AIChE

Safe-Steering of Batch Process Systems

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

DOI 10.1002/aic.11920
Published online August 11, 2009 in Wiley InterScience (www.interscience.wiley.com).

This work considers the problem of controlling batch processes to achieve a desired final product quality subject to input constraints and faults in the control actuators. Specifically, faults are considered that cannot be handled via robust control approaches, and preclude the ability to reach the desired end-point, necessitating fault-rectification. A safe-steering framework is developed to address the problem of determining how to utilize the functioning inputs during fault rectification to ensure that after fault-rectification, the desired product properties can be reached upon batch termination. To this end, first a novel reverse-time reachability region (we define the reverse time reachability region as the set of states from where the desired end point can be reached by batch termination) based MPC is formulated that reduces online computations, as well as provides a useful tool for handling faults. Next, a safe-steering framework is developed that utilizes the reverse-time reachability region based MPC in steering the state trajectory during fault rectification to enable (upon fault recovery) the achieving of the desired end point properties by batch termination. The proposed controller and safe-steering framework are illustrated using a fed-batch process example. © 2009 American Institute of Chemical Engineers *AIChE J,* 55: 2861–2872, 2009

Keywords: process control, control, safety, batch control

Introduction

Faults in processing or auxiliary equipment (sensors, actuators, etc.) are ubiquitous in the chemical process industry and can have a serious impact on product quality, lead to undesirable pollutant emissions, and negatively impact the overall plant productivity and economy. Batch and fedbatch process productivity is particularly susceptible to faults as there is an emphasis on final product quality, and a fault during a batch may ruin the entire batch product. Specialized sectors of the chemical industry operate exclusively in batch or fed-batch modes. Key examples include (1) the production of bio-chemicals (e.g., ethanol), polymers, and pharmaceutical products, (2) the separation and transformation of materials by batch distillation, and (3) the processing of materials by injection molding. Additionally, for continuous processes with an initial batch mode of operation, namely bio-processes, conditions at batch termination can ultimately dictate process performance upon the transition to continuous mode.[1] In addition to faults, other operational issues such as constraints and disturbances must be accounted for with suitable batch operating policies, appropriate monitoring schemes, and well-designed batch process control structures.

The primary control objective in batch processes is to reach a desired product quality by batch termination. As a result, batch process control structures are designed using end-point based control approaches. Early approaches to batch process operation consisted of open-loop policies in which optimal state trajectories were determined off-line by solving a dynamic optimization (DO) problem (e.g. Ref. 2) or through past experience. These state trajectories, which incorporated desired end-point properties, were subsequently tracked using local PID or predictive controllers (e.g. Refs. 3–5).

In many instances, however, process noise and disturbances can render the optimal state trajectories sub-optimal or the tracking problem infeasible. In response to this, one modification to a completely open loop control policy has been to correct optimal state trajectories mid-course at specified decision points in the batch based on the projected end-point properties.[6] With increased computational power

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

together with more efficient optimization algorithms, shrinking horizon model predictive control (MPC) of batch processes are becoming commonplace. In this approach, the optimization problem is solved at each sampling instance until batch termination, using plant measurements (or state variable estimates) to update model predictions at each sampling time. Rather than relying on tracking controllers, the resulting optimal input changes are then implemented directly on the process. Differentiating features among these MPC approaches include the complexity of the model used for predictions and the nature of the input parametrization in the dynamic optimization problem. With respect to the former, the predictive model may be a linearized version of the full non-linear model or the rigorous, first-principles non-linear model. Because of the strong non-linearities present in most batch processes, performance using linear models is severely limited. Successive linearization techniques or scheduling of multiple linear models represent a work-around to non-linear MPC in these cases, but with the significant reduction of computational times of real-time optimization algorithms, MPC based on full non-linear models is becoming increasingly tractable.[7–10] Another factor affecting the computational effort in an MPC approach is input parametrization. Inherent to the typical MPC algorithm, the optimal solution consists of the entire input trajectory from the time at which the problem is solved to batch termination, implying significant computational effort especially during the start of the batch. In fact, a majority of the developments in real-time optimization algorithms are based on improved input parametrization techniques, which can be derived from an off-line characterization of an optimal solution and strive to reduce the number of decision variables in the optimization problem (e.g. Refs. 11–14). However, with modelling errors and process noise, actual process trajectories may deviate considerably from the nominal one, rendering a certain input parametrization sub-optimal or infeasible in the worst case. One contribution of the present work is a computationally efficient model predictive control scheme, which only requires the online computation of the current control action, while guaranteeing end-point reachability, if achievable.

For batch (as well as continuous) processes, the occurrence of a fault can invalidate the desirable properties of a control design. Compared to batch systems, there has been extensive research on fault-tolerant control structures (FTCS) for continuous processes. Most of the existing methods for FTC rely on the assumption of availability of sufficient control effort or redundant control configurations to maintain operation at the nominal equilibrium point in the presence of faults. These methods can be categorized within robust/reliable control approaches (also called passive FTC; e.g. Ref. 15) and reconfiguration-based fault-tolerant control approaches (also called active FTC, e.g. Refs. 16–22). More recently, the control of non-linear (continuous) processes subject to input constraints and faults that preclude the possibility of operation at the nominal equilibrium point during a fault has been studied. This led to the development of a safe-parking framework.[23] The safe-parking fault tolerant control framework specifically considers the class of equipment failure that does not allow continued operation at the nominal operating point due to input constraints. The framework answers the problem of choosing what steady state to operate the plant during fault rectification such that a smooth transition back to the nominal (i.e., fault-free) equilibrium point is feasible and optimal with respect to some measure of plant economics. The safe-parking framework was recently generalized to address the availability of limited measurements and uncertainty.[24]

The extensive results for handling faults for continuous processes, however, do not carry over to batch processes. Specifically, the absence of equilibrium points in batch processes and fundamental differences in the control objectives between batch and continuous processes[4] prevent the direct applicability of research results from continuous processes. As with the robust and reconfiguration based approaches for continuous processes, although several robust non-linear MPC structures for batch processes are available[25,26] (as well as dedicated fault monitoring designs for batch systems; e.g. Refs. 27,28), the fault-tolerant characteristic in these formulations stems from the underlying assumption of availability of sufficient control effort such that the desired objective of the batch, or the tracking of desired state trajectories, remains achievable following fault repair. The recently developed safe-parking framework[23,24] does not remain applicable due to the absence of nominal equilibrium point or safe-parking points. In particular, the requirement to safe-park in[23,24] is that of the existence of operating points where the process can be stabilized in the event of a fault, and this requirement does not hold in the event of a fault in a batch process. With the basic premise of the safe-parking framework not holding for batch processes, the subsequent method of handling faults[23,24] does not apply to batch processes.

In the absence of a framework for handling faults in batch processes, continuation of the implementation of controllers to drive the process to the desired end-point may not be the best option. For instance, if one of the inputs in a batch process fails (i.e., an actuator is ''stuck'' at a fail-safe value), it is likely that the optimization problem in a conventional end-point based MPC may become infeasible during the faulty period because the desired end point properties can no longer be reached with limited available input for the rest of the batch duration. If the fault is repaired sufficiently fast, on the other hand, it may still be possible to reach the desired end-point. However, without the knowledge of the fault-repair time, traditional MPC approaches (during fault-rectification) would dictate computing the manipulated input trajectory using the reduced control action till batch termination (therefore yielding an infeasible solution). By repeatedly applying a saturated version of an infeasible input trajectory, the process can be driven to a point where it is no longer possible to meet desired end point properties even if the fault is repaired in due time. Therefore, the batch process control problem may continue to remain infeasible even after fault rectification and the desired end point properties will not be reached. This could result in the loss of the batch product, as well as significant wastage of time and money for reactor cleanup, if required.

Motivated by the above considerations, this work considers the problem of control of batch processes to achieve a desired final product quality subject to input constraints and faults in the control actuators. Specifically, faults are considered that cannot be handled via robust control approaches. A safe-steering framework is developed to address the problem of determining how to utilize the functioning inputs during

fault rectification to achieve the desired product properties at batch termination. The rest of this manuscript is organized as follows: the class of processes considered is first presented followed by a review of a representative end-point based non-linear MPC formulation for batch process control. Next, we design a reverse-time reachability region based predictive controller that requires the online computation of only the immediate control move. The safe-steering problem is then formulated, and we use the reachability region based controller design to come up with the safe-steering framework. Then, a fed-batch bioreactor example is used to illustrate the details of the safe-steering framework. Finally, we summarize our results.

Preliminaries

In this section, the class of batch processes considered is presented followed by a representative formulation of existing non-linear, shrinking horizon predictive controller used for batch process control.

Process description

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

$$\begin{aligned} &\dot{x} = f(x(t), u_\sigma(t)) \\ &t \in [t_0, t_\mathrm{f}], u_\sigma(\cdot) \in \mathrm{U}_\sigma, x(t_0) = x_0 \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ denotes the vector of state variables and $u_\sigma(t) \in \mathbb{R}^m$ denotes the vector of constrained manipulated inputs, taking values in a non-empty convex subset U_σ of $\mathbb{R}^m$, where $\mathrm{U}_\sigma = \{u \in \mathbb{R}^m : u_{\min,\sigma} \leq u \leq u_{\max,\sigma}\}$, where $u_{\min,\sigma}, u_{\max,\sigma} \in \mathbb{R}^m$denote the constraints on the manipulated inputs. 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, specifically $\sigma = 1$ denotes fault-free operation and $\sigma = 2$ denotes faulty operation. The faulty operation specifically considered in this work involves actuator failure for a finite duration of time defined by the time of fault occurrence, t^{fault}, and time of fault recovery, t^{recovery}. An example of such a failure condition is a flow valve feeding a fedbatch reactor becoming "stuck" at its fail-safe value between t^{fault} and t^{recovery} while remaining operational at all other times. According to the definition of σ, $\sigma = 1$ for $t \in [t_0\ t^{\mathrm{fault}})$, $\sigma = 2$ for $t \in [t^{\mathrm{fault}}\ t^{\mathrm{recovery}})$, and $\sigma = 1$ for $t \in [t^{\mathrm{recovery}}\ t_\mathrm{f}]$. The vector function $f : \mathbb{R}^n \times \mathrm{U}_\sigma \rightarrow \mathbb{R}^n$ is assumed to be sufficiently smooth on its domain of definition. The notation, $\|\cdot\|_Q$, refers to the weighted norm, defined by $\|x\|_Q = x^T Q x$ for all $x \in \mathbb{R}^n$, where Q is a positive-definite symmetric matrix and x^T denotes the transpose of x. Throughout the manuscript, we assume for any $u \in \mathrm{U}_\sigma$, the solution of the batch system of Eq. 1 exists and is continuous for all $t \in [t_0, t_\mathrm{f}]$. For this work, we focus on the state feedback problem where $x(t)$ 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, non-linear predictive controller is presented. Note that the representative formulation is not meant to generalize all variations of published MPC formulations for batch processes, but only meant to capture the key idea in most existing formulations, which is the computation of the entire manipulated trajectory from the current time to the end of the batch. To this end, consider the batch process described by Eq. 1 for $\sigma(t) = 1$ (i.e., a fault-free environment where all manipulated inputs can be changed via a feedback law), for which the control action at each sampling instance is computed by solving a dynamic optimization problem of the form:

$$u_{\mathrm{MPC}}(x) := \arg\min\{J_{\mathrm{EP}}(x, t, u(\cdot)) | u(\cdot) \in S\} \tag{2}$$

subject to:

$$\dot{\tilde{x}} = f(\tilde{\mathrm{x}}(\tau), u(\tau)) \tag{3}$$

$$\tilde{\mathrm{x}}(0) = x(t) \tag{4}$$

$$\tilde{x}(t_\mathrm{f} - t) = x_\mathrm{d} \tag{5}$$

where $S = S(t,T)$ is the family of piecewise continuous functions (functions continuous from the right), with period Δ, mapping $[t, t + T]$ into U, x_0 represents the initial condition of the batch, and x_d denotes specified desired states at t_f. A control $u(\cdot)$ in S is characterized by the sequence $\{u[j]\}$ where $u[j] := u(j\Delta)$ and satisfies $u(t) = u[j]$ for all $t \in [j\Delta, (j + 1)\Delta)$. The minimizing control $u_{\mathrm{MPC}}(\cdot) \in S$ is then applied to the plant over the interval $[t, t + \Delta)$ and the procedure is repeated until batch termination. The performance objective, $J_{\mathrm{EP}}(x, t, u(\cdot))$, in the optimization problem can have the following (general) form:

$$J_{\mathrm{EP}}(x, t, u(\cdot)) = M(t_0, \tilde{x}(t_0), t_\mathrm{f}, \tilde{x}(t_\mathrm{f}), x_\mathrm{d}) + \int_t^{t_\mathrm{f}} L(\tilde{x}(\tau), u(\tau))\, \mathrm{d}\tau \tag{6}$$

where $M(\cdot)$ and $L(\cdot)$ represent the Mayer and Lagrangian terms, respectively. For end-point optimization or end-point constrained problems, Eq. 6 typically simplifies to the Mayer form (i.e., $L = 0$), a functional term that explicitly involves initial and final conditions. The Lagrangian term, L, is frequently used to implement soft-constraints on the control rate or minimize deviations from nominal optimal state and input trajectories.

Remark 1. Note that evaluating the end-point constraint (Eq. 5) and the objective function in the above formulation (Eq. 6) requires state evaluation at batch termination. This, in turn, implies the non-linear model (Eq. 3) must be integrated at each sampling time to t_f and the input trajectory up to t_f must be computed by the optimizer at each time step. Consequently, both the integration and optimization become computationally expensive (whether using a sequential or simultaneous approach to solving the optimization problem). Note that while the solution to the optimization problem at a certain time step is a good initial guess for the next time step (exact-in the absence of uncertainties) modeling errors and disturbances may require significant computational effort in computing the manipulated input trajectory at every time step. We present in the next section an inherently different predictive controller formulation that does not require the online integration of the process model nor the online computation of the input trajectory up to the final time at each sampling instance, and thus, significantly reduces online computational demands (as expected much of the computational expense is transferred off-line; see the next section).

Reverse-Time Reachability Region Based MPC

In this section, we present a non-linear model predictive controller for batch processes. The key idea behind this design is to require the computation of only the immediate value of the manipulated variable while ensuring the desired end point properties remain attainable throughout the batch. Preparatory to the controller design, we first review the notion of reverse-time reachability regions, which are essential in the control design and analysis.

Reverse-time reachability regions

As previously discussed, the objective in batch processes is to reach a desired end-point, and of interest are the set of states from where the desired end-point can be reached. This set can be expressed in the form of reverse-time reachability regions, formally defined below:

Definition 1. *For the process described by* Eq. 1, *the reverse-time reachability region (RTRR) at time*, t, $R^{-1}(t)$ *is the set*:

$$R^{-1}(t) = \{x_0 | x(t) = x_0, u(\tau) \in \mathrm{U}\ \forall \tau \in [t, t_\mathrm{f}], \dot{x} = f(x,u), x(t_\mathrm{f}) = x_\mathrm{d}\} \quad (7)$$

The reverse-time reachability region at time t, $R^{-1}(t)$, therefore, consists of all process states from which the batch process can be steered to x_d, the desired end-point, by the end of the batch (i.e., in a time $t_\mathrm{f} - t$) while satisfying the input constraints. The reason behind naming this set the "reversetime reachability region" is as follows: Note that a reachability region, $R(t)$, for both batch and continuous processes is defined as the set of states that can be reached from a given initial condition in a time, t, subject to constraints on the manipulated inputs. If the "reverse time" version of the process is considered [i.e., $\dot{x} = -f(x,u)$], and the reachability region for this reverse-time process is computed (setting the initial condition as the desired end-point of the original process), this yields the set of states from where the desired end-point can be reached for the original process (and hence the name reverse-time reachability region). An algorithm to generate reverse-time reachability region and the associated computational issues are discussed in the following remarks.

Although the set is defined allowing for $u(t)$ to take values in U, computation of the reachability region can only be carried out by discretizing (in time) the control action (i.e., subject to a control action held constant for a pre-defined period of time). Below we define the discrete time version (where the control action is held for a time Δ till batch termination).

Definition 2. *The reverse-time reachability region at a time* $t = t_\mathrm{f} - k\Delta$ (*where* $k \leq \frac{t_\mathrm{f}-t_0}{\Delta} \in Z$), *indexed by* k *is defined as*:

$$R_k^{-1} = \{x_0 | x(t) = x_0,\ u_i \in \mathrm{U}\ \forall i = 0, \ldots, k, \dot{x} = f(x,u), x(t_\mathrm{f}) = x_\mathrm{d}\} \quad (8)$$

One way to compute R_k^{-1} is to scan the state space and test the feasibility of an optimization problem that requires the end-point constraint to be met subject to input constraints and to include every state for which the optimization has a feasible solution. However, the understanding of these sets as being the reverse-time reachability regions allows the sequential determination (of an estimate) of these sets without having to solve optimization problems. In particular, for a given x_d, the reverse-time process model can be integrated backwards in time for the duration of Δ, holding the value of the manipulated input constant. Performing this integration for all possible (appropriately discretized) values of the manipulated inputs in turn yields an (under) estimate of R_1^{-1} (the fact that the computation yields an under-estimate, however, does not negatively impact its use within the controller design; refer to remark 8 for a discussion). A finer discretization in terms of the manipulated input naturally yields a better estimate of the reverse-time reachability region. R_2^{-1} can in turn be determined by repeating the process for all elements in R_1^{-1}, and the process repeated to yield the reverse-time reachability region for the initial time.

A pseudocode on the construction of RTRRs is presented below to clarify the described algorithm above. Let the number of sampling instances during the batch be indexed by $k = 0,\ldots,K$ with $k = 0$ corresponding to t_f and $k = K$ corresponding to t_0. Additionally, let u_ALL be a $m \times N$ matrix where m denotes the number of inputs as before and N is the total number of possible input combinations following discretization of the m inputs. Finally, let x_d be a vector with dimensions $n \times 1$.

```
Set IC_0 = x_d
for k from 0 to K − 1:
    Set L as the current number of columns of IC_k
    for l from 1 to L
      for j from 1 to N
        Integrate reverse time model using jth column of
          u_ALL with lth column of L as initial conditions
        Store final conditions from integration as column
          in IC_{k+1}
    end
  end
end
```

Using the algorithm just described, the computational demands of generating reverse-time reachability regions increases when computing R_2^{-1} compared to R_1^{-1} (since R_2^{-1} is the set of initial conditions from where a state in R_1^{-1} can be reached, compared to R_1^{-1} which is the set of initial conditions from where a point, x_d can be reached). In general, the increase in computational demand is related to the increase in size of the reverse-time reachability regions, as we go back in time. However, it is worth noting that the size of the reverse-time reachability region does not necessarily grow as fast when going back in time for systems where the desired end-point is an equilibrium point (i.e., continuous processes). To understand this, consider the reverse-time reachability region for a desired end point which is an equilibrium point of the process (denoted by x_e). Every state in R_i^{-1} would be included in R_{i+1}^{-1} because it is possible for the process to stay at x_e for another Δ, having reached x_e in i time steps (by virtue of x_e being an equilibrium point). This is not true when the desired end-point is not an equilibrium

point, and therefore, the size of the reverse-time reachability regions, and therefore, the associated computational cost, does not prohibitively increase as we proceed further towards the initial time.

In addition to being dependent on the size of the previously generated RTRR, the computational demands of the generation algorithm is also generally dependent on the number of states and inputs in the system. For systems with a high number of state variables, the required computational effort is influenced by the efficiency of the integrator as well as its ability to handle large scale systems. In these cases, one of the host of efficient large scale integrators available in the public domain (i.e., DASSL, CVODE, etc.) can be utilized. While increasing the number of states makes the RTRR generation more complex through internal computations (i.e., the Jacobian) performed by the integrator, the number of inputs affects the number of necessary integrations. The number of integrations, in fact, grows exponentially with the number of manipulated inputs. For instance, consider a system with two bounded inputs, which are both discretized at 10 different points that span the bounds. Therefore, the number of required integrations to generate R_1^{-1} is 100 as the reverse-time model of the system must be integrated for every discretization point. Note, however, that the generation algorithm has an important feature in that integrations of the model equations may be done independently for different values of the input and initial conditions. Accordingly, to alleviate potential computational issues associated with having a high number of inputs, starting from a given reverse-time reachability region, integrations of the reverse-time model may be done independently, which in turn implies parallel computing schemes can be readily used to generate reverse-time reachability regions to significantly reduce the computation times.

Remark 2. The RTRR generation algorithm outlined above describes how RTRRs can be constructed via only integrations of the reverse time model of the system, $\dot{x} = -f(x,u)$. With the assumptions that $-f(x, u)$ is continuous on (x, u) and is locally Lipschitz in x on $D \times \mathrm{U}$, where $D \subset \mathbb{R}^n$, the continuity of solutions of $\dot{x} = -f(x,u)$ in terms of the initial condition and input is guaranteed. As a result, these assumptions ensure that RTRRs generated at each time step will be compact sets. While these continuity assumptions ensure against disjointed sets, no such general assumptions can be made to guarantee the convexity of RTRRs generated at each time step. Note that we do not require the sets to be convex for the implementation of our results. Also, note that for batch processes which can be approximated well using convex differential equations in (x,u) (i.e., linear or linearized models), RTRRs generated via the algorithm in this section will be convex and this can lead to a significant reduction in the computational requirements for generating RTRRs. For the current work, however, we focus on non-linear processes and only the continuity of the model equations is assumed.

Remark 3. Although existing Lyapunov-based control designs can be very well used in the context of tracking state trajectories in batch processes, the fact that the desired end point in a batch is typically not an equilibrium point precludes the use of Lyapunov-based techniques to determine the set of initial conditions from where a desired end-point can be reached in finite time. To begin with, the basic assumption in Lyapunov-based control designs, that of $f(x_d, 0) = 0$ is not satisfied in the case of batch processes. Note that this cannot be achieved by a co-ordinate transformation either, because x_d is simply not an equilibrium point of the process. Of course a positive definite function $V_{\text{candidate}}$ can be defined such that $V_{\text{candidate}}(x_d) = 0$. The set of states for which $\dot{V}_{\text{candidate}}$ can be made negative, however, does not form a neighborhood around x_d, which in turn precludes the construction of "invariant" sets around x_d. In summary, in contrast to continuous processes where the desired operating point is an equilibrium point, Lyapunov-based techniques do not allow for computing the set of states from where the process can be guaranteed to be steered towards the desired end-point.

Remark 4. Although seemingly conceptually similar, the reverse-time reachability regions in the context of continuous systems are inherently different from those in the context of batch processes. In particular, when considering stabilization to an equilibrium point, the reverse-time reachability regions, with the time tending to infinity, yield the so called null-controllable regions (the set of initial conditions from where a process can be stabilized at an equilibrium point). For stabilization at an equilibrium point, $R^{-1}(t_2) \subset R^{-1}(t_1)$ when $t_2 < t_1$. To understand this, consider the set of states which constitute $R^{-1}(t_1)$; there naturally exists a subset of states within $R^{-1}(t_1)$ that can be steered to x_d in a time $t_f - t_1$ (since $t_f - t_1$ is less than $t_f - t_2$) and simply kept there until t_f as x_d is an equilibrium point. In contrast, in the context of a desired end-point that is not an equilibrium point, just because a point is in $R^{-1}(t_2)$ does not ensure that it is in $R^{-1}(t_1)$ since the process cannot be "maintained" at x_d.

Remark 5. The presence of constraints on the manipulated input has significant implications on the ability to control continuous (e.g., Ref. 29) as well as batch systems. Despite their differences, one common property among null controllable regions for continuous systems and reverse-time reachability regions in the context of batch systems is that they are both dependent only on system dynamics and input constraints and do not depend on a specific control law.

MPC formulation

An MPC formulation that utilizes the reverse-time reachability region characterization is presented in this section. To this end, consider the process described by Eq. 1, for which the reverse-time reachability regions have been characterized for a given hold and implement time Δ. The control action at a time t [with $k = (t_f - t)/\Delta$] is computed by solving the optimization problem below:

$$u_{\text{MPC}}(x) := \arg\min\{J(x,t,u(\cdot))|u \in S\} \tag{9}$$

subject to:

$$\dot{\tilde{x}} = f(\tilde{x}(\tau), u) \tag{10}$$

$$\tilde{x}(0) = x(t) \tag{11}$$

$$\tilde{x}(\Delta) \in R_{k-1}^{-1} \tag{12}$$

The objective function can be appropriately chosen to meet desired performance objectives. For instance, if the

objective is to minimize discrepancies between actual state trajectories and some nominal optimal state trajectories (x^*_{nom}), and the variation in the control moves, the performance objective could be formulated as:

$$J = \int_t^{t+\Delta} \left\| \tilde{x}(\tau) - x^*_{\text{nom}} \right\|_Q + \| u_k - u_{k-1} \|_R \qquad (13)$$

where Q and R are weighting matrices.

As evidenced by the constraint shown in Eq. 12, implementation of the reverse-time reachability region based controller necessitates an explicit characterization of reachability regions. In this manuscript, the constraint in Eq. 12 is chosen to be represented as an inequality constraint [i.e., $\tilde{x}(\tau), u(\tau) \leq 0$; see Remark 8]. Note that by definition, reverse-time reachability regions take into account desired end properties of a batch. Consequently, any existing end-point constraints in the end-point based MPC scheme can be replaced with a constraint that requires, at each sampling instance, the process states to remain in the reverse-time reachability region at the next time step (Eq. 12). Implications on the guarantees of reachability to the desired end point through this replacement are formalized in Theorem 1 below.

Theorem 1. *Consider the constrained system of* Eq. 1 *with* $\sigma = 1$ *under the MPC law of* Eqs. 9–12. *If and only if* $x(t_0) \in R^{-1}(t_0)$, *the optimization problem defining the MPC law in* Eqs. 9–12 *remains feasible for all* $t \in [t_0, t_f]$ *and* $x(t_f) = x_d$.

Proof of theorem 1. We first show the only if part of the theorem. To this end, consider an initial condition $x_0 \notin R^{-1}(t_0)$. If the constraint of Eq. 12 is feasible and it is implemented in the closed-loop, then there exists a sequence of control moves that take the state to x_d by t_f, in turn implying that $x_0 \in R^{-1}(t_0)$. This argument can be repeated at every time step, eventually leading to the MPC laws in Eqs. 9–12 remaining feasible for all $t \in [t_0, t_f]$ and $x(t_f) = x_d$ only if $x_0 \in R^{-1}(t_0)$. We now show the sufficiency of the condition. To this end, consider the case when $x(t_0) \in R^{-1}(t_0)$. By definition, there exists a set of control moves that takes the state to x_d by t_f. For such a set of control moves (and the associated state trajectory), this must imply that the state trajectory at $t_0 + \Delta$ resides in $R^{-1}(t_0+\Delta)$ [invoking the necessity of the condition proved earlier for $x(t_0 + \Delta)$]. In essence, this implies that there exists a feasible solution to the constraint of Eq. 12. This completes the proof of Theorem 1. ■

Remark 6. The statement of Theorem 1 essentially formalizes that the existence of the state in the reverse-time reachability region is a necessary and sufficient condition for the states to be steered to the desired end point. The necessity of the condition has an important implication in that if at any time during the batch, the states are driven outside the reverse-time reachability region, the desired end-point simply cannot be reached. In other words, the condition of continued existence in successive reverse-time reachability region cannot be relaxed because if the state goes outside the reverse-time reachability region, it is simply not possible (whether using the proposed reverse-time reachability region based predictive controller or any other control law) to steer the states back into the reachability region and then to the desired end-point by the batch termination time.

Remark 7. In this work, we assume that the batch process design includes a specification for the desired end point properties, x_d. This allows for the computation of reverse-time reachability regions using the algorithm presented earlier in this section. In some cases, not all the desired state values at batch termination are explicitly constrained at the process design phase. In these cases, the objective may be to maximize or minimize a certain product concentration. Accordingly, for these cases, a nominal optimization problem can be solved off-line with a given performance objective, and x_d can be taken to be the state variable vector at t_f from the optimal state trajectories. Additionally, rather than use a single process state value as the desired end point, a suitable neighborhood around x_d, B_{x_d}, can be the desired neighborhood to be reached. The proposed MPC naturally allows for incorporating such a scenario by computing $R^{-1}(t_f - \Delta)$ as the set of states that can be steered to the neighborhood B_{x_d} and successively continuing backwards.

Remark 8. When using the algorithm outlined for generating the reverse-time reachability regions, the true reverse-time reachability regions are estimated as point sets. Depending on the specific system under investigation, the shape and orientation of the point set may permit different strategies for explicitly characterizing the point set. In any case, the explicit characterization must be either an exact characterization, which is unlikely, or an underestimate of the true region. With the characterizations that represent an overestimate, process states may be incorrectly identified as belonging to the true reverse-time reachability region. Hence, the constraint in Eq. 12 may be satisfied initially even when the states are not contained in the true reverse-time reachability region, invalidating guarantees of successive feasibility. In contrast, if the explicit characterization is an underestimate (generated appropriately), successive feasibility of the optimization problem can be still guaranteed.

Remark 9. One approach to generate the approximate characterization of reverse-time reachability regions is to consider 2D projections of the reverse-time reachability regions and posing it as a geometric optimization problem and using polynomial discrimination as the explicit characterization method. Briefly, polynomial discrimination finds the t-level set of a polynomial with degree less than or equal to d, $\{x | \sum_{i_1+\cdots+i_n \leq d} a_{i_1 \ldots i_d} x_1^{i_1} x_n^{i_n} = t\}$, that discriminates two sets of points.[30] An underestimate of the true reverse-time reachability regions can be obtained through specifying an interior subset of the point set as one class of points and the complements as the other class. Such subsets can be identified using, e.g., the hidden point removal (HPR) algorithm presented in.[31] In the HPR algorithm for 2D point sets, points visible from a certain perspective location can be identified. Thus, by appropriately selecting the perspectives from which to view the point set, boundary points of the set can be identified and removed. It should be noted that the distance between the view point and point set should also be chosen carefully, and the results[31] include an optimal way to choose such a distance such that the number of false negatives is minimized. By successively applying the HPR algorithm to a 2D point set and removing boundary points at each iteration, a strictly interior subset of the point set can be identified. As

the results from the HPR algorithm are not exact, visual verification can be used to ensure that strictly interior points are being identified with the HPR algorithm. Geometrically, polynomial discrimination seeks an algebraic surface (as defined by the degree and coefficients of the polynomial) that can separate the two classes. If it is observed that the two classes cannot be separated by a polynomial of a given degree, one natural step to take is to increase the degree of the polynomial and retry the fit. The details regarding polynomial discrimination and approximate discrimination approaches are available.[30]

Remark 10. With the attainment of the desired end-point achieved via the constraint, the objective function in the MPC formulation can be utilized to satisfy performance (as shown in Eq. 13) or even robustness objectives. Specifically, to enhance disturbance rejection and robustness of the closed loop batch system, the objective function can be used to penalize the Euclidean distance between a process state and the analytical center of a reverse-time reachability region. This would tend to drive the process state towards the center of the reverse-time reachability region, thereby reducing the chances of disturbances driving the process to a point from where the end-point is no longer reachable. Also worth noting is that it is possible to employ a multi-objective function, which consists of a weighted sum of performance and robustness terms.

Safe-Steering of Batch Processes

In the previous section, we presented a reverse-time reachability region based predictive controller that was designed with a fault-free assumption. As discussed earlier, in contrast to continuous processes, the problem of fault tolerance is fundamentally different in batch processes. Specifically, a fault tolerant control framework for batch processes must be designed with the desired end point in mind, which is rarely an equilibrium point. In this section, we utilize the reverse-time reachability region based MPC scheme to develop what we call the safe-steering framework. To this end, the safe-steering problem is initially formulated and then the safe-steering framework is presented.

Problem definition

We consider the class of faults where a control actuator fails and reverts to its fail-safe value. Failsafe positions are intended to minimize the possibilities of excursions to dangerous conditions such as high temperatures and pressures. Examples of fail-safe positions are fully open for a coolant valve and fully closed for a steam flow valve. More specifically, w.l.o.g., we characterize the fault occurring in the first control actuator at a time t^{fault}, which is subsequently rectified at a time, t^{recovery} (i.e., for $t \leq t^{\text{fault}}$ and $t \geq t^{\text{recovery}}$, $\sigma(t) = 1$ while for $t^{\text{fault}} \leq t \leq t^{\text{recovery}}$, $\sigma(t) = 2$) as $u_2^1(t) = u_{\text{failed}}^1$ $\forall t \in [t^{\text{fault}}, t^{\text{recovery}}]$, with $u_{\min}^1 \leq u_{\text{failed}}^1 \leq u_{\max}^1$, where u^i denotes the ith component of the input vector u, the only inputs available for control between t^{fault} and t^{recovery} are u_2^i, $i = 2,\ldots,m$. Note that if t^{fault} is at a time which is not an integer multiple of the batch sampling time, Δ, t^{fault} can be taken to be at the upcoming integer multiple of Δ, and the safe-steering framework can be implemented as presented in this section. With one of the inputs fixed at a fail-safe position, while it may still be possible to reach the desired end-point if the fault is rectified sufficiently fast, continued operation of the representative end-point based MPC scheme may drive the states to a point from where it is not be possible to reach the desired end point even upon fault-repair. We define the safe-steering problem as the one of identifying trajectories of the functioning inputs between the time of fault occurrence and repair (without requiring the value of t^{recovery}, or an estimate thereof, to be known a-priori) that will preserve reachability to the desired end-point properties.

Safe-steering to desired end point properties

The main requirement for safely steering a batch during fault rectification is to repeatedly maintain states in regions from which the desired end point remains reachable. To this end, the reverse-time reachability region based predictive controller can be utilized. Consider the batch system described by Eq. 1 for which the first control actuator faults at t^{fault} and is repaired at t^{recovery}. Also, assume the reverse-time reachability regions for fault-free operation have been characterized for all sampling instances between t^{fault} and t^{recovery}. The following theorem addresses the safe-steering problem.

Theorem 2. *Consider the constrained system of* Eq. 1 *under the reverse-time reachability region based model predictive controller given by* Eqs. 9–12. *In the event of a fault at* t^{fault}, if and only if the optimization problem remains feasible $\forall t \in [t^{\text{fault}}, t^{\text{recovery}}]$, *then the optimization problem continues to be feasible* $\forall t \in [t^{\text{recovery}}, t_{\text{f}}]$ and $x(t_{\text{f}}) = x_{\text{d}}$.

Sketch of Proof of Theorem 2: The proof of this theorem follows from Theorem 1. Essentially equating the time t^{recovery} to t_0, results in satisfaction of the requirements of Theorem 1 and the optimization problem continues to be feasible $\forall t \in [t^{\text{recovery}}, t_{\text{f}}]$ and $x(t_{\text{f}}) = x_{\text{d}}$ follows. This completes the proof of Theorem 2. ■

Remark 11. The key idea in the safe-steering framework, as formally expressed in Theorem 2, is to continuously maintain the process (if possible) in the reverse-time reachability regions after t^{fault}. If the reverse-time reachability region based controller continues to be feasible after t^{fault} up until the fault gets rectified at t^{recovery}, this implies that at t^{recovery}, the states will be contained within $R^{-1}(t^{\text{recovery}})$, and according to the definition of reverse-time reachability regions, the desired endpoint will be reachable. Note that during the fault-rectification period, there is no guarantee that the states can be maintained within the (fault-free) reachability regions. The result of Theorem 2, therefore, relies on the assumption of continued feasibility of the optimization problem after t^{fault}. This assumption, however, is not restrictive for the reason that it simply cannot be relaxed. If any time during fault-repair, the state moves outside the reverse-time reachability region, the end point state properties cannot be reached even if the fault is immediately repaired.

Remark 12. Note also that the key reasoning behind the statement of Theorem 2 is as follows: Once a fault takes place, since the desired end-point cannot be reached with the limited available control action, if one continues to try and implement the end-point based MPC it would yield an infeasible solution. (Note that if the end-point based MPC continues to be feasible following a fault, then it could be continued to be used for the duration of the batch. This work looks

at the specific class of input failures for which an end-point based MPC will become infeasible due to the limited available control action.) With the limited control action available, while it may not be possible to drive the state to the desired end-point, it may still be possible to maintain the state for some time in the successive reverse-time reachability regions. Implementation of a truncated version of the infeasible solution from an end-point based MPC may not retain the states in the reverse time-reachability region, however, the implementation of the reverse-time reachability region based MPC (if feasible) allows for finding such a control action that preserves the states within the reverse time-reachability regions and subsequent end point being reached upon fault-recovery. Note that if the fault-repair time was known ahead of time (which of course, is not) then in principle the end-point based MPC would also have been able to find the manipulated input trajectory utilizing the reduced available control action between t^{fault} and t^{recovery}), and full control action thereafter to achieve the desired end-point. The utilization of the reverse-time reachability region based MPC allows to achieve this without the need for knowing t^{recovery} ahead of time.

Remark 13. The safe-parking framework and safe-steering framework for continuous and batch systems, respectively, both address the problem of how to operate a plant during fault recovery and both frameworks address the kind of faults that prevent the desired plant operation under nominal control laws. Safe-parking for continuous processes handle faults that preclude operation at a nominal equilibrium point while safe-steering for batch processes addresses the kind of faults that preclude driving the states to the desired end point. The safe-steering and safe-parking framework both answer the question of how to compute the input trajectories between fault occurrence and recovery such that the desired nominal operation can be preserved or resumed. This is where the similarity between the approaches ends. As discussed previously in the introduction, in the presence of faults that prevent operation at a desired equilibrium point, safe-parking[23,24] involves transition to a new safe-park (equilibrium) point which allows the transition back to the nominal equilibrium in some optimal way following fault recovery. Before to considering any optimality criteria regarding the transitions, the safe-parking framework must locate the feasible operating points which allow such transitions, and the main criteria used in locating these "safe-park points" is the existence of such equilibrium points and then preserving process stability by utilizing stability region characterization[32,33] for the nominal and safe-park points. In particular, the nominal equilibrium point must be contained within the stability region of the safe-park point and vice versa. Neither the safe-park points, nor the controller designs for continuous systems or the associated stability regions remain applicable in the context of batch processes. In particular, the primary concern in the safe-steering framework is reachability. In the absence of equilibrium point, the process cannot be "parked", but possibly "steered" in a way that allows for end-point reachability if the fault is rectified sufficiently fast. This is achieved in the safe-steering framework via using the proposed reverse-time reachability region based MPC.

Remark 14. Although in this work we assume full state availability to enable clear presentation of the key ideas, the framework can readily incorporate the use of state observers that utilize the measurements to generate estimates of the state which can in turn be used for implementing the proposed safe-steering mechanisms. The effect of possible errors in the state estimates can be mitigated by requiring the states (predicted using the estimates) to go to subsets of the reversetime reachability regions. A detailed analysis of the output feedback problem, however, remains outside the scope of the present work.

Remark 15. Note that the RTRR regions are estimated by integrating the process model equations. Consequently, the shapes and sizes of the estimated regions are sensitive to modeling errors. Because of the possibility of discrepancies between the estimated and true RTRR sets in the presence of modeling errors, there is the potential of misclassifying batch conditions as being contained within a true RTRR of the process. In such cases, when process states have escaped the true RTRR (even though the states may be classified as lying within the estimated RTRR), reaching the desired end point becomes impossible. One potential solution to this problem is to utilize more conservative characterizations of RTRRs to eliminate the possibility of misclassification (based on bounds on the modelling errors, if known). A detailed exploration of the robustness of the proposed framework and the sensitivity of RTRR regions to modeling errors is outside the scope of the current work but is an important topic of our future work.

Remark 16. Note, also, that the framework can be utilized to exploit scenarios where the batch termination time is allowed to be changed, or at the very least, allowed to be changed over a reasonable range. This can be achieved as follows: at the time of fault repair, if the state is not present in the reverse-time reachability region to go to the desired end-point by batch termination (due to taking it exceedingly long for fault rectification), the explicit characterization of the reverse-time reachability region can be utilized to compute the value of the (least) batch termination time for which the state would reside in the reverse-time reachability region. If such a time exists, the batch can possibly be run for a longer duration (if economic and operation constraints permit) to yield the desired product. If no such time exists, this points to the fact that the batch may have gone beyond redemption, and again helps in taking the immediate (and necessary) action of discarding the batch, instead of unsuccessfully trying to reach the desired end point and wasting valuable time.

Simulation Example

We illustrate in this section the proposed reverse-time reachability region-based MPC and the safe-steering framework via a fed-batch process. To this end, consider an isothermal fed-batch fermentation process for which the mathematical model takes the following form:

$$\dot{x}_1 = \frac{\mu_{\max} x_2 x_1}{K_1 + x_2 + K_2 x_2^2} - \frac{x_1}{x_3}(F_1 + F_2) \tag{14}$$

$$\dot{x}_2 = -\frac{\mu_{\max} x_2 x_1}{Y\left(K_1 + x_2 + K_2 x_2^2\right)} + \frac{S_f - x_2}{x_3}(F_1 + F_2) \tag{15}$$

$$\dot{x}_3 = F_1 + F_2 \tag{16}$$

where x_1, x_2, and x_3 denote the biomass concentration, substrate concentration, and volume (respectively) and con-

Table 1. Process Parameters for the Fedbatch Bioreactor of Eqs. 14–16

Parameter	Description	Value	Unit
μ_{max}	Maximum growth rate	25	h^{-1}
K_1	Monod constant	0.03	g/L
K_2	Kinetic parameter	0.25	L/g
Y	Yield coefficient	0.25	
S_f	Influent substrate concentration	10	g/L

stitute the state vector, x. The physical meaning of the parameters in the model equations and their nominal values can be found in Table 1. The primary control objective for this fed-batch process is to achieve the arbitrarily chosen desired end point of $x_d = [8\ 7.5\ 30]^T$. The fermentation time is taken to be 1 h with a controller execution period of 0.05 h. The manipulated variables in the fermentation process are the two flow rates, F_1 and F_2, with constraints: $0 \leq F_1 \leq 10$ L/h and $0 \leq F_2 \leq 10$ L/h.

Calculation of reverse-time reachability region

The reverse-time reachability regions, $R^{-1}(t)$, for this system are first generated as three-dimensional point sets using the procedure described earlier in the Reverse time Reachability Regions section. To aid in the visualization and the eventual explicit characterization, the point set corresponding to a given reverse-time reachability region is first projected onto three two-dimensional spaces, namely the $x_1 - x_2$, $x_1 - x_3$, and $x_2 - x_3$ spaces. Thus, $R^{-1}(t)$ is characterized through $R^{-1}_{x1-x2}(t)$, $R^{-1}_{x1-x3}(t)$, and $R^{-1}_{x2-x3}(t)$, where the subscripts denote the states considered in each projection. At a given point in time, $x(t)$ is considered to be within $R(t)$, only if the three projected reverse-time reachability regions contained the corresponding projections of $x(t)$ simultaneously. Ellipses were utilized for the explicit characterization of the two-dimensional (2D) point sets in the three projections. Figures 1 and 3 illustrate the results of using ellipses for all three projections at two distinct times in the batch, $t = 0.40$ h and 0.85 h.

Implementing the reverse-time reachability region based MPC

The effectiveness of the representative end-point controller and the reverse-time reachability region based controller are compared first during fault free operation. The objective functions used in the end-point and reverse-time reachability region based formulations, J_{EP} and J_R, were:

$$J_{EP} = \int_t^{t_f} \Delta u dt \tag{17}$$

$$J_R = \left\| x_{1,2}(t+\Delta) - x^c_{1,2}(t+\Delta) \right\| + \left\| x_{1,3}(t+\Delta) - x^c_{1,3}(t+\Delta) \right\| + \left\| x_{2,3}(t+\Delta) - x^c_{2,3}(t+\Delta) \right\| \tag{18}$$

where $x_{i,j}(t)$ represents the projection of the states at time t onto the $x_i - x_j$ space and $x^c_{i,j}(t)$ represents the center point of the reverse-time reachability region at time t in the $x_i - x_j$ space. The initial condition for both controllers was specified to be $x(0) = [0.7853\ 50.7515\ 20.1629]^T$. This initial condition was verified to be in $R^{-1}(0)$, guaranteeing the desired end-point is reachable in a fault-free batch.

The state and input profiles resulting from the implementation of both controllers are presented together in Figure 2. In both cases, the desired end-point properties of the batch are reached at batch termination, meeting the primary control objective. Note that the input variation observed for the reverse-time reachability region based controller could have been reduced further via a penalty term for Δu. Using the MATLAB functions, tic and toc, the required run times on an Intel P4 3.0 Ghz machine for the end-point based predictive controller (with a sequential solution strategy) and the reverse-time reachability region based controller were 30.47 min and 26.91 min, respectively.

The significant reduction in the computational effort from the reverse-time reachability region based MPC scheme is evident from these results. For batch processes with additional inputs and states and/or a longer duration, the integration and optimization routines in an end-point based MPC strategy will demand additional computational effort; therefore, the reduction in the run times will be even more substantial with reverse-time reachability region based MPC.

Safe-steering of the fed-batch process

Having demonstrated the computational efficiency of the reverse-time reachability region based MPC, we now consider the application of the two MPC schemes in the presence of faults. To this end, consider the case where at $t = 0.4$ h, the valve regulating F_1 fails and reverts to its fail-safe position (completely shut), resulting in $F_1 = 0$. Thus, the inputs to the system become restricted to $F_1 = 0$ and $0 \leq F_2 \leq 10$. At $t = 0.85$ h, the fault is repaired. The state and input profiles resulting from the use of an end-point based predictive controller and the reverse-time reachability region based controller are presented in Figure 4.

We observe that even after fault recovery, the end-point based MPC scheme is not able to drive the process to the desired end-point states. Instead, due to repeated

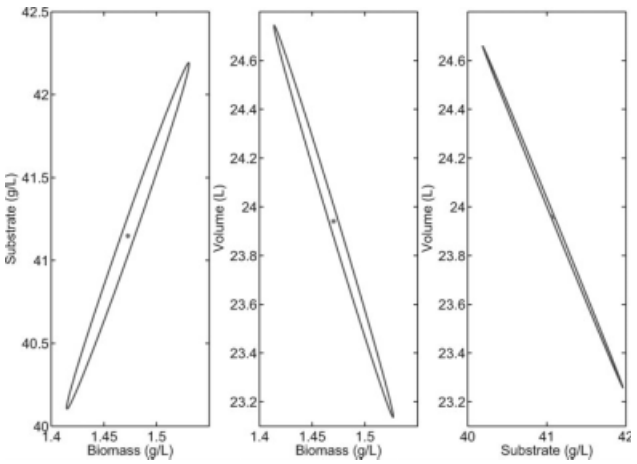

Figure 1. Reverse time reachability region projections at $t = 0.4$ h.

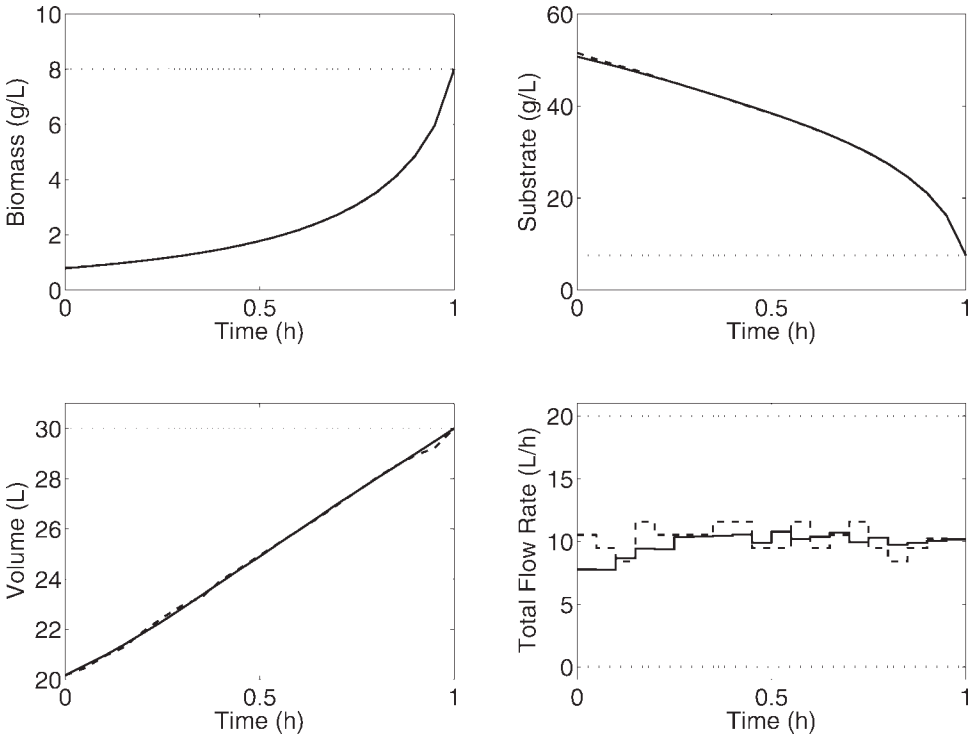

Figure 2. State and input profiles of the fed-batch bioreactor system under the end point based MPC (solid) and reverse-time reachability region based MPC (dashed) schemes during a fault-free batch.

Both controllers successfully drive the process to the desired end-point, but the reverse-time reachability region based MPC takes significantly less online computational effort.

infeasibilities during fault repair ($t^{\text{fault}} < t < t^{\text{recovery}}$) as a consequence of limited available control effort (with the controller requiring the states to go to the desired end point by batch termination using the reduced control effort for the entire remaining batch duration), the controller steers the system to the end point of $x(t_f) = [8.0634\ 7.6367\ 29.6084]^T$. The states at the time of fault-recovery for the end-point based MPC are $x(t = 0.85) = [4.1155\ 24.7323\ 28.3706]^T$. Examining the reverse-time reachability regions at $t = 0.85$ h, this process state is not included in any of the three projections of the reverse-time reachability region (see Figure 3). Hence, upon fault recovery, the end-point based MPC scheme prescribes extreme control action in an (unsuccessful) attempt to arrive at the desired end point by the end of the batch. In contrast, in the reverse-time reachability region based MPC scheme, there is no need to resort to extreme values of the input to recover reachability to the desired end point because the input moves during fault duration are computed to preserve reachability. Although the end-point based MPC scheme ends up implementing control action that drives the states to a point by $t = 0.85$ h from where it is no longer possible to reach the desired end-point, the reverse-time reachability region based controller, by only considering one time step ahead, remains feasible during the failure period, ensuring the states evolve within reverse-time reachability regions. Consequently, the desired end point is reachable upon fault rectification. Note that if the fault was not rectified by $t = 0.85$ h, neither the end-point based MPC nor the reverse-time reachability region based MPC would have been able to reach the desired end-point. However, the key point is that in this case (with the fault being rectified at $t = 0.85$ h), there exists a set of control moves utilizing reduced control action during faulty operation and full control action

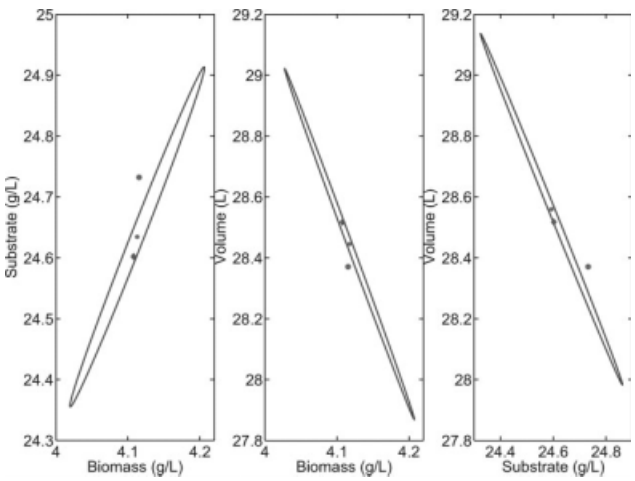

Figure 3. Reverse time reachability region projections at $t = 0.85$ h.

The ○ denotes the state values at t^{recovery} using the end-point based MPC while the ◇ shows the state values using the proposed reverse-time reachability region based MPC.

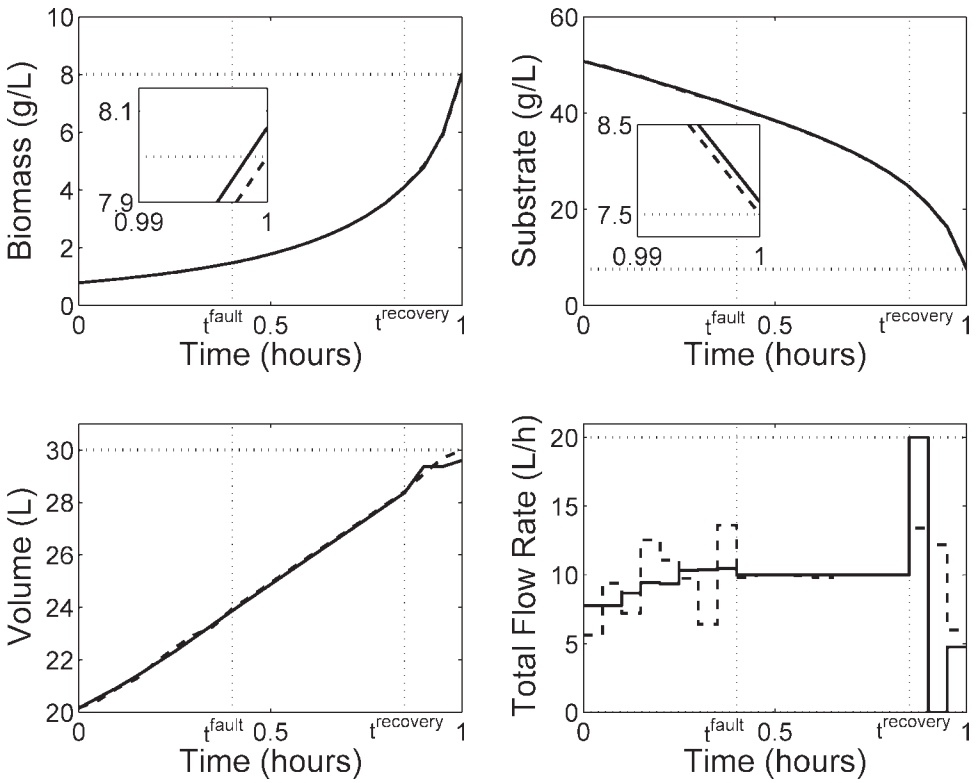

Figure 4. State and input profiles of the fed-batch bioreactor system under the end point based MPC (solid) and reverse-time reachability region based MPC (dashed) schemes with input failure of F_1 between 0.4 and to 0.85 h.

The insets show the desired end-point being reached using the proposed safe-steering framework.

after fault repair that allows reaching the desired end-point, and the reverse-time reachability region based MPC is able to find such a manipulated input trajectory (without a-priori knowing t^{recovery}).

Conclusions

This work considered the problem of control of non-linear batch processes to achieve a desired final product quality subject to input constraints and faults in the control actuators. Specifically, faults were considered that cannot be handled via robust control approaches, and preclude the ability to reach the desired end-point, necessitating fault-rectification. A safe-steering framework was developed to address the problem of determining how to utilize the functioning inputs during fault rectification to ensure that after fault-rectification, the desired product properties can be reached upon batch termination. To this end, first a novel reverse-time reachability region based MPC was formulated that reduces online computations, as well as provides a useful tool for handling faults. Next, a safe-steering framework was developed that utilizes the reverse-time reachability region based MPC in steering the state trajectory during fault rectification to enable (upon fault recovery) the achieving of the desired end point properties by batch termination. The online efficiency of the proposed controller and its utility in the context of safe-steering framework were illustrated using a fed-batch process example.

Acknowledgments

Financial support from the Natural Sciences and Engineering Research Council of Canada through a PGS(D) award is gratefully acknowledged.

Literature Cited

1. Mhaskar P, Aumi S. Transition from batch to continuous operation in bio-reactors: a model predictive control approach and application. *Can J Chem Eng*. 2007;45:416–423.
2. Cruickshank SM, Daugulis AJ, McLellan PJ. Dynamic modeling and optimal fedbatch feeding strategies for a two-phase partitioning bioreactor. *Biotech Bioeng*. 2000;67:224–233.
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 part 1. Theoretical framework. *Ind Eng Chem Res*. 1993; 32:866–881.
5. Soroush M, Kravaris C. Optimal-design and operation of batch reactors part 2. A case-study. *Ind Eng Chem Res*. 1993;32:882–893.
6. Yabuki Y, Nagasawa T, MacGregor JF. An industrial experience with product quality control in semi-batch processes. *Comput Chem Eng*. 2000;24:585–590.
7. Hua XM, Rohani S, Jutan A. Cascade closed-loop optimization and control of batch reactors. *Chem Eng Sci*. 2004;59:5695–5708.
8. Shi D, Mhaskar P, El-Farra NH, Christofides PD. Predictive control of crystal size distribution in protein crystallization. *Nanotechnology* 2005;16:S562–S574.
9. 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
10. 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.

11. Palanki S, Vemuri J. Optimal operation of semi-batch processes with a single reaction. *Int J Chem Reac Eng*. 2005;3.
12. 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.
13. Huynh N, Kazantzis N. Parametric optimization of digitally controlled nonlinear reactor dynamics using zubov-like functional equations. *J Math Chem*. 2005;38:499–519.
14. Mhaskar P. Robust model predictive control design for fault-tolerant control of process systems. *Ind Eng Chem Res*. 2006;45:8565–8574.
15. Wang ZD, Huang B, Unbehauen H. Robust reliable control for a class of uncertain nonlinear state-delayed systems. *Automatica* 1999;35:955–963.
16. Cheng LL, Kwok KE, Huang B. Closed-loop fault detection using the local approach. *Can J Chem Eng*. 2003;81:1101–1108.
17. Mehranbod N, Soroush M, Panjapornpon C. A method of sensor fault detection and identification. *J Proc Contr*. 2005;15:321–339.
18. Mhaskar P, Gani A, El-Farra NH, McFall C, Christofides PD, Davis JF. Integrated fault-detection and fault-tolerant control for process systems. *AIChE J*. 2006;52:2129–2148.
19. El-Farra NH. Integrated fault detection and fault-tolerant control architectures for distributed processes. *Ind Eng Chem Res*. 2006;45: 8338–8351.
20. Mhaskar P, McFall C, Gani A, Christofides PD, Davis JF. Isolation and handling of actuator faults in nonlinear systems. *Automatica* 2008;44:53–62.
21. El-Farra NH, Giridhar A. Detection and management of actuator faults in controlled particulate processes using population balance models. *Chem Eng Sci*. 2008;63:1185–1204.
22. Armaou A , Demetriou MA. Robust detection and accommodation of incipient component and actuator faults in nonlinear distributed processes. *AIChE J*. 2008;54:2651–662.
23. Gandhi R, Mhaskar P. Safe-parking of nonlinear process systems. *Comp Chem Eng*. 2008;32:2113–2122.
24. 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.
25. Alamir M, Balloul I. Robust constrained control algorithm for general batch processes. *Int J Contr*. 1999;72:1271–1287.
26. Terwiesch P, Agarwal M, Rippin DWT. Batch unit optimization with imperfect modelling: a survey. *J Proc Contr*. 1994;4:238–258.
27. Undey C, Ertunc S, Cinar A. Online batch/fed-batch process performance monitoring, quality prediction, and variable-contribution analysis for diagnosis. *Ind Eng Chem Res*. 2003;42:4645–4658.
28. Undey C, Tatara E, Cinar A. Intelligent real-time performance monitoring and quality prediction for batch/fed-batch cultivations. *J Biotech*. 2004;108:61–77.
29. Kapoor N, Daoutidis P. Stabilization of nonlinear processes with input constraints. *Comp Chem Eng*. 2000;24:9–21.
30. Boyd S, Vandenberghe L. *Convex Optimization*. New York, USA, Cambridge University Press, 2004.
31. Katz S, Tal A, Basri R. Direct visibility of point sets. *ACM Trans Graphics* 2007;26:1–11.
32. 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.
33. Mhaskar P, El-Farra NH, Christofides PD. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Syst Contr Lett*. 2006;55:650–659.

Manuscript received Nov. 10, 2008, and revision received Mar. 18, 2009.