procedure.

The identification experiment was carried out in two steps. In the first step, PI control was conducted with TC1 and the resulting input and output trajectories $T_r^*(\cdot)$ and $T_j^{\text{set}*}(\cdot)$ were recorded and taken as the nominal trajectories for the linear model. In the second step, a zero-mean ML-PRBS (maximum length pseudo-random binary signal) (Söderström and Stoica, 1989) was superimposed on $T_j^{\text{set}*}(\cdot)$ and the resulting signal was applied to the jacket temperature controller TC2 while TC1 is open. From the resulting input and output trajectories, $T_r(\cdot)$ and $T_j^{\text{set}}(\cdot)$, the input and output deviations were computed as

$$\begin{aligned} y(t) &\overset{\Delta}{=} T_r(t) - T_r^*(t) \\ u(t) &\overset{\Delta}{=} T_j^{\text{set}}(t) - T_j^{\text{set}*}(t) \end{aligned} \tag{41}$$

and the coefficients of the ARX model were estimated. Orders of each polynomial in Eq. 39 were chosen as $na = 3$, $nb = 2$, and $nc = 2$. For given t_1 and t_2, the model coefficients were estimated using the least-squares method such that the sum of the squared prediction errors over the batch horizon is minimized

$$\hat{\theta}(t_1,t_2) = \arg\min_{\theta}\left[J(\theta,t_1,t_2) = \sum_{t=1}^{N}\left(y(t) - \hat{y}(t;\theta,t_1,t_2)\right)^2\right] \tag{42}$$

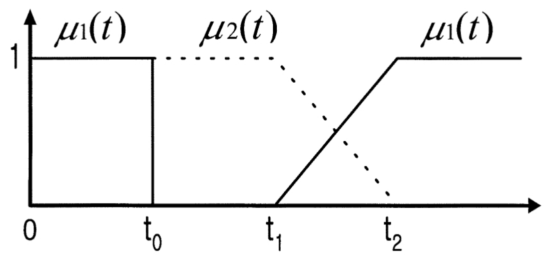

Figure 3. Time-varying weighting functions for the model construction.

where

$$\hat{y}(t;\theta,t_1,t_2) \overset{\Delta}{=} \phi(t;t_1,t_2)^T\theta$$

$$\theta \overset{\Delta}{=} [a_1\ a_2\ a_3\ b_1\ b_2\ c_1\ c_2]^T$$

$$\phi(t;t_1,t_2) \overset{\Delta}{=} [-y(t-1),\ -y(t-2),\ -y(t-3), \quad (43)$$

$$\mu_1(t;t_1,t_2)u(t-1),\ \mu_1(t;t_1,t_2)u(t-2),$$

$$\mu_2(t;t_1,t_2)u(t-1),\ \mu_2(t;t_1,t_2)u(t-2)]^T$$

Once the coefficients were obtained, t_1 and t_2 were optimized in the outer loop and the calculation was repeated until $J(\theta,t_1,t_2)$ reaches the minimum.

In Figure 4, open-loop (not one-step-ahead) output prediction by the resulting model $\hat{y}(t) = \mu_1(t)(B/A)u(t) + \mu_2(t)(C/A)u(t)$ is compared with the measured $y(t)$. We can see that the model prediction is biased from the measured values, but captures the coarse dynamics of the process. Since the proposed BMPC technique can withstand certain amounts of model errors, we took the resulting model and did not attempt to improve it further.

Choice of tuning parameters

The tuning parameters in the BMPC algorithm are weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, the control horizon m, and noise covariance matrices $\boldsymbol{R}_w$ and $\boldsymbol{R}_v$. Among them, $\boldsymbol{Q}$ and $\boldsymbol{R}$ were fixed at $\boldsymbol{I}$ and $0.005\boldsymbol{I}$, and $m = 20$ was used throughout the experiments.

In order to choose $\boldsymbol{R}_w$ and $\boldsymbol{R}_v$ in a rigorous manner, prior statistics of $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are needed. As was discussed in the previous section, the optimal covariance matrices of $\boldsymbol{w}$ and $\boldsymbol{v}$ were difficult to determine (it should account for the effect of model errors, feed disturbances, and so on). So we decided to use Eq. 34 which was obtained by assuming that these vectors were generated by summing integrated white noise and white noise. In our study, $\sigma_{w,1}^2 = 0.5$, $\sigma_{w,2}^2 = 0.001$ and $\sigma_{v,1}^2 = 0.5$, $\sigma_{w,2}^2 = 0.001$ were used throughout the experiments.

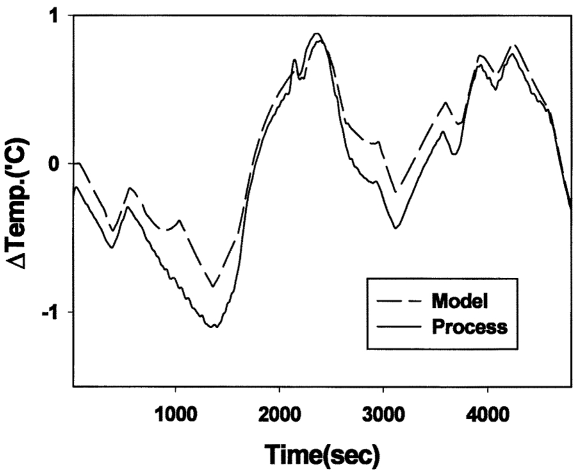

Figure 4. Process measurement vs. open-loop prediction by the identified model.

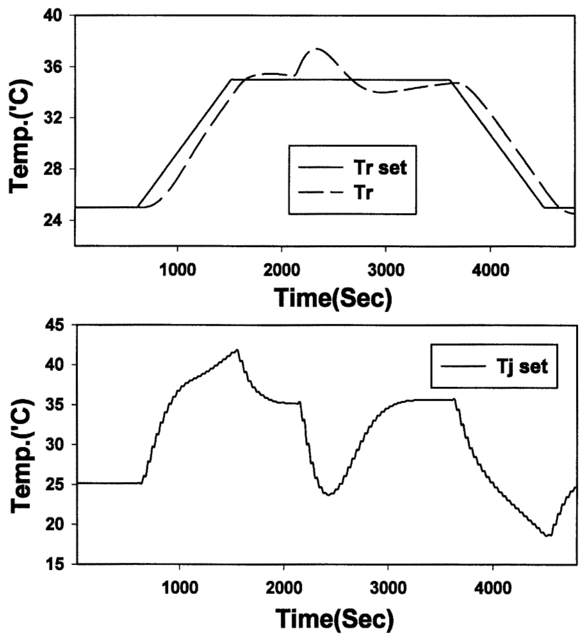

Figure 5. Performance of PI control (0th batch).

Results and Discussion

In Figure 5, the results from the initial batch run under PI control are shown. The temperature tracking exhibits a typical PI control response pattern with large lags during the upward and downward ramping. As soon as the reaction starts, a large deviation arises due to the heat of reaction, which is slowly removed after a rather long undershoot.

Based on the results from the initial run, BMPC was next applied with no additional disturbance for six batch runs. Figures 6 to 8 show the results from the first, the fourth, and

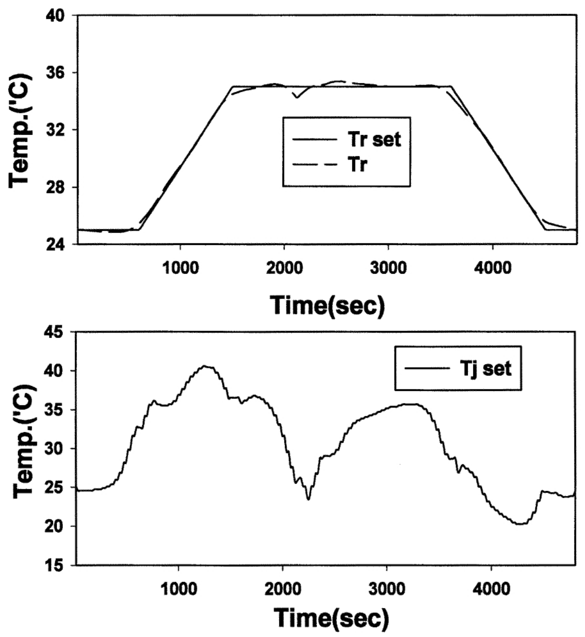

Figure 6. Performance of BMPC against input bias and model error (first batch).

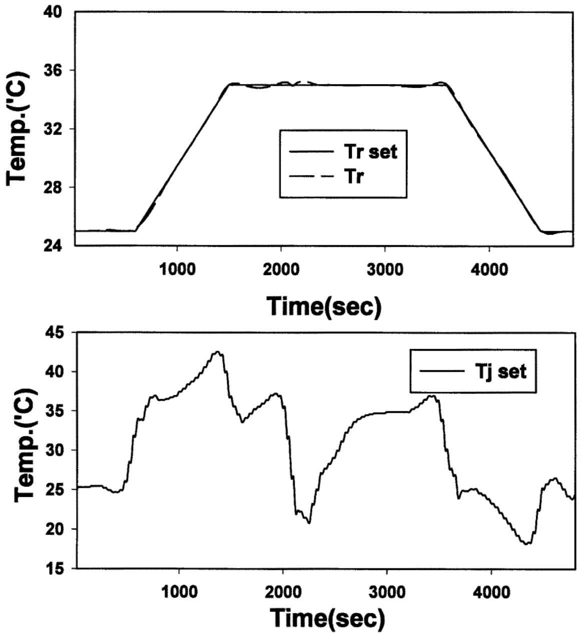

Figure 7. Performance of BMPC against input bias and model error (fourth batch).

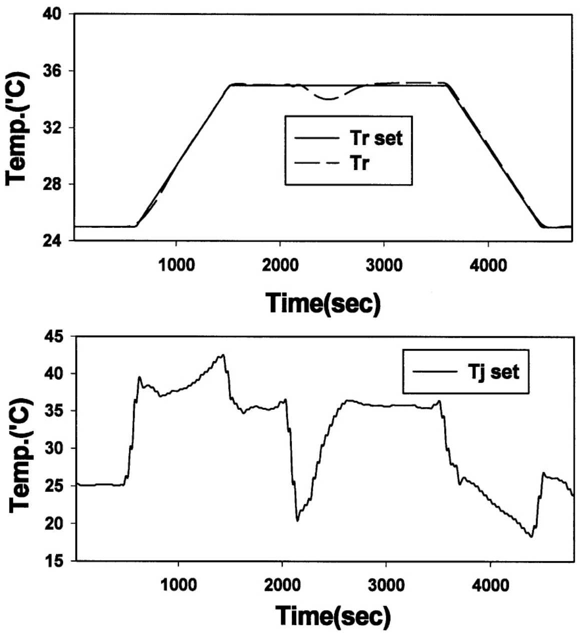

Figure 9. Performance of BMPC against a change in the reactant concentration ($C(0)$ is decreased from 1 to 0.85 (mol/L)).

the sixth runs with BMPC. As can be seen from Figure 6, the tracking error is substantially reduced in just one BMPC run. After five more consecutive runs, the output trajectories have virtually conversed and no further significant improvement can be made. In fact, as can be seen from Figure 7, the output trajectory already agrees very closely with the reference trajectory after just four batch runs.

For comparison, we attempted a pure learning algorithm without real-time feedback [direct-error-feedback Q-ILC as discussed in Lee and Lee (1996)] with the same tuning values and observed that at least ten or more batch runs were needed to attain the similar convergence. Apparently, the feedback were useful in reducing the effects of model errors—which tends to change from one batch to next with the input trajectory.

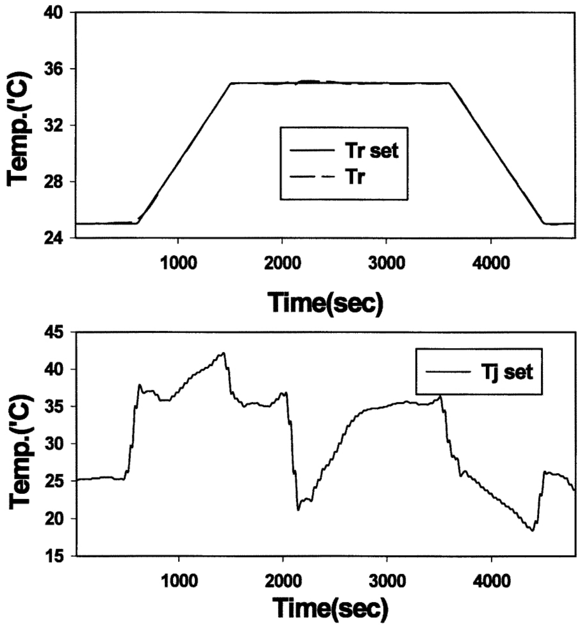

Figure 8. Performance of BMPC against input bias and model error (sixth batch).

After the six successive runs with no disturbances, $C(0)$ was decreased by 15% at the seventh batch. Figure 9 shows the performance of BMPC. As the heat release by the reaction is reduced with this disturbance, the reactor temperature showed a drop at first, but recovered soon to the set point with the aid of feedback. For comparison, we show the performance of the pure learning algorithm Q-ILC in Figure 10 for the same disturbance. Since Q-ILC does not update the input in real time whereas the output had almost converged to the reference by the end of the sixth run, for the seventh run, the algorithm generated almost the same input trajectory as the one used in the sixth run. As a consequence, the disturbance was not handled at all resulting in a significant tracking error for a long period.

Starting with the results from the sixth run, this time we perturbed the initial reaction temperature by $-1°C$. As can be observed in Figure 11, BMPC increases the jacket temperature as soon as it detects the output deviation and returns the reactor temperature to the set point. Though not tested, Q-ILC, if had been applied, would have produced an output deviating from the reference trajectory by about $-1°C$ throughout the batch run since it lacks the real-time feedback correction capability.

Besides the above mentioned tests, we have tried other disturbances such as pulse-type heat inputs during the heatup stage. For every case we have tried, BMPC performed comparably to the above.

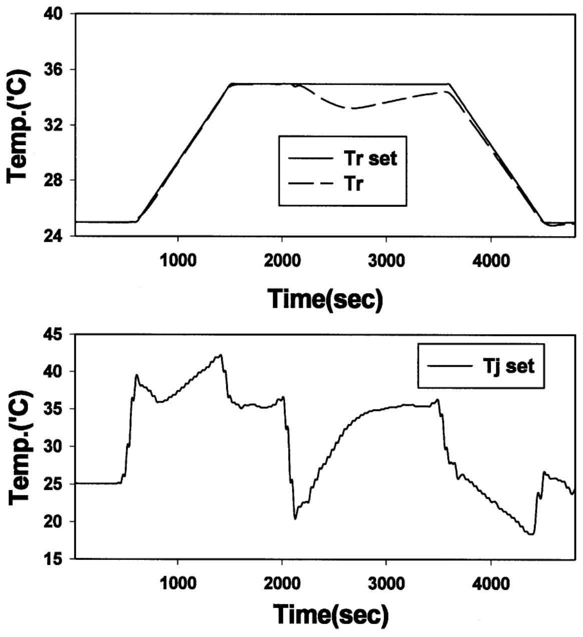

Figure 10. Performance of Q-ILC against a change in the reactant concentration ($C(0)$ is decreased from 1 to 0.85 (mol/L)).

Since BMPC is a model-based control technique, its performance depends on the model quality. However, the simple time-varying linear model adopted in this study was found to be sufficient for BMPC design for a highly nonlinear batch reactor system. In case that disturbances are not significant, the entire algorithm can be detuned either by increasing the input weight $\boldsymbol{R}$ or decreasing $\sigma_{w,1}^2$ and $\sigma_{w,2}^2$—in relation to $\boldsymbol{Q}$ and $\sigma_{v,1}^2$ and $\sigma_{v,2}^2$. In this way, larger model errors can be tolerated for convergence at the expense of the convergence rate. When the disturbances are significant and their rejections are important, the whole algorithm cannot be detuned in this manner. However, the flexibility of the BMPC algorithm allows the learning rate to be slowed down without affecting the real-time control performance. Note that the real-time control part of the algorithm does not distinguish between $\boldsymbol{w}$ and $\boldsymbol{v}$. After decreasing $\sigma_{w,1}^2$ and $\sigma_{w,2}^2$ to slow down the learning rate, one can simply increase $\sigma_{v,1}^2$ and $\sigma_{v,2}^2$ by same amount to bring up the sensitivity of real-time control.

Finally, if the batch length must be varied for whatever reason, the proposed algorithm cannot be applied as is. However, BMPC does not have to be applied over the entire batch horizon. In many applications, it will be sufficient to apply the technique only over a limited critical time zone, for example, from the final stage of heatup to the first stage of reaction, where the operation is required to proceed under a precise time schedule.

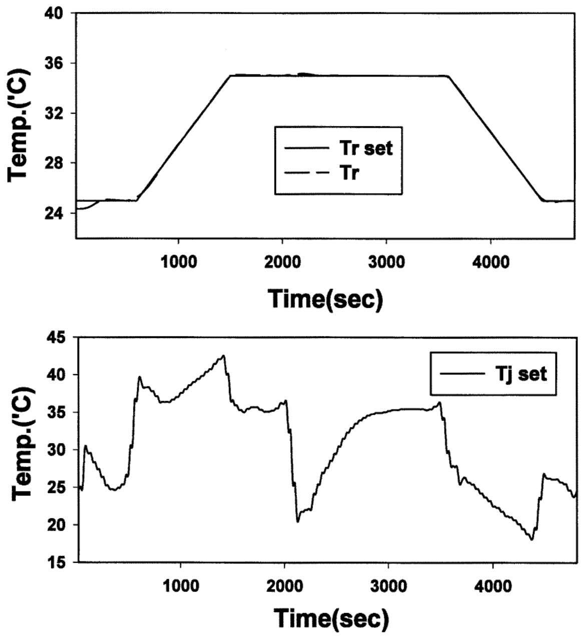

Figure 11. Performance of BMPC against a change in the initial reactor temperature ($T_r(0)$ is decreased by 1°C).

Conclusions

In this article, we presented a novel MPC technique for batch processes and applied it to a bench-scale batch reactor system where a highly exothermic reaction takes place. The technique—which we called BMPC—combines batch-wise iterative learning with real-time predictive control. The BMPC technique is based on a time-varying MIMO linear model representing the underlying nonlinear batch process and can attain asymptotically perfect tracking despite initialization errors (or constant disturbances) and model errors.

In the experimental study, the process was identified as a simple linear ARX model, where the coefficients were represented by pre-specified time-varying functions. The resulting model, though simplistic, was observed to yield satisfactory control performance. In addition, we discussed in detail how the tuning factors are to be determined on the basis of disturbance characteristics.

Although there still remain issues to resolve and improvements to be made, the presented BMPC technique should open some new opportunities (and challenges) for batch system modeling and control where standards are yet to be established.

Acknowledgment

The first author (KSL) would like to acknowledge the financial support from the Korea Science and Engineering Foundation (KOSEF) through the Automation Research Center at POSTECH and also the partial support from the LG Yonam Foundation (Korea). The second author (JHL) is grateful to the National Science Foundation's Young Investigator Program (USA) under Grant CTS No. 9357827 for the financial support.

Literature Cited

Amann, N., D. H. Owens, and E. Rogers, "Iterative Learning Control for Discrete-Time Systems with Exponential Rate of Convergence," *IEEE Proc.-Control Theory and Applications*, **143**, 217 (1996).

Arimoto, S., S. Kawamura, and F. Miyazaki, "Bettering Operation of Robots by Learning," *J. Robotic Syst.*, **1**, 123 (1984).

Åström, K. J., and B. Wittenmark, *Computer Controlled Systems*, Prentice Hall, Englewood Cliffs, NJ (1990).

Cabassud, M., M. V. Le Lann, A. Chamayou, and G. Casamatta, "Modeling and Adaptive Control of a Batch Reactor," *Dechema-Monographs*, **116**, 81 (1989).

Chen, Y., "A Bibliographical Library for ILC Research," http://www.ee.nus.sg/~ yangquan/ILC/ilcref.html (1998).

Cott, B. J., and S. Macchietto, "Temperature Control of Exothermic Batch Reactors using Generic Model Control," *Ind. Eng. Chem. Res*., **28**, 1177 (1989).

Fileti, A. M., and J. A. Pereira, "Adaptive and Predictive Control Strategies for Batch Distillation—Development and Experimental Testing," *Comput. Chem. Eng*., **21**(S), 1227 (1997).

Garcia, C. E., and A. M. Morshedi, "Quadratic Programming Solution of Dynamic Matrix Control," *Proc. of ACC*, San Diego, CA (1984).

Jarupintusophon, P., M. V. Lelann, M. Cabassud, and G. Casamatta, "Realistic Model-Based Predictive and Adaptive-Control of Batch Reactor," *Comput. Chem. Eng*., **18**(S), 445 (1994).

Juba, M. R., and J. W. Hamer, "Process and Challenges in Batch Process Control," *CPC-III*, Asilomar, CA, p. 139 (1986).

Katoh, N., K. Nakao, and H. Hanawa, "Learning Control of a Batch Reactor," *Comput. Chem. Eng*., **13**, 1273 (1988).

Kershenbaum, L. S., and P. Kittisupakorn, "An Experimental Testing of Model Predictive Control Algorithms Using a Partially Simulated Exothermic (PARSEX) Reactor," AIChE Meeting, San Francisco, CA (1994).

Kuc, T. Y., and J. S. Lee, "Learning Strictly Positive Real Linear-Systems with Uncertain Parameters and Unknown Input Disturbances," *AUTOMATICA*, **32**, 791 (1996).

Lee, J. H., and A. K. Datta, "Nonlinear Inferential Control of Pulp Digesters," *AIChE J.*, **40**, 50 (1994).

Lee, J. H., and M. Morari, "State-Space Interpretation of Model Predictive Control," *Automatica*, **30**, 707 (1994).

Lee, K. S., S. H. Bang, and K. S. Chang, "Feedback-Assisted Iterative Learning Control Based on an Inverse Process Model," *J. Proc. Cont*., **4**, 77 (1994).

Lee, K. S., and J. H. Lee, "Model-Based Refinement of Input Trajectories for Batch and Other Transient Processes," AIChE Meeting, Chicago (1996). (Also *Automatica*, in press, 1999, http://procd.sogang.ac.kr/pub.html#jint).

Lee, P. L., and G. R. Sullivan, "Generic Model Control," *Comput. Chem. Eng*., **12**, 573 (1988).

Marroquin, G., and W. L. Luyben, "Experimental Evaluation of Nonlinear Cascade Controllers for Batch Reactor," *Ind. Eng. Chem. Fund*., **II**, 552 (1972).

Morari, M., and J. H. Lee, "Model Predictive Control: Past, Present and Future," *PSE/ESCAPE*, Trondheim, Norway (1997). Also *Comput. Chem. Eng*., in press, 1999).

Nagy, Z., and S. Agachi, "Model-Predictive Control of a PVC Batch Reactor," *Comput. Chem. Eng*., **21**(6), 571 (1997).

Russell, S., P. Kesavan, J. H. Lee, and B. Ogunnaike, "Recursive Data-Based Prediction and Control of Product Quality for Batch and Semi-Batch Processes Applied to Nylon 6,6 Autoclave," *AIChE J.*, **44** (Nov., 1998).

Söderström, T., and P. Stoica, *System Identification*, Prentice Hall, Englewood Cliffs, NJ (1989).

Soroush, M., and C. Kravaris, "Discrete-Time Nonlinear Controller Synthesis by Input-Output Linearization," *AIChE J.*, **38**, 1923 (1992).

Wang, Z. L., F. Pla, and J. P. Corriou, "Nonlinear Adaptive-Control of Batch Styrene Polymerization," *Chem. Eng. Sci*., **50**(13), 2081 (1995).

Zafiriou, E., and H. W. Chiou, "Output Constraint Softening for SISO Model Predictive Control," *Proc. of ACC*., 372 (1993).

Zafiriou, E. R., A. Adomaitis, and G. Gattu, "Approach to Run-to-Run Control for Rapid Thermal Processing," *Proc. of ACC*., Seattle, 1286 (1995).

Zheng, A., and M. Morari, "Stability of Model Predictive Control with Mixed Constraints," *IEEE Trans. A. C*., **40**, 1818 (1995).

Manuscript received June 10, 1998, and revision received Mar. 23, 1999.