Robust Model Predictive Control and Fault Handling of Batch Processes

Siam Aumi and Prashant Mhaskar
Dept. of Chemical Engineering, McMaster University, Hamilton, ON, Canada, L8S 4L7

DOI 10.1002/aic.12398
Published online September 20, 2010 in Wiley Online Library (wileyonlinelibrary.com).

This work considers the control of batch processes subject to input constraints and model uncertainty with the objective of achieving a desired product quality. First, a computationally efficient nonlinear robust Model Predictive Control (MPC) is designed. The robust MPC scheme uses robust reverse-time reachability regions (RTRRs), which we define as the set of process states that can be driven to a desired neighborhood of the target end-point subject to input constraints and model uncertainty. A multilevel optimization-based algorithm to generate robust RTRRs for specified uncertainty bounds is presented. We then consider the problem of uncertain batch processes subject to finite duration faults in the control actuators. Using the robust RTRR-based MPC as the main tool, a robust safe-steering framework is developed to address the problem of how to operate the functioning inputs during the fault repair period to ensure that the desired end-point neighborhood can be reached upon recovery of the full control effort. The applicability of the proposed robust RTRR-based controller and safe-steering framework subject to limited availability of measurements and sensor noise are illustrated using a fed-batch reactor system. © 2010 American Institute of Chemical Engineers *AIChE J*, 57: 1796–1808, 2011

Keywords: control, process control, reachability regions, batch process, control, model predictive control

Introduction

Batch and fed-batch processes have widespread applications within a variety of sectors especially those that are used for the manufacture of high-quality low-volume products such as biochemicals and polymers. The primary control objective in batch processes is to reach a specified product quality by batch termination. Consequently, batch control strategies are typically designed to reach a desired end-point and can be categorized into trajectory tracking approaches or end-point-based approaches. In trajectory tracking approaches, optimal state trajectories that terminate at the desired end-point are generated off-line by solving a dynamic optimization problem[1] or using iterative dynamic programing,[2] and subsequently tracked on-line using local PID or predictive controllers.[3–6] A drawback in these approaches is that process noise and disturbances can render optimal state trajectories suboptimal or in the worst case, the tracking problem infeasible.

With increased quality of computational resources and more efficient optimization algorithms, shrinking horizon end-point-based model predictive control (MPC) is becoming a possibility for batch process control. In computing the control action in this approach, a dynamic optimization problem that incorporates the desired end-point (in the objective function and/or constraints) is solved at each sampling instance until batch termination, using plant measurements (or state variable estimates) to update model predictions at each sampling time. A truncated version of the resulting optimal input changes is then implemented directly on the process, the prediction horizon is shifted one sampling instance, and the

Correspondence concerning this article should be addressed to P. Mhaskar at mhaskar@mcmaster.ca.

process is repeated at the next sampling instance. One drawback of end-point-based MPC approaches is the significant computational demand associated with solving the dynamic optimization problem at each sampling instance. More specifically, the solution of the dynamic optimization problem consists of the entire input trajectory from the time at which the problem is solved to batch termination, implying significant computational demands especially during the early stages of the batch. Additionally, with model uncertainty, significant discrepancies can result between the predicted and actual behavior of a process. For instance, a predictive model integrated with nominal values of the uncertainty can indicate the process will be driven to the desired end-point for a specific control move, but applying the identical control move on the actual process can lead to a violation of product end-use properties and/or safety constraints because of inaccurate parameter values. Therefore, incorporating model uncertainty in the control scheme either by modifying the conventional dynamic optimization problem[7–9] or by reducing biases in state estimates arising from model uncertainty is essential[10–13] for obtaining acceptable controller performance in the presence of uncertainty.

Although several end-point-based MPC formulations that explicitly account for model uncertainty using a min-max optimization framework are available,[8,9] these approaches are often more computationally prohibitive. This is because the control moves are computed by taking into account the worst-case values of the uncertainty, which are computed using an embedded optimization problem within the dynamic optimization. Techniques to reduce the real-time optimization computational demands include those that attempt to reduce the complexity of the model used for predictions and those in which a parametrization of an open-loop optimal input profile is derived off-line. With respect to the former, some of these approaches include successive linearization of the nonlinear model equations at each sampling instance and the scheduling of multiple linear models (see Ref. 9 for a review). However, with significant improvements in real-time optimization algorithms and performance limitations associated with using linear MPC for highly nonlinear batch processes, MPC using the full nonlinear model is becoming increasingly common.[14–17] Input parametrization techniques,[18–20] on the other hand, strive to reduce the number of decision variables in the dynamic optimization problem. However, with modeling errors and process noise, plant process trajectories can deviate considerably from the nominal optimal trajectories on which the input parametrization is based, rendering a certain input parametrization suboptimal or infeasible.

The variability in the raw material availability adds another layer of complexity to batch process control and motivates designing methods for determining the suitability of running a batch with the given raw material. In particular, for a given control law, it is important to ascertain the initial conditions (without running the batch in its entirety) for which the desired control objectives are obtainable to minimize resource and time wastage. Although there exist MPC designs for continuous systems that allow the explicit characterization of the set of initial conditions from where stability is achievable,[21–23] these results are not applicable for batch systems because the desired end-point is usually not an equilibrium point. Currently, there exist no end-point-based MPC designs for batch systems that provide an explicit characterization of a feasibility region from where it can be guaranteed that the desired control objectives can be met. Recently, in Ref. 24, a computationally efficient, nonlinear MPC design based on the concept of "reverse-time reachability regions" (RTRRs) was presented. RTRRs were defined as the set of states from where the desired end-point can be reached by batch termination. In contrast to existing results, the MPC formulation in Ref. 24 provides an explicit characterization of the set of initial conditions from where a desired end-point is achievable. The reachability guarantees provided by the RTRR-based controller, however, do not hold in the presence of model uncertainty owing to the significant impact of the presence of uncertainty on RTRRs. In particular, a direct application of RTRR-based MPC of Ref. 24 could very likely result in off-spec product and a wasted batch. One contribution of this work is the redesign of the RTRR-based predictive controller to account for model uncertainty by explicitly incorporating uncertainty bounds in the generation of RTRRs and their subsequent use within a robust MPC formulation.

Faults in processing or auxiliary equipment are ubiquitous in the chemical process industry and are an aspect of plant operation that requires special attention in the control design. The emphasis on the final product quality in batch and fed-batch processes makes their productivity particularly susceptible to faults as a fault can invalidate the desirable properties of a control design. Although there has been significant work on fault detection and isolation for batch processes,[25–29] fault-tolerant control structures (FTCS) specific to batch systems have received limited research attention. The majority of the extensive research on FTCS for continuous processes[30] including recent results that address how to operate a plant during temporary faults[31] cannot be applied to batch processes because of the absence of equilibrium points and fundamental differences in the control objectives between batch and continuous processes.[4]

As is the case for continuous processes, the majority of the FTCS for batch processes[8,32] essentially rely on the robustness of the control design to handle faults as disturbances during the failure period. The fault-tolerant characteristic in these formulations stems from the underlying assumption of availability of sufficient control effort such that the primary control objective remains achievable even in the presence of the fault. However, processes often encounter faults where the nominal control objective cannot be achieved if the fault persists, and furthermore, in the absence of a framework for explicitly handling such faults in batch processes, continuation of the implementation of controllers with limited fault-tolerant properties could lead to a missed opportunity to implement control action that could enable achieving the primary control objective after fault repair. A desirable property in a framework for handling faults in the context of batch systems, therefore, would be one that can identify an input trajectory (if it exists) without requiring any prior knowledge of the fault repair time to ensure end-point reachability upon fault repair. A safe-steering framework was recently developed in Ref. 24 that addressed the problem of determining how to use functioning inputs during fault rectification to enable desired product properties reachability following fault repair. The safe-steering framework in Ref. 24, however, does not account for the presence of uncertainty, and neither the controllability nor the safe-steering guarantees of Ref. 24 remain applicable in the presence of uncertainty.

Motivated by the above considerations, this work considers the problem of designing computationally efficient nonlinear MPC for batch processes subject to input constraints, faults in the control actuators, and model uncertainty. Specifically, faults are considered that cannot be handled via robust control approaches and (if not rectified) preclude the reachability to the desired end-point with limited control effort. The rest of this manuscript is organized as follows: First, the class of processes considered is presented followed by reviews of a conventional end-point-based MPC formulation, the RTRR-based predictive controller developed in Ref. 24, and the safe-steering framework also developed in Ref. 24. Next, we generalize the RTRR-based predictive controller to explicitly account for model uncertainty by first developing a methodology and algorithm to generate robust RTRRs for their subsequent use within a robust MPC design for batch processes. Then, after formulating the safe-steering problem, a safe-steering framework is developed that uses the robust RTRR-based MPC to ensure states can be driven inside a desired end-point neighborhood if the fault is repaired sufficiently fast. It is shown that the proposed safe-steering framework is guaranteed to find a viable input trajectory if it exists (i.e, if the fault is repaired sufficiently fast) that can drive the system to a desired end-point neighborhood following fault repair. Then, closed-loop simulation results of a fed-batch reactor subject to actuator failure, model uncertainty, limited availability of measurements, and sensor noise are presented to illustrate the efficacy of the proposed robust MPC formulation and the details of the safe-steering framework. Finally, we summarize our results.

Preliminaries

In this section, we describe the class of batch processes considered and review a representative end-point-based nonlinear, shrinking horizon predictive controller, the RTRR-based predictive controller of Ref. 24, and the safe-steering framework developed in Ref. 24 without accounting for uncertainty.

Process description

We consider batch process systems with uncertainty subject to input constraints and failures described by:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}_\sigma(t), \theta(t)) \qquad t \in [t_0, t_\mathrm{f}], \mathbf{u}_\sigma(\cdot) \in \mathbf{U}_\sigma, \theta(\cdot) \in \Theta, \mathbf{x}(t_0) = \mathbf{x}_0, \quad (1)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the vector of state variables, $\mathbf{u}_\sigma(t) \in \mathbb{R}^m$ denotes the vector of constrained manipulated inputs, taking values in a nonempty convex subset $\mathbf{U}_\sigma$ of $\mathbb{R}^m$, where $\mathbf{U}_\sigma = \{\mathbf{u} \in \mathbb{R}^m | \mathbf{u}_{\min,\sigma} \le \mathbf{u} \le \mathbf{u}_{\max,\sigma}\}$, where $\mathbf{u}_{\min,\sigma}, \mathbf{u}_{\max,\sigma} \in \mathbb{R}^m$ denote the constraints on the manipulated inputs, and $\theta(t) = [\theta^1(t) \cdots \theta^q(t)]^T \in \Theta \subset \mathbb{R}^q$ where $\Theta = \{\theta \in \mathbb{R}^q | \theta_{\min} \le \theta \le \theta_{\max}\}$, where $\theta_{\min}, \theta_{\max} \in \mathbb{R}^q$ denote the bounds on the vector of uncertain (possibly time–varying) but bounded variables taking values in a nonempty compact convex subset of $\mathbb{R}^q$. The times t_0 and t_f denote the initial time and batch termination times, respectively. The variable, $\sigma \in \{1,2\}$, is a discrete variable that indexes fault-free and faulty operation with $\sigma = 1$ signifying fault-free operation and $\sigma = 2$ signifying faulty operation. The fault scenarios considered in this work involve actuator failure for a finite duration of time defined by the time of fault occurrence, t^{fault}, and time of fault repair, t^{repair}. An example of such a failure scenario is a flow valve feeding a fed-batch reactor becoming ''stuck'' at its fail-safe value between t^{fault} and t^{repair} while remaining operational at all other times. According to the definition of σ, $\sigma = 1$ for $t \in [t_0, t^{\text{fault}})$, $\sigma = 2$ for $t \in [t^{\text{fault}}, t^{\text{repair}})$, and $\sigma = 1$ for $t \in [t^{\text{repair}}, t_f]$. The vector function $\mathbf{f} : \mathbb{R}^n \times \mathbf{U}_\sigma \times \Theta \to \mathbb{R}^n$ is assumed to be continuous on $(\mathbf{x},\mathbf{u},\boldsymbol{\theta})$ and locally Lipschitz in $\mathbf{x}$ on $\mathcal{D} \times \mathbf{U} \times \Theta$, where $\mathcal{D} \subset \mathbb{R}^n$. The notation, $\|\cdot\|_\mathbf{Q}$, refers to the weighted norm, defined by $\|\mathbf{x}\|_\mathbf{Q} = \mathbf{x}^T\mathbf{Q}\mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{Q}$ is a positive-definite symmetric matrix and $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$. Throughout the manuscript, we assume that for any $\mathbf{u}(\cdot) \in \mathbf{U}_\sigma$ and $\boldsymbol{\theta}(\cdot) \in \Theta$, the solution of the batch system of Eq. 1 exists and is continuous for all $t \in [t_0, t_\mathrm{f}]$, and we focus on the state feedback problem, i.e., $\mathbf{x}(t)$ is assumed to be available for all $t \in [t_0, t_\mathrm{f}]$.

End-point based MPC

In this section, a representative formulation of a shrinking horizon, nonlinear predictive controller for batch processes is outlined to convey the key idea in most existing formulations, which is the computation of the entire input trajectory from the time the control action is being calculated to the end of the batch. To this end, consider a batch process described in Eq. 1 under an end-point-based predictive controller for fault-free conditions (i.e., $\sigma(t) = 1$). The control action at each sampling instance is computed by solving (on-line) a dynamic optimization problem of the form:

$$\min_{\mathbf{u}(\cdot)} J_E = M(t_0, \tilde{\mathbf{x}}(t_0), t_f, \tilde{\mathbf{x}}(t_f), \mathbf{x}_\mathrm{d}) + \int_t^{t_f} L(\tilde{\mathbf{x}}(\tau), \mathbf{u}(\tau))\, d\tau \quad (2)$$

$$\text{subject to : } \dot{\tilde{\mathbf{x}}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \mathbf{u}(\tau)) \quad (3)$$

$$\tilde{\mathbf{x}}(0) = \mathbf{x}(t) \quad (4)$$

$$\tilde{\mathbf{x}}(t_f - t) = \mathbf{x}_\mathrm{d}, \quad (5)$$

where $M(\cdot)$ and $L(\cdot)$ represent the Mayer and Lagrangian terms, respectively. The constraint, $\tilde{\mathbf{x}}(0) = \mathbf{x}(t)$ (Eq. 4), represents the initialization of the dynamic optimization problem at the current process conditions and can be understood as the feedback mechanism in the controller to account for plant-model mismatch. In the terminal constraint (Eq. 5), $\mathbf{x}_\mathrm{d}$ denotes the specified desired states at $t = t_\mathrm{f}$. The minimizing control $\mathbf{u}(\cdot)$ is then directly implemented on the plant over the interval $[t, t+\delta)$ and the procedure is repeated until batch termination.

Remark 1. The evaluation of the objective function (Eq. 2) and terminal constraint (Eq. 5) necessitates the integration of the nonlinear model equations and optimization of the manipulated inputs up to t_f at each sampling instance. Thus, the dynamic optimization problem becomes computationally expensive regardless of the optimization strategy (sequential or simultaneous). Note that in the absence of uncertainty, the solution to the optimization problem is only required at the first time step because the solution at the *j*-th time step is simply the initial solution trajectory from $(j + 1)\delta$ to t_f where δ is the sampling period. With the presence of model uncertainty, the "tail" of the initial solution is no longer a solution for subsequent optimization problems. Although the solution at a certain time step can serve as a good initial guess for the next time step, significant computation time may still be required to arrive at a solution in the presence of uncertainty.

Remark 2. Although we present a "nominal" MPC formulation to emphasize the fact that the control calculation at every instance requires the solution of the entire input trajectory, a min-max dynamic optimization problem can in principle be used to handle the problem of uncertainty. The solution to such a min-max problem would be a manipulated input trajectory from the current time to the end of the batch that minimizes (using the inputs as decision variables) the maximum (over all realizations of the uncertainty) value of the objective function. This added layer of optimization renders min-max-based MPC approaches for batch systems even more computationally expensive than the nominal end-point-based MPC formulation in Eqs. 2–5 and motivates the development of a computationally efficient robust nonlinear MPC formulation for batch processes.

Reverse-time reachability region–based MPC

RTRRs and the RTRR-based MPC strategy developed in Ref. 24 are briefly reviewed in this section. The notion of RTRRs was introduced in Ref. 24 for the purposes of designing an MPC scheme that only required the computation of the immediate value of the input while ensuring the desired end-point remains attainable throughout the batch.

Reverse-Time Reachability Regions. In Ref. 24, the RTRR of a batch system at time, t, $\mathcal{R}(t)$, was defined as the set of all process states from where a batch process could be steered to $\mathbf{x}_d$, the desired end-point, by the end of the batch (i.e., in a time $t_f - t$) while satisfying the input constraints. The formal definition of the discrete time version of this set is reproduced below:

Definition 1. (Ref. 24) For the batch process described in Eq. 1 with sampling period δ and no model uncertainty, the RTRR at time $t = t_f - k\delta$, indexed by k, is the set:

$$\begin{aligned}\mathcal{R}_k = \{\mathbf{x}_0 | \mathbf{x}(t) = \mathbf{x}_0, \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}(t))\\ \exists\, \mathbf{u}(t) = \{\mathbf{u}[i]\} \in \mathbf{U}\ \forall i = 1, \ldots, k \text{ where}\\ \mathbf{u}[i] = \mathbf{u}(i\delta) \text{ and satisfies } \mathbf{u}(t) = \mathbf{u}[i]\\ \forall t \in [i\delta, (i+1)\delta), \text{ such that } \mathbf{x}(t_f) = \mathbf{x}_d\}. \quad (6)\end{aligned}$$

Remark 3. An algorithm to generate estimates of RTRRs as point-sets at each sampling instance for the batch was proposed in Ref. 24 that consisted entirely of successive off-line integrations of the reverse-time model of the system (i.e., $\dot{\mathbf{x}}(t) = -\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$). Specifically, for a given $\mathbf{x}_d$, the reverse-time process model can be integrated for the duration of δ, holding the value of the manipulated inputs constant over δ. Performing this integration for all possible appropriately discretized values of the manipulated inputs in turn yields an (under) estimate of $\mathcal{R}_1$ as a point-set. A point-set estimate of $\mathcal{R}_2$ can then be determined by repeating the process for all elements in $\mathcal{R}_1$, and the process repeated to yield RTRRs up to the initial time. The details regarding alleviating the associated computational costs with this algorithm, assumptions to ensure the generation of compact sets, and characterization strategies for the point-set estimates are addressed in Ref. 24. Also, note that the presence of uncertainty cannot be accounted for by simply repeating the above set of calculations for all allowable values of the uncertainty and requires redesigning the RTRR generation algorithm.

Reverse-time Reachability Region Based MPC Formulation. Once RTRRs have been computed for every sampling instance of the batch, they can be subsequently used in a RTRR-based MPC design. To this end, consider the process described in Eq. 1 for fault-free conditions (i.e., $\sigma(t) = 1$) with no model uncertainty, where the control action at sampling instance $k = \frac{t_f - t}{\delta}$ is computed by solving the following dynamic optimization problem:

$$\min_{\mathbf{u}} J_{\mathcal{R}} \quad (7)$$

$$\text{subject to}: \dot{\tilde{\mathbf{x}}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \mathbf{u}) \quad (8)$$

$$\tilde{\mathbf{x}}(0) = \mathbf{x}(t) \quad (9)$$

$$\tilde{\mathbf{x}}(\delta) \in \mathcal{R}_{k-1}. \quad (10)$$

The key idea in the RTRR-based MPC formulation is to successively maintain states in RTRRs until batch termination. The existence of the process states in the RTRR at each sampling instance during the batch was also established as a necessary and sufficient condition for states to be steered to the desired end-point (the theorem and proof are available in Ref. 24). The necessity of this condition has a particularly important implication because if at any point, the process states are driven outside of the RTRR, it is simply not possible, regardless of the control law, to drive the states back into a RTRR and ultimately to the desired end-point by the batch termination time. Note that in the RTRR formulation, the terminal constraint (Eq. 5) in the end-point-based MPC scheme has been replaced with a reachability constraint (Eq. 10) that requires, at each sampling instance, the process states to remain within the RTRR at the next time step. Therefore, state evaluation and input optimization are not necessary beyond the next time step, considerably reducing the size and computational cost of the dynamic optimization problem compared with conventional end-point-based MPC approaches.

Remark 4. An important advantage of the RTRR-based predictive controller over conventional end-point-based MPC approaches is its computational efficiency while guaranteeing end-point reachability even when using the full nonlinear model of the system. In the presence of uncertainty, however, it becomes important to explicitly account for the uncertainty in the RTRR-based MPC design from not only a theoretical standpoint but also from a practical standpoint. In particular, ignoring the effect of the uncertainty will lead to larger estimates of the RTRRs; therefore, if the initial conditions are within the RTRR at t_0, initial feasibility of the optimization problem can still be achieved even in the presence of uncertainty. However, this initial feasibility does not imply successive feasibility for all future time steps as large deviations in the uncertain parameters can drive the states out of the RTRR estimates at which point the control law would become infeasible. This, in turn, of course invalidates the reachability guarantees of the MPC design. Even worse, a possible scenario is that of the RTRR MPC laws remaining feasible by virtue of the large region estimates, but the uncertainty driving the process states significantly away from the desired end-point at the last time step, resulting in a wasted batch.

Safe-steering of batch processes

In Ref. 24, a fault-tolerant control framework, called the "safe-steering" framework, was designed for batch processes with finite duration control actuator failure that precluded meeting the desired end-point if not rectified. Specifically, faults were considered where a control actuator fails and reverts to its fail-safe value at time t^{fault} and is eventually repaired at an unknown later time $t^{\text{repair}} < t_{\text{f}}$. The safe-steering problem was defined as the one of identifying trajectories of the functioning inputs during the failure period such that the desired end-point remains reachable following fault repair (i.e., following full recovery of the control effort).

In Ref. 24, the necessary and sufficient conditions, namely feasibility of the RTRR-based MPC during the failure period, for safe-steering a batch were established. If the RTRR-based MPC is feasible from t^{fault} up to and including t^{repair}, the implication is that at t^{repair}, the states are contained with the RTRR at t^{repair} at which point the full control effort is recovered. Consequently, by the definition of RTRRs, the desired end-point will be reachable following t^{repair}. In the presence of uncertainty, however, the end-point reachability guarantees in the safe-steering framework are lost even if the process states at t^{repair} are contained within the RTRR at t^{repair}. This is because RTRRs as defined in Ref. 24 are generated-based on nominal values of the uncertainty. Consequently, using the RTRR-based MPC formulation to preserve states in "nominal" RTRRs during the failure period in the presence of uncertainty has no implications regarding end-point reachability and is therefore not a suitable solution to the safe-steering problem. This motivates the development of a robust MPC design for use in the safe-steering framework such that the desired control objectives in the batch can be met if the fault is recovered sufficiently fast.

Robust Reverse-Time Reachability Region-Based MPC

In this section, we present a computationally efficient robust nonlinear predictive controller for batch processes. Preparatory to the robust controller formulation, we introduce the notion of robust RTRRs, which are essential in the control design and analysis, and then present an algorithm to generate these regions. This is followed by the formulation of a robust RTRR-based MPC design.

Robust reverse-time reachability regions

To define robust RTRRs, we first note that in batch process control, the implication of the presence of model uncertainty is that in general, exact end-point reachability guarantees cannot be made regardless of the control law. Instead, only reachability to a certain neighborhood of the end-point can be guaranteed. To understand this, consider the batch system described in Eq. 1 subject to a predictive control law. At time $t = t_{\text{f}} - \delta$, a dynamic optimization problem is solved to compute the set of input moves that drives the system to the desired end-point in time δ for nominal values of the uncertainty, $\boldsymbol{\theta}_{\text{nom}}$. However, if this same control move is implemented on the plant, there is no guarantee that the process will be driven to $\mathbf{x}_{\text{d}}$ because it is unknown if $\boldsymbol{\theta}_{\text{nom}}$ represents the true values of the plant parameters. Based on this argument, one desirable property of a robust MPC formulation for batch process is to guarantee the existence of inputs to drive the process states inside a desired neighborhood around the end-point, which we denote by $\mathcal{B}(\mathbf{x}_{\text{d}})$, that can be chosen-based on the acceptable level of variance in the final product quality. Accordingly, of interest is the set of process states from where a desired end-point neighborhood can be reached in the presence of model uncertainty while satisfying all input constraints. These sets are denoted as robust RTRRs and are formally defined below:

Definition 2. For the process described in Eq. 1, the robust RTRR at time, t, is the set:

$$\begin{aligned}\tilde{\mathcal{R}}(t) = \{\mathbf{x}_0 | \mathbf{x}(t) = \mathbf{x}_0, \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}(t))\\ \exists \mathbf{u}(\tau) \in \mathbf{U}\ \forall \tau \in [t, t_f], \text{such that}\\ \forall \boldsymbol{\theta}(\tau) \in \boldsymbol{\Theta}\ \forall \tau \in [t, t_{\text{f}}], \mathbf{x}(t_{\text{f}}) \in \mathcal{B}(\mathbf{x}_{\text{d}})\}.\end{aligned} \tag{11}$$

To account for the discrete nature of control implementation, we define the discrete time version of robust RTRRs below:

Definition 3. For the process described in Eq. 1, the robust RTRR at a time $t = t_{\text{f}} - k\delta$, indexed by k is the set:

$$\begin{aligned}\tilde{\mathcal{R}}_k = \{\mathbf{x}_0 | \mathbf{x}(t) = \mathbf{x}_0, \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}(t))\\ \exists\ \mathbf{u}(t) = \{\mathbf{u}[i]\} \in \mathbf{U}\ \forall i = 1, \ldots, k\\ \text{where } \mathbf{u}[i] = \mathbf{u}(i\delta) \text{ and satisfies } \mathbf{u}(t) = \mathbf{u}[i]\\ \forall t \in [i\delta, (i+1)\delta) \text{ such that}\\ \forall \boldsymbol{\theta}(\tau) \in \boldsymbol{\Theta}\ \forall \tau \in [t, t_f], \mathbf{x}(t_f) \in \mathcal{B}(\mathbf{x}_{\text{d}})\}.\end{aligned} \tag{12}$$

Note that for the special case of $k = 0$, $\tilde{\mathcal{R}}_k$ is defined to be the desired end-point neighborhood or $\mathcal{B}(\mathbf{x}_{\text{d}})$. Before presenting an algorithm to generate robust RTRRs, the existence of these regions must first be established. This is formalized below in Theorem 1. The proof is available in the Appendix.

Theorem 1. *For the batch system described in Eq.* 1, *given* $\boldsymbol{\Theta}$ *and a nonempty* $\tilde{\mathcal{R}}_{k-1}$, *there exists a* δ^* *such that for any* $\delta \leq \delta^*$, $\tilde{\mathcal{R}}_k \neq \emptyset$ *(i.e,* $\tilde{\mathcal{R}}_k$ *is non-empty).*

Remark 5. In the absence of model uncertainty, the existence of nonempty RTRRs at each sampling instance of the batch is guaranteed simply from the existence of a solution for the system over a finite time. When considering uncertainty, however, the size of robust RTRRs depends on the size of the desired end-point neighborhood and also the sampling period in the batch. For example, if we consider a fixed $\mathcal{B}(\mathbf{x}_d)$, robust RTRRs may cease to exist as we proceed toward the initial time if the given sampling time is too large. Theorem 1 is therefore important in establishing the trade-off between $\mathcal{B}(\mathbf{x}_d)$ and the sampling period. From a practical perspective, for a specified desired end-point neighborhood, the result of Theorem 1 implies that the sampling time can be used to mitigate the reduction in the size of robust RTRRs as we proceed toward the initial time.

Generating Robust Reverse-Time Reachability Regions. In this section, given a desired end-point neighborhood, $\mathcal{B}(\mathbf{x}_d)$, we develop a methodology to sequentially generate robust RTRR estimates off-line. More specifically, starting at $k = 1$, for a given $\mathcal{B}(\mathbf{x}_d)$, we identify an explicitly characterized estimate of $\tilde{\mathcal{R}}_k$ from where the process can be driven inside $\tilde{\mathcal{R}}_{k-1}$ in the presence of model uncertainty. The explicit characterization is also a necessity for the practical implementation of the MPC formulation to be presented in the next section. In this work, we use n-dimensional ellipsoids to mathematically express estimates of robust RTRRs at each sampling instance (note that our results are not limited to this choice of the characterization; the use of n-dimensional ellipsoids is simply to illustrate our results). The general form of the ellipsoid expression is given below:

$$\tilde{\mathcal{R}}_k = \{\mathbf{x} \mid ||\mathbf{x} - \tilde{\mathbf{c}}_k||_{\tilde{\mathbf{P}}_k} \leq 1\}, \qquad (13)$$

where the vector $\tilde{\mathbf{c}}_k \in \mathbb{R}^n$ denotes the center of the ellipsoid, the positive-definite symmetric matrix $\tilde{\mathbf{P}}_k \in \mathbb{R}^n \times \mathbb{R}^n$ defines its size and orientation, and k indexes the batch sampling instances as before. Note that because $k = 0$ corresponds to t_f, $\tilde{\mathbf{c}}_0 = \mathbf{x}_d$ and $\tilde{\mathbf{P}}_0$ is a user-defined matrix based on the acceptable variance level of the final product quality.

To determine if a set is a valid estimate for $\tilde{\mathcal{R}}_k$ (defined by $\tilde{\mathbf{c}}_k$ and $\tilde{\mathbf{P}}_k$), we solve, for a given δ and Θ (and an $\tilde{\mathcal{R}}_{k-1}$ defined by $\tilde{\mathbf{c}}_{k-1}$ and $\tilde{\mathbf{P}}_{k-1}$), the following multilevel nonlinear program (NLP):

$$\min_{\mathbf{x}_0} J_1 = 0 \qquad (14)$$

$$\text{subject to} : ||\mathbf{x}_0 - \tilde{\mathbf{c}}_k||_{\tilde{\mathbf{P}}_k} \leq 1 \qquad (15)$$

$$J_2 \geq 1 \qquad (16)$$

$$\min_{\mathbf{u}\in\mathbf{U}} J_2 = ||\tilde{\mathbf{x}}(\delta) - \tilde{\mathbf{c}}_{k-1}||_{\tilde{\mathbf{P}}_{k-1}} \qquad (17)$$

$$\text{subject to} : \dot{\tilde{\mathbf{x}}}(t) = \mathbf{f}(\tilde{\mathbf{x}}(t), \mathbf{u}, \boldsymbol{\theta}) \qquad (18)$$

$$\max_{\boldsymbol{\theta}\in\Theta} J_3 = ||\tilde{\mathbf{x}}(\delta) - \tilde{\mathbf{c}}_{k-1}||_{\tilde{\mathbf{P}}_{k-1}} \qquad (19)$$

$$\text{subject to} : \dot{\tilde{\mathbf{x}}}(t) = \mathbf{f}(\tilde{\mathbf{x}}(t), \mathbf{u}, \boldsymbol{\theta}). \qquad (20)$$

If the multi-level NLP is infeasible, we deem the estimate to be a valid robust RTRR. To understand this, consider the different levels in the multi-level NLP: for a given initial state, the two bottom most layers solve the (min-max) robust control problem, i.e., determine the input that minimizes the worst case effect of the uncertainty. In other words, the bottom two levels compute the control action that, for the worst realization of the uncertainty, drives the state to the lowest level set of the n -dimensional ellipsoid at the next time step. The top level problem then searches over all initial conditions within the given $\tilde{\mathcal{R}}_k$ to find (if it exists) an initial condition for which the state at the next time step ends up being driven outside the robust RTRR at the next step. If the multilevel optimization problem is feasible, it implies that there are no guarantees that the process starting from the given estimate of the robust RTRR will be driven inside the robust RTRR at the next step in the presence of uncertainty even when implementing the robust control move. On the other hand, if the problem is infeasible, this implies that for every initial condition in the given robust RTRR, the process states are always contained within the robust RTRR at the next time, even in the worst case effect of the uncertainty. An infeasible solution therefore represents that a valid robust RTRR has been found.

Remark 6. Note that in principle, one can add another layer in the optimization problem wherein the $\tilde{\mathbf{P}}_k$ matrix and the $\tilde{\mathbf{c}}_k$ vector are the decision variables and the objective is to maximize the "volume" of the n -dimensional ellipsoid. Even if carried out off-line, the determination of the largest robust RTRR would become an unwieldy problem. In this work, we address this problem by appropriately preselecting the $\tilde{\mathbf{P}}_k$ matrix (that determines the orientation of the ellipsoid) and the $\tilde{\mathbf{c}}_k$ vector (that determines the center of the ellipsoid). In particular, the system is reverse-time integrated from all the elements in $\tilde{\mathcal{R}}_{k-1}$ (using nominal values of the uncertainty and all possible values of the inputs), and then an ellipsoid is found that best covers a subset of these points. If the multilevel optimization problem is feasible for this ellipsoid, the set is scaled down and the problem is resolved until the multilevel optimization becomes infeasible. On the other hand, if the problem is infeasible to begin with, the set is scaled up and this process is repeated until the optimization problem becomes feasible. The final ellipsoid obtained through this procedure then represents the (approximately) largest estimate of the robust RTRR (given a predecided orientation of the ellipsoid).

Remark 7. Note that the problem of determining the robust RTRRs cannot be addressed by extending the method of generating the RTRRs in Ref. 24 to generate point sets using all values of the uncertainty (over and above the different values of the inputs as done in Ref. 24). The only conclusion that can be drawn for a point in such a set is that there exists a pair of input and uncertainty values such that the process can be driven to the RTRR at the next time. No guarantees can be made for the existence of an input (for any allowable value of the uncertainty) and the determination of that input (without knowing the value of the uncertain parameter) that can drive the process to the robust RTRR at the next step. This necessitates the development of the multilevel optimization-based framework for the generation of the robust RTRRs. Note also that the objective of this work is not to characterize the true robust RTRR (that is to determine all points that are contained within the robust RTRR) but to generate a workable estimate for which the existence and determination of the input to drive the process to the next robust RTRR can be guaranteed.

Robust MPC formulation

In this section, we present an MPC formulation that uses robust RTRRs to steer a batch system to a desired neighborhood around the end-point. Similar to the RTRR-based formulation, most of the computational burden associated with the MPC is off-line and the formulation is therefore amenable to on-line implementation. To this end, consider a batch process described in Eq. 1 for fault-free conditions (i.e., $\sigma(t) = 1$) and for which robust RTRR estimates have been characterized for a given δ, Θ, and $\mathcal{B}(\mathbf{x}_\mathrm{d})$. The control action at sampling instance $k = \frac{t_\mathrm{f}-t}{\delta}$ is computed by solving the following bilevel optimization problem:

$$\min_{\mathbf{u}\in\mathbf{U}} J_{\tilde{\mathcal{R}}} = \int_t^{t+\delta} ||\tilde{\mathbf{x}}(\tau) - \mathbf{x}^*_{\mathrm{nom}}||_\mathbf{Q} + ||\Delta\mathbf{u}||_\mathbf{R} dt \tag{21}$$

$$\text{subject to} : \tilde{\mathbf{x}}(0) = \mathbf{x}(t) \tag{22}$$

$$\dot{\tilde{\mathbf{x}}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \mathbf{u}, \boldsymbol{\theta}) \tag{23}$$

$$||\tilde{\mathbf{x}}(\delta) - \tilde{\mathbf{c}}_{k-1}||_{\tilde{\mathbf{P}}_{k-1}} \le 1 \tag{24}$$

$$\max_{\boldsymbol{\theta}\in\Theta} J_\theta = ||\tilde{\mathbf{x}}(\delta) - \tilde{\mathbf{c}}_{k-1}||_{\tilde{\mathbf{P}}_{k-1}} \tag{25}$$

$$\text{subject to} : \dot{\tilde{\mathbf{x}}}(\tau) = \mathbf{f}(\tilde{\mathbf{x}}(\tau), \mathbf{u}, \boldsymbol{\theta}), \tag{26}$$

where $\Delta\mathbf{u}$ is a vector of deviations between successive control moves, $\mathbf{Q}$ and $\mathbf{R}$ are positive definite weighting matrices, and the objective function, $J_{\tilde{\mathcal{R}}}$, is formulated to minimize the deviations between process state trajectories and some nominal optimal state trajectories, $\mathbf{x}^*_{\mathrm{nom}}$, and variations in the control moves. This two-tier optimization problem is formulated in a similar fashion as the bottom two levels in the robust RTRR generation algorithm of Eqs. 17–20. Following initialization at the current plant values (Eq. 22), the worst-case values of the uncertainty are found from the maximization problem given in Eqs. 25–26. The worst-case uncertainty values are identified as those which drive the process states to the highest level set of the estimate of $\tilde{\mathcal{R}}_{k-1}$. For these worst-case values, the top level of the optimization problem searches for an input that minimizes some cost function, $J_{\tilde{\mathcal{R}}}$, and ensures the states at the next time step are contained within the robust RTRR. The methodology and algorithm used to compute the robust RTRR guarantees the feasibility of the MPC optimization problem in the fault-free scenario. Additionally, by definition, robust RTRRs take into account the requirement to drive the process to a desired end-point neighborhood. The implications on the guarantees of feasibility and driving the system to a desired end-point neighborhood are formalized below in Theorem 2. A sketch of a proof of this theorem is available in the Appendix.

Theorem 2. *Consider the constrained system of Eq.* 1 *with $\sigma = 1$ under the robust RTRR-based MPC law of Eqs.* 21–26. *If $\mathbf{x}(t_0) \in \tilde{\mathcal{R}}(t_0)$, the MPC bilevel optimization problem remains feasible for all $t \in [t_0, t_\mathrm{f}]$ and $\mathbf{x}(t_\mathrm{f}) \in \mathcal{B}(\mathbf{x}_\mathbf{d})$).*

Remark 8. In Theorem 2, the condition $x(t_0) \in \tilde{\mathcal{R}}(t_0)$ guarantees the existence of an input trajectory via guaranteed feasibility of the robust RTRR-based MPC to drive the process states inside $\mathcal{B}(\mathbf{x}_\mathbf{d})$ in the presence of uncertainty. However condition $x(t_0) \in \tilde{\mathcal{R}}(t_0)$ is not a necessary condition for driving the states inside $\mathcal{B}(\mathbf{x}_\mathbf{d})$) by batch termination even if we consider exact characterizations of true robust RTRRs in the robust RTRR-based MPC formulation. Consider the case where $\mathbf{x}(t_0) \notin \tilde{\mathcal{R}}(t_0)$. Although we cannot guarantee the existence of an input trajectory to ensure $\mathbf{x}(t_f) \in B(\mathbf{x}_\mathrm{d})$, this trajectory may still exist because it might be possible to drive the state inside $\mathcal{B}(\mathbf{x}_\mathrm{d})$ by t_f for some realization of the uncertainty (if not for all values of the uncertainty, as required in the robust RTRR definition). The theorem, however, does establish that the robust RTRR-based MPC problem will remain initially and successively feasible and drive the process to the desired end-point neighborhood.

Remark 9. In the MPC formulation in Eqs. 21–26, the performance index (Eq. 21) can be chosen to meet desired performance criteria or robustness objectives. For example, improving the disturbance rejection capabilities in the controller could be achieved by penalizing the Euclidean distance between the process states at sampling instance $k - 1$ and the center of the $\tilde{\mathcal{R}}_{k-1}$ estimate. This objective function would tend to steer the system through the centers of the robust RTRRs, thereby lowering the possibility of any unaccounted for disturbances driving the process outside of robust RTRRs.

Remark 10. One characteristic of batch processes is that the desired end-point quality, which is based on the values of quality variables at batch termination, typically remains consistent from batch to batch unless a new product is being manufactured. The main source of variation between batches is typically the initial condition in the batch as this is dictated by raw material properties, which are subject to variance depending on the source of raw materials. The robust RTRR-based MPC scheme is designed with these key properties in mind as robust RTRRs are generated for specific values of the quality variables at batch termination and also provide an explicit characterization of initial conditions for which the desired end-point quality can be met. Note that if the end-point quality in a batch is subject to change and discrete values of the other possible end-point qualities are known, robust RTRRs corresponding to all possible desired end-points can be generated beforehand, and the suitable robust RTRRs can be used during controller implementation.

Robust Safe-Steering of Batch Processes

In this section, we generalize the safe-steering framework to handle model uncertainty by using the robust RTRR-based MPC scheme presented in the previous section. First, we formulate the safe-steering problem and then present the robust safe-steering framework.

Problem statement

For systems described by in Eq. 1, we consider faults in the control actuator under the assumption that upon failure, the available control effort is reduced. More specifically, without loss of generality, we characterize the fault occurring in the first control actuator at a time t^{fault} which is repaired at time, t^{repair} (i.e., for $t < t^{\mathrm{fault}}$ and $t \ge t^{\mathrm{repair}}$, $\sigma(t) = 1$, whereas for $t^{\mathrm{fault}} \le t < t^{\mathrm{repair}}$, $\sigma(t) = 2$) as $u^1_{\mathrm{min,failed}} \le u^1_2(t) \le u^1_{\mathrm{max,failed}} \forall t \in [t^{\mathrm{fault}}, t^{\mathrm{repair}})$, where u^i denotes the i-th component of the input vector $\mathbf{u}$. Reduced control effort corresponds to a situation where $u^1_{\mathrm{min,failed}} > u^1_{\mathrm{min}}$ and $u^1_{\mathrm{max,failed}} < u^1_{\mathrm{max}}$. Note that in the case where a valve reverts to a completely open or shut position, we have $u^1_{\mathrm{min,failed}} = u^1_{\mathrm{max,failed}}$. The inputs available for control between t^{fault} and t^{repair} are therefore $u^1_2,\ldots,u^m_2$. We define the robust safe-steering problem as the one of identifying trajectories of these inputs during the fault rectification period (without requiring the value of t^{repair} or an estimate thereof to be known a priori) in the presence of model uncertainty that will ensure the process can be driven inside the desired end-point neighborhood upon recovery of the full control effort.

Safe-steering to a desired end-point neighborhood

The key idea in the robust safe-steering problem is to preserve the states in robust RTRRs during the failure period by using the robust RTRR-based MPC design. By doing so, the robust RTRR-based MPC scheme is able to drive the process to a desired end-point neighborhood following fault repair. Also note that the ability to go to a desired end-point neighborhood after fault repair is dependent on the duration of the fault. To this end, consider a batch system described in Eq. 1 for which the first control actuator fails at t^{fault} and is repaired at t^{repair} and the robust RTRRs for fault-free operation have been characterized for all sampling instances in the fault rectification period. We formalize the requirements for safe-steering the batch in Theorem 3.

Theorem 3. Consider the constrained system of Eq. 1 with $\mathbf{x}(t_0) \in \tilde{\mathcal{R}}(t_0)$ subject to the robust RTRR-based MPC scheme given in Eqs. 21–26. If the MPC optimization problem remains feasible $\forall t \in [t^{\mathrm{fault}}, t^{\mathrm{repair}}]$, then $x(t_{\mathrm{f}}) \in \mathcal{B}(\mathbf{x}_{\mathrm{d}})$.

Remark 11. The key idea formally expressed in Theorem 3 is that if a fault is repaired sufficiently fast (i.e., there exists an implementable input trajectory during the fault repair period, and one after fault repair), the robust RTRR-based MPC finds this trajectory via preserving the process states within robust RTRRs. The implication of this is that process states at t^{repair} will then belong to $\tilde{\mathcal{R}}(t^{\mathrm{repair}})$, and according to the definition of robust RTRRs, the process can then be driven to a desired end-point neighborhood. Therefore, maintaining the system within the robust RTRRs provides a sufficient condition to ensure that the desired neighborhood can be reached upon fault recovery. In other words, the proposed robust RTRR-based MPC is able to identify the input trajectory during faulty operation (if one exists) that will enable reaching the desired end-point neighborhood upon fault recovery. In contrast, end-point-based-MPC approaches can fail to find this trajectory even if it exists. The end-point MPC problem can become infeasible because it simply may not be possible to satisfy the terminal constraint with reduced control effort, which implies (appropriately truncated) infeasible solutions have to be implemented on the process. By repeatedly applying saturated versions of the infeasible solutions during the failure period, the states can be driven to an unrecoverable point from where reaching the desired end-point neighborhood is not possible even after fault recovery.

Remark 12. Theorem 3 provides sufficient conditions for fault-tolerant control in batch systems. To address the issue of necessary conditions, we note that if the fault is repaired too late, it can become impossible to preserve the system states in robust RTRRs using reduced control effort at some point between t^{fault} and t^{repair}. In this case, the states escape the robust RTRRs by t^{repair}; however, this does not necessarily imply the system states at batch termination will be outside $\mathcal{B}(\mathbf{x}_{\mathrm{d}})$. This is because the states at t^{repair} in this situation could reside in a region for which there exists a specific uncertainty realization and corresponding input trajectory that can drive the system inside the desired neighborhood. This can occur as $\mathbf{x}(t^{\mathrm{repair}}) \in \tilde{\mathcal{R}}(t^{\mathrm{repair}})$ is a sufficient but not necessary condition for driving the system inside $\mathcal{B}(\mathbf{x}_{\mathrm{d}})$.

Remark 13. The robust safe-steering framework can also be used to handle faults in batch systems where the batch time is flexible. To understand this, consider the case when the robust RTRR-based MPC law encounters infeasibility during the fault rectification period at sampling instance k_{infs}. In such a scenario, although there exists no control move using the limited control effort to drive the system from $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}}$ to $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}-1}$, it may be possible instead, given suitable system dynamics, to drive the system from $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}}$ to $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}+i}$, where i is a positive integer such that $k_{\mathrm{infs}} + i$ is a sampling instance for which the robust RTRR has been generated and characterized. In such a scenario, the MPC optimization problem given in Eqs. 21–26 can be modified accordingly by substituting $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}+i}$ in place of $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}-i}$. Next, we revert to the original MPC law that requires the process states to be driven from $\tilde{\mathcal{R}}_k$ to $\tilde{\mathcal{R}}_{k-1}$. Consequently, in i time steps, the MPC law will again require driving the system from $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}}$ to $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}-1}$. If the fault is rectified within these i time steps, Theorem 2 guarantees the MPC law will now be feasible. Using this procedure maintains the process states within robust RTRRs during the failure period at the cost of an increase in the batch termination time by $i\delta$. Note that it is also possible to repeat this procedure until the fault is rectified and use different values of i each time. Also worth noting is that if no $\tilde{\mathcal{R}}_{k_{\mathrm{infs}}+i}$ exists, this points to the fact that the batch may have gone beyond recovery. In such a case, the immediate (and necessary) action of discarding the batch can be taken instead of unsuccessfully trying to reach the desired end-point neighborhood and wasting valuable material and time.

Remark 14. In this work, we consider the full-state feedback problem to convey the key ideas in the robust safe-steering framework. If all system states are not measured, state observers or estimators can be incorporated into the framework such that the batch system can be safe-steered-based on state estimates (see the simulation example for an illustration). However, theoretical analysis of this effect remains outside the scope of this work.

Simulation Example

In this section, we first consider a fault-free environment and demonstrate the need for accounting for uncertainty. We then illustrate the robust safe-steering framework by considering an actuator failure in the process. To this end, consider a fed-batch reactor where an irreversible, first-order exothermic reaction of the form $A \rightarrow B$ takes place. The state-space model for this process takes the following form:

$$\begin{bmatrix} \dot{C}_A(t) \\ \dot{T}(t) \\ \dot{V}(t) \end{bmatrix} = \begin{bmatrix} -k_{A0}\exp\left\{\frac{E}{R}\left(\frac{1}{T_R}-\frac{1}{T(t)}\right)\right\}C_A(t) + \frac{F(t)(C_{A0}-C_A(t))}{V(t)} \\ \frac{UA(T_a(t)-T(t))}{\rho C_p V(t)} + \frac{F(t)(T_{A0}-T(t))}{V(t)} + \frac{(\Delta H)\left(-k_{A0}\exp\left\{\frac{E}{R}\left(\frac{1}{T_R}-\frac{1}{T(t)}\right)\right\}C_A(t)\right)}{\rho C_p} \\ F(t) \end{bmatrix}, \tag{27}$$

where $C_A(t)$, $T(t)$, and $V(t)$ denote the concentration of A, reactor temperature, and volume, respectively and constitute the state vector, $\mathbf{x}(t) = [C_A(t) \quad T(t) \quad V(t)]^T$. The physical meaning of the model parameters and their nominal values can be found in Table 1. The primary control objective considered in the simulation studies is to achieve the (arbitrarily chosen) desired end-point of $\mathbf{x}_d = [0.1 \quad 465 \quad 65]^T$ in the presence of uncertainty. The desired neighborhood around $\mathbf{x}_d$ is taken to be an ellipsoid with $\tilde{\mathbf{P}}_0 = \text{diag}\{10^4, 4, 11.1\bar{1}\}$ and $\tilde{\mathbf{c}}_0 = \mathbf{x}_d$. We consider uncertainty in the inlet temperature, T_{A0}, and the heat exchanger co-efficient (UA) of $\pm$ 5% around their nominal values. That is, $\boldsymbol{\theta} = [T_{A0} \quad UA]^T$ and the uncertainty bounds are defined by $\boldsymbol{\theta}_{\min} = [278.35 \quad 9.50 \times 10^3]^T$ and $\boldsymbol{\theta}_{\max} = [307.65 \quad 1.05 \times 10^4]^T$. The uncertainty in T_{A0} is representative of a process disturbance, whereas the heat transfer coefficient is a model parameter that is often not known precisely and varies with time because of the effects of fouling. The total batch time is taken to be $t_f = 30$ min with a controller execution period of $\delta = 36$ s. The manipulated variables in the process are the inlet feed rate, F (L/h), and the heating coil temperature, $T_a(K), \mathbf{u}(t) = [F(t) \quad T_A(t)]^T$, with constraints, $\mathbf{u}_{\min} = [0 \quad 285]^T$ and $\mathbf{u}_{\max} = [25 \quad 400]^T$. To demonstrate the applicability of the proposed method subject to limited (noisy) measurements, we consider the case when only noisy measurements of T and V are available, $\mathbf{y}(t) = [T(t) \quad V(t)]^T$, with standard deviations of 0.15 and 0.05, respectively.

To estimate C_A using on-line measurements of T and V, an extended Luenberger observer (ELO) is used that takes the following form:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), \boldsymbol{\theta}_{\text{nom}}) + \mathbf{L}(\mathbf{y}(t) - \hat{\mathbf{y}}(t)), \tag{28}$$

where $\hat{\mathbf{x}}(t) = [\hat{C}_A(t) \quad \hat{T}(t) \quad \hat{V}(t)]^T$ and $\hat{\mathbf{y}}(t) = [\hat{T}(t) \quad \hat{V}(t)]^T$ are vectors of the estimated states and outputs, respectively, the vector function, $\mathbf{f}(\cdot)$, is the right-hand side of the differential equations in Eq. 27, $\boldsymbol{\theta}_{\text{nom}}$ is the nominal set of parameters listed in Table 1, and $\mathbf{L}$ is a matrix of observer gains.

For this example, the model was successively linearized at the current estimate of the states, the computed value of the input, and the nominal set of parameters, and the gain was subsequently computed using the typical procedure used for linear dynamic systems. Specifically, the eigenvalues of the matrix, ($\mathbf{A} - \mathbf{LC}$), were placed on the left side of the complex plane where the ij-th element of $\mathbf{A}$ is given by:

$$a_{i,j} = \frac{\partial f_i}{\partial x_j}\Big|_{\hat{\mathbf{x}}(t),\mathbf{u}(t),\boldsymbol{\theta}_{\text{nom}}}. \tag{29}$$

Table 1. Process Parameters for the Fed-Batch Reactor of Eq. 27

Parameter	Description	Value	Unit
k_{A0}	Rate constant of reaction $A \rightarrow B$ at T_R	0.15	1/h
E	Activation energy of reaction $A \rightarrow B$	10^4	cal/mol
R	Universal gas constant	1.986	cal/(mol K)
T_R	Reference temperature	290	K
C_{A0}	Concentration of A in inlet feed stream	5	mol/L
UA	Heat transfer coefficient × Area	10^4	cal/(h K)
ρ	Solution and inlet feed density	1	kg/L
C_p	Solution and inlet feed heat capacity	65	cal/(kg K)
T_{A0}	Temperature of inlet feed stream	293	K
ΔH	Heat of reaction $A \rightarrow B$	−4000	cal/mol

Implementation of the robust reverse-time reachability region-based MPC scheme

To demonstrate the need for accounting for uncertainty, closed-loop simulations of the fed-batch system were performed using the nominal RTRR-based MPC design in Ref. 24 and the proposed robust MPC design. First, for the given desired neighborhood, uncertainty bounds, and input constraints, nominal and robust RTRRs were generated and characterized as ellipsoids for all sampling instants using the algorithm described in the "Generating Robust Reverse-Time Reachability Regions" section. The following objective functions were used for the MPC formulations.

$$J_{\mathcal{R}} = ||\tilde{\mathbf{x}}(\delta) - \mathbf{c}_{k-1}||_{\mathbf{P}_{k-1}} + ||\Delta\mathbf{u}||_{\mathbf{R}} \tag{30}$$

$$J_{\tilde{\mathcal{R}}} = ||\tilde{\mathbf{x}}(\delta) - \tilde{\mathbf{c}}_{k-1}||_{\tilde{\mathbf{P}}_{k-1}} + ||\Delta\mathbf{u}||_{\mathbf{R}} \tag{31}$$

where $J_{\mathcal{R}}$ and $J_{\tilde{\mathcal{R}}}$ denote the objective functions for the nominal and robust formulations, respectively. The move suppression matrix was set to $\mathbf{R} = \text{diag}\{5 \times 10^{-4}, 9 \times 10^{-4}\}$. The initial conditions were specified to be $\mathbf{x}_{\mathcal{R}}(0) = [2.648 \quad 275.16 \quad 59.88\,3]^T$ and $\mathbf{x}_{\tilde{\mathcal{R}}}(0) = [2.65 \quad 279.395 \quad 58.483]^T$ for the nominal and robust MPC schemes respectively and verified to be within the appropriate RTRR. The initial-state estimates were also chosen such that they resided within the appropriate RTRR. The deviation between the initial-state estimates and the actual states was consistent for both cases. The estimates were set to $\hat{\mathbf{x}}_{\mathcal{R}}(0) = [2.675 \quad 276.786 \quad 59.09]^T$ and $\hat{\mathbf{x}}_{\tilde{\mathcal{R}}}(0) = [2.659 \quad 281.021 \quad 58.691]^T$ for the nominal and robust cases, respectively. The following

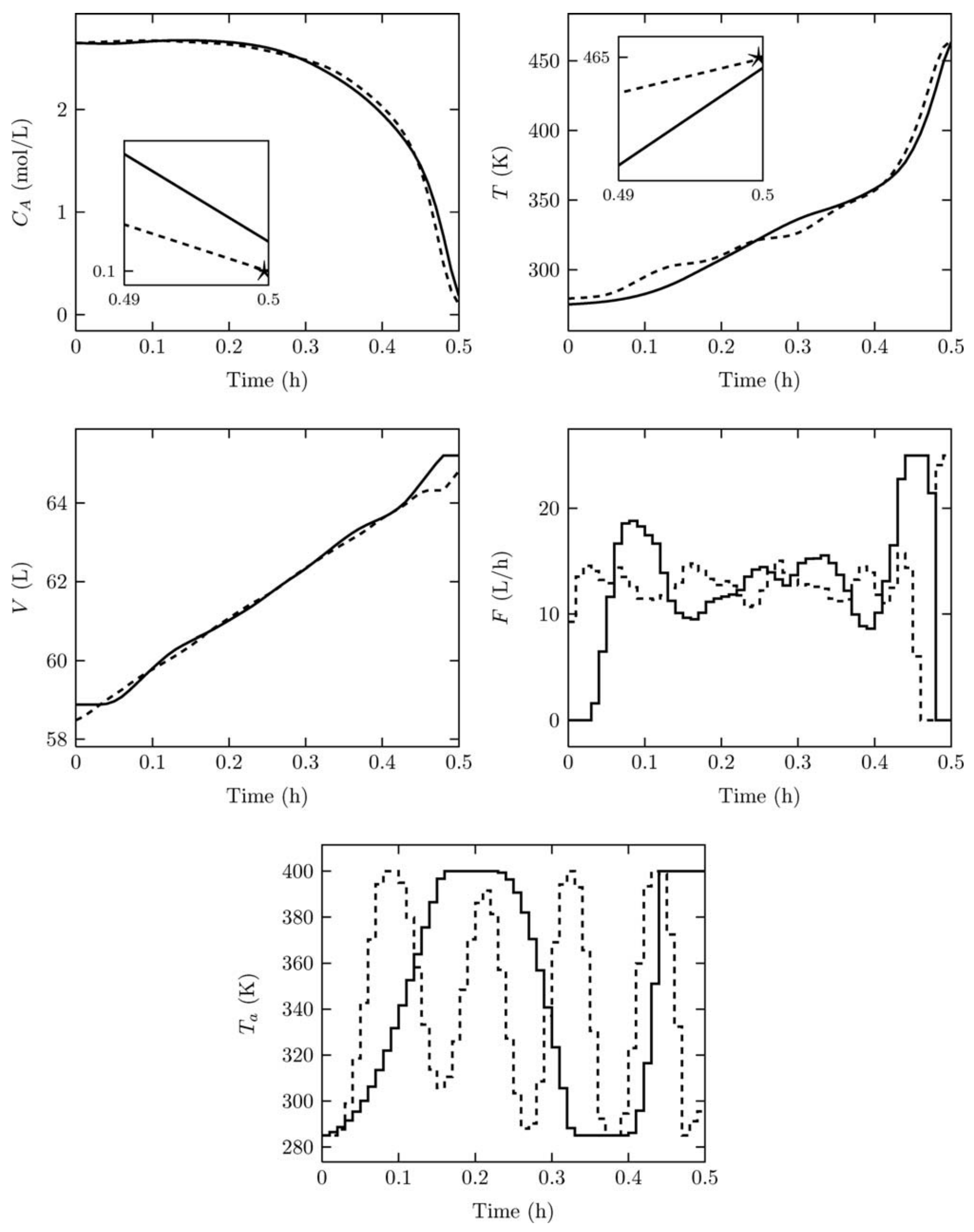

Figure 1. State and input profiles of the fed-batch reactor system under the nominal (solid) and robust (dashed) RTRR-based MPC designs during a fault-free batch.

The insets show the ability of the proposed robust MPC formulation to drive the states inside a desired neighborhood of $\mathbf{x}_d$. The ★ denotes the desired end-point value.

set of eigenvalues were chosen for the ELO design: $-0.9, -1.05$, and -1.1.

The state and input profiles from the closed-loop simulations for both MPC designs are presented in Figure 1. For the nominal case, $\mathbf{x}_{\tilde{\mathcal{R}}}(t_f) = [0.185 \quad 466.448 \quad 64.212]^T$, which does not reside in $\mathcal{B}(\mathbf{x}_d)$ as $||\tilde{\mathbf{x}}(t_f) - c_0||_{\mathbf{P}_0} = 82.158$, whereas the robust MPC scheme was able to steer the system inside $\mathcal{B}(\mathbf{x}_d)$ as $\mathbf{x}_{\tilde{\mathcal{R}}}(t_f) = [0.0992 \quad 464.835 \quad 64.817]^T$ and $||\tilde{\mathbf{x}}(t_f) - \tilde{\mathbf{c}}_0||_{\tilde{\mathbf{P}}_0} = 0.488$. These results indicate the practical importance of explicitly accounting for uncertainty in the controller design. Using the Matlab functions, tic and toc, the average CPU times required per MPC calculation on an Intel Quad Core machine for the nominal and robust RTRR-based MPC schemes were 0.146 and 1.507 s, respectively. The higher CPU time for the robust RTRR-based design is certainly expected as the MPC optimization problem is more complex; however, the computational trade-offs are well worth the savings acquired from achieving a batch quality within the desired neighborhood.

Robust safe-steering of the fed-batch process

To demonstrate the effectiveness of the proposed safe-steering framework for batch processes with model uncertainty, we consider a fault in one of the control actuators in the fed-batch process. Specifically, we consider the scenario where at $t^{\text{fault}} = 0.25$ h, the actuator associated with the

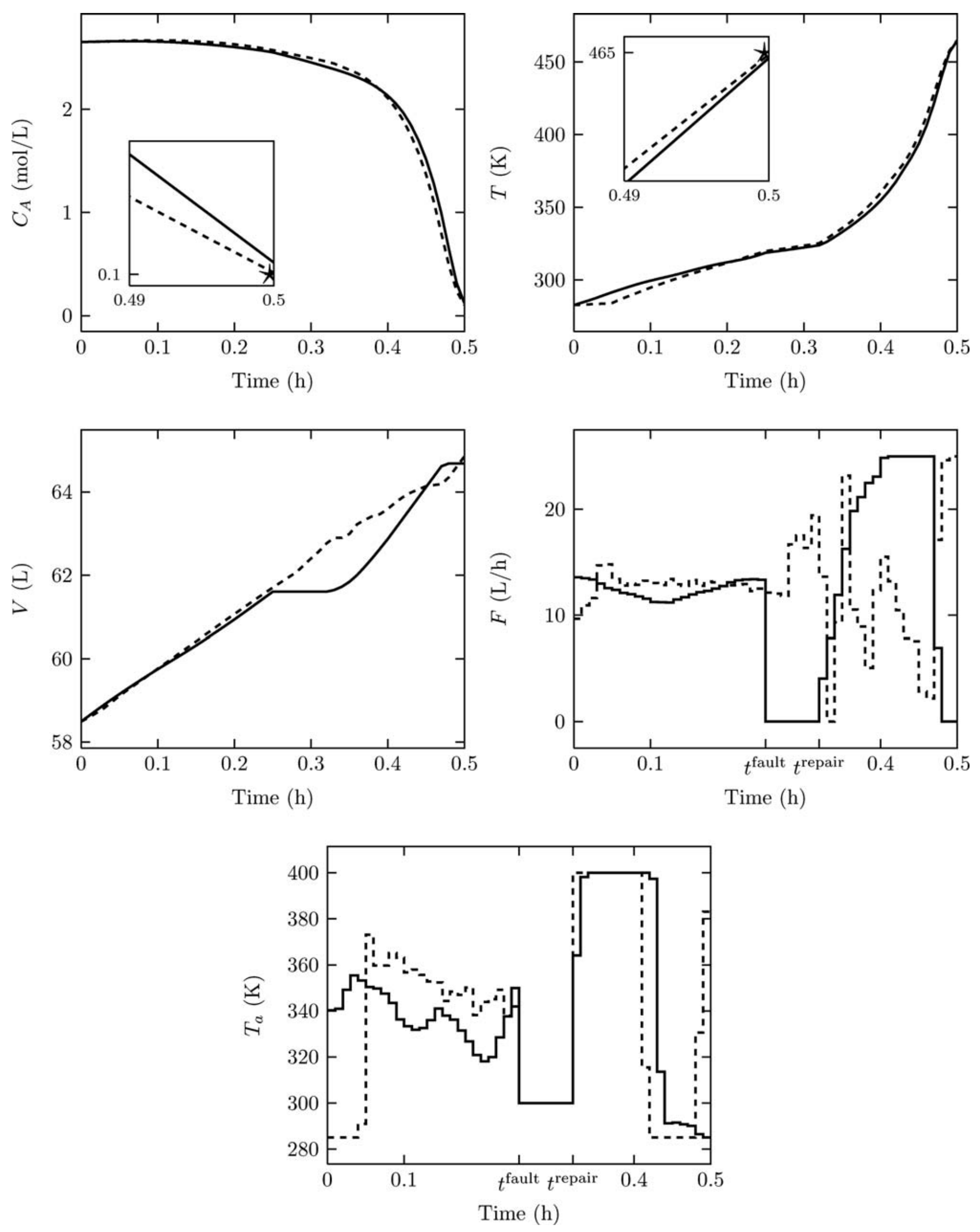

Figure 2. State and input profiles of the fed-batch reactor system under the end-point (solid) and robust RTRR (dashed)-based MPC designs with input failure of T_a between 0.25 and 0.32 h.

The insets show the ability of the proposed safe-steering framework to drive the states inside a desired neighborhood of $\mathbf{x}_d$. The ★ denotes the desired end-point value.

heating coil temperature fails. During the failure period, the heating coil can only supply limited heat to the reactor and the heating coil temperature becomes restricted to $285 \leq T_a \leq 300$ K. At $t^{\text{repair}} = 0.32$ h, the fault is rectified and the full control effort is recovered. For the end-point-based predictive controller, the following objective function was used.

$$J_E = ||\tilde{\mathbf{x}}(t_f) - \tilde{\mathbf{c}}_0||_{\tilde{\mathbf{P}}_0} + \int_t^{t_f} ||\Delta\mathbf{u}||_{\mathbf{R}} dt \tag{32}$$

The objective function in Eqn. 31 was used for the robust RTRR-based MPC design. The move suppression matrix, initial conditions, and initial-state estimates were identical in the simulations for both designs. Specifically, we set: $\mathbf{R} = \text{diag}\{5 \times 10^{-5}, 5 \times 10^{-6}\}, \mathbf{x}(0) = [2.651 \quad 282.712 \quad 58.493]^T$ and $\hat{\mathbf{x}}(0) = [2.66 \quad 284.337 \quad 58.701]^T$. The actual state and state estimates were verified to be within $\tilde{\mathcal{R}}(0)$. The state and input profiles for the two MPC schemes are given in Figure 2. For the end-point-based MPC design, the batch is driven to the end-point of $\mathbf{x}_\varepsilon(t_f) = [0.121 \quad 464.659 \quad 64.686]^T$, which is outside of the desired neighborhood as $||\mathbf{x}_\varepsilon(t_f) - \mathbf{c}_0||_{\mathbf{P}0} = 5.994$.

During the failure period, the heating coil temperature in both designs is prescribed to be 300 K whereas the flow rate trajectories are markedly different. Using only the flow rate, the end-point MPC design was unable to compute a trajectory that steered the system inside the desired neighborhood. Because of the repeated application of a truncated version of these poor input trajectories, the system is driven to a point by t^{repair} from where it cannot be steered inside $\mathcal{B}(\mathbf{x}_d)$ even after fault recovery. In contrast, the robust RTRR MPC design prescribed flow rates during the failure period which maintained the process states from where the batch could be driven inside the desired neighborhood upon recovery of the full control effort. The batch is ultimately steered to the end-point of $\mathbf{x}_{\tilde{\mathcal{R}}}(t_f) = [0.103 \quad 464.882 \quad 64.854]^T$, which is inside the desired neighborhood as $||\mathbf{x}_{\tilde{\mathcal{R}}}(t_f) - \mathbf{c}_0||_{\mathbf{P}_0} = 0.380$.

We also note the total CPU simulation time required for the robust RTRR-based MPC was 2.126 min, which was significantly shorter compared with the end-point-based MPC simulation time of 2.318 h. Closed-loop simulations of batch systems with a higher number of states or inputs would exhibit an even more substantial difference in the simulation CPU time. Moreover, with additional parameter uncertainties, wider uncertainty ranges, and the introduction of disturbances into the system, the end-point-based MPC would require more computational time because the solution at one time step would become a poorer initial guess for the next time step. On the other hand, because the robust RTRR-based MPC formulation accounts for the presence of the uncertainty and its bounds in off-line calculations (which would certainly increase), the CPU time for on-line control calculations would not increase significantly.

Conclusions

In this work, we first considered the control of batch processes subject to input constraints and model uncertainty with the objective of reaching a desired end-point neighborhood. To this end, a computationally efficient nonlinear robust MPC scheme based on robust RTRRs was formulated. Before the MPC formulation, a multilevel optimization-based algorithm was established to generate robust RTRRs for specified uncertainty bounds, sampling period, and desired end-point neighborhood. Following the controller design, we considered the problem of uncertain batch processes subject to finite duration faults in the control actuators that cannot be handled via robust control approaches. Using the robust RTRR-based MPC as the main tool, the robust safe-steering framework was developed to address the problem of how to operate the functioning inputs during the fault repair period to ensure that the process can be driven inside a desired end-point neighborhood upon recovery of the full control effort. The computational efficiency and control performance of the robust RTRR-based MPC and its usefulness in the robust safe-steering framework were demonstrated using simulations of a fed-batch process example.

Acknowledgments

Financial support from the McMaster Advanced Control Consortium and the Natural Sciences and Engineering Research Council of Canada (NSERC) through the CGS(D) award for Siam Aumi is gratefully acknowledged.

Literature Cited

1. Cruickshank SM, Daugulis AJ, McLellan PJ. Dynamic modeling and optimal fed-batch feeding strategies for a two-phase p6artitioning bioreactor. *Biotech Bioeng*. 2000;67:224–233.
2. Dadebo SA, McAuley KB. Dynamic optimization of constrained chemical engineering problems using dynamic programming. *Comput Chem Eng*. 1995;19:513–525.
3. Zhang GP, Rohani S. On-line optimal control of a seeded batch cooling crystallizer. *Chem Eng Sci*. 2003;58:1887–1896.
4. Soroush M, Kravaris C. Optimal-design and operation of batch reactors. I. Theoretical framework. *Ind Eng Chem Res*. 1993;32:866–881.
5. Soroush M, Kravaris C. Optimal-design and operation of batch reactors. II. A case-study. *Ind Eng Chem Res*. 1993;32:882–893.
6. Flores-Cerrillo J, MacGregor JF. Latent variable MPC for trajectory tracking in batch processes. *J Process Control*. 2005;15:651–663.
7. Valappil J, Georgakis C. State estimation and nonlinear model predictive control of end-use properties in batch reactors. In: *Procedings of the 2001 American Control Conference*, Vol. 2. Arlington, VA, IEEE; 2001:999–1004.
8. Alamir M, Balloul I. Robust constrained control algorithm for general batch processes. *Int J Control*. 1999;72:1271–1287.
9. Mayne DQ. Non-linear model predictive control: challenges and opportunities. In: Allgower F, Zheng A, editor. *Non-Linear Model Predictive Control*., Basel: Birkhauser; 2000: 23–44.
10. Crowley T, Choi K. Experimental studies on optimal molecular weight distribution control in a batch-free radical polymerization processes. *Chem Eng Sci*. 1998;53:2769–2790.
11. Dimitratos J, Georgakis C, El-Aasser MS, Klein A. Dynamic modeling and state estimation for an emulsion copolymerization reactor. *Comput Chem Eng*. 1989;13:21–33.
12. Kozub D, Macgregor JF. Feedback control of polymer quality in semi-batch copolymerization reactors. *Chem Eng Sci*. 1992;47:929–942.
13. De Valliere P, Bonvin D. Application of estimation techniques to batch reactors. II. Experimental studies in state and parameter estimation. *Comput Chem Eng*. 1989;13:11–20.
14. Hua XM, Rohani S, Jutan A. Cascade closed-loop optimization and control of batch reactors. *Chem Eng Sci*. 2004;59:5695–5708.
15. Shi D, Mhaskar P, El-Farra NH, Christofides PD. Predictive control of crystal size distribution in protein crystallization. *Nanotechnology*. 2005;16:S562–S574.
16. Shi D, El-Farra NH, Li M, Mhaskar P, Christofides PD. Predictive control of particle size distribution in particulate processes. *Chem Eng Sci*. 2005;61:268–281.
17. Sheikhzadeh M, Trifkovic M, Rohani S. Real-time optimal control of an anti-solvent isothermal semi-batch crystallization process. *Chem Eng Sci*. 2008;63:829–839.
18. Palanki S, Vemuri J. Optimal operation of semi-batch processes with a single reaction. *Int J Chem React Eng*. 2005;3:A17.
19. Pistikopoulos EN, Dua V, Bozinis NA, Bemporad A, Morari M. On-line optimization via off-line parametric optimization tools. *Comput Chem Eng*. 2002;26:175–185.
20. Huynh N, Kazantzis N. Parametric optimization of digitally controlled nonlinear reactor dynamics using Zubov-like functional equations. *J Math Chem*. 2005;38:499–519.
21. Mhaskar P, El-Farra NH, Christofides PD. Predictive control of switched nonlinear systems with scheduled mode transitions. *IEEE Trans Automat Contr*. 2005;50:1670–1680.
22. Mhaskar P. Robust model predictive control design for fault-tolerant control of process systems. *Ind Eng Chem Res*. 2006;45:8565–8574.
23. Mahmood M, Gandhi R, Mhaskar P. Safe-parking of nonlinear process systems: handling uncertainty and unavailability of measurements. *Chem Eng Sci*. 2008;63:5434–5446.
24. Aumi S, Mhaskar P. Safe-steering of Batch Processes. *AIChE J*. 2009;55:2861–2872.
25. Nomikos P, Macgregor JF. Monitoring batch processes using multi-way principal component analysis. *AIChE J*. 1994;40:1361–1375.
26. Cinar A, Parulekar SJ, Undey C, Birol G. *Batch Fermentation: Modeling: Monitoring, and Control*. New York: CRC Press. 2003.
27. Undey C, Tatara E, Williams B, Birol G, Cinar A. A hybrid supervisory knowledge-based system for monitoring penicillin fermentation. In: *Proceedings of American Control Conference*, Vol. 6. Chicago, Il; IEEE; 2000:3944–3948.
28. Undey C, Tatara E, Williams B, Birol G, Cinar A. *On-line real-time monitoring of penicillin fermentation*. In: *International Symposium on Advanced Control of Chemical Processes*, Vol. 6, Pisa, Italy, IFAC;2000:243–248.
29. Undey C, Cinar A. Statistical monitoring of multiphase, multistage batch processes. *IEEE Control Syste Mag*. 2002;22:40–52.

30. Mhaskar P, McFall C, Gani A, Christofides PD, Davis JF. Isolation and handling of actuator faults in nonlinear systems. *Automatica.* 2008;44:53–62.
31. Gandhi R, Mhaskar P. Safe-parking of nonlinear process systems. *Comput Chem Eng.* 2008;32:2113–2122.
32. Terwiesch P, Agarwal M, Rippin DWT. Batch unit optimization with imperfect modelling: a survey. *J Proc Control.* 1994;4:238–258.
33. Khalil HK. *Nonlinear Systems*; 3rd ed. New York: Prentice Hall, 2002.

Appendix

Proof of Theorem 1: Consider an element $\mathbf{x}_{k-1,\mathrm{nom}}$ in the interior of $\tilde{\mathcal{R}}_{k-1}$ (i.e., there exists a ρ^* such that $\mathcal{I}_{k-1} = \{\mathbf{y}| \|\mathbf{y} - \mathbf{x}_{k-1}\| \le \rho^*\} \subset \tilde{\mathcal{R}}_{k-1}$) and the point $\mathbf{x}_{k,\mathrm{nom}}$ given by:

$$\mathbf{x}_{k,\mathrm{nom}} = \mathbf{x}_{k-1,\mathrm{nom}} + \int_t^{t+\delta} -\mathbf{f}(\mathbf{x}(t), \mathbf{u}_{\mathrm{nom}}, \boldsymbol{\theta}_{\mathrm{nom}})dt, \quad (33)$$

where δ is to be determined and $\mathbf{u}_{\mathrm{nom}}$ and $\boldsymbol{\theta}_{\mathrm{nom}}$ are nominal values of the input and uncertainty, respectively. It follows that

$$\mathbf{x}_{k-1,\mathrm{nom}} = \mathbf{x}_{k,\mathrm{nom}} + \int_t^{t+\delta} \mathbf{f}(\mathbf{x}(t), \mathbf{u}_{\mathrm{nom}}, \boldsymbol{\theta}_{\mathrm{nom}})dt, \quad (34)$$

Define:

$$\mathbf{x}_{k-1} = \mathbf{x}_{k,\mathrm{nom}} + \int_t^{t+\delta} \mathbf{f}(\mathbf{x}(t), \mathbf{u}_{\mathrm{nom}}, \boldsymbol{\theta}(t))dt, \quad (35)$$

i.e., $\mathbf{x}_{k-1}$ is the state of the system at $t + \delta$ starting at t from $\mathbf{x}_{k,\mathrm{nom}}$ subject to the true value of the uncertain variable. From the continuity of $\mathbf{f}(\mathbf{x},\mathbf{u},\boldsymbol{\theta})$ on $(\mathbf{x},\mathbf{u},\boldsymbol{\theta})$ and that it is locally Lipschitz in $\mathbf{x}$ on $\mathcal{D} \times \mathbf{U} \times \Theta$, the continuity of (Theorem 3.5 in Ref. 33). From the proof of Theorem 3.5 in Ref. 33, it follows that given a desired bound on the discrepancy between the evolution of the nominal and perturbed system (i.e., $\|\mathbf{x}_{k-1} - \mathbf{x}_{k-1,\mathrm{nom}}\| \le \rho^*$), there exists a value δ^* such that if the sampling period, $\delta \le \delta^*$ then it is guaranteed that $\|\mathbf{x}_{k-1} - \mathbf{x}_{k-1,\mathrm{nom}}\| \le \rho^*$. Therefore, $\mathbf{x}_{k,\mathrm{nom}}$ is an element of $\tilde{\mathcal{R}}_k$ showing $\tilde{\mathcal{R}}_k \ne \emptyset$. This completes the proof of Theorem 1. ■

Sketch of the Proof of Theorem 2: The sufficiency of the condition in Theorem 2 can be shown by considering any $\mathbf{x}(t_0) \in \tilde{\mathcal{R}}(t_0)$. From the properties of the generation algorithm for $\tilde{\mathcal{R}}(t)$, there exists a control moves that takes the states inside $\tilde{\mathcal{R}}(t+\delta)$ in a time δ. Repeating this for the duration of the batch implies that the state is driven to $\mathcal{B}(\mathbf{x}_\mathrm{d})$ by t_f for all possible realizations of the uncertainty. In essence, this implies that there always exists a feasible solution to the MPC law of Eqs. 21–26 for the entire duration of the batch. The bilevel optimization problem defining the MPC law in Eqs. 21–26 remains feasible for all $t \in [t_0,t_f]$ and $\mathbf{x}(t_f) \in B(\mathbf{x}_\mathrm{d})$. This completes the sketch of the proof of Theorem 2. ■

Proof of Theorem 3: The proof of this theorem follows from Theorem 2. Equating t^{repair} to t_0 results in the satisfaction of the requirements of Theorem 2 and therefore, the bilevel MPC optimization problem of Eqs. 21–26 continues to remain feasible $\forall t \in [t^{\mathrm{repair}}, t_f]$ and $\mathbf{x}(t_f) \in \mathcal{B}(\mathbf{x}_\mathrm{d})$ follows. This completes the proof of Theorem 3. ■

Manuscript received Sept. 30, 2009, revision received Jun. 11, 2010, and final revision received Aug. 4, 2010.