MODELLING AND CONTROL
of
BATCH PROCESSES

MODELLING AND CONTROL

of

BATCH PROCESSES

Siam Aumi, B. Eng., (Chemical Engineering)

November 2011

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

to the

Department of Chemical Engineering
Faculty of Engineering

Title: Modelling and control of batch processes

Author: Siam Aumi, B. Eng., (Chemical Engineering)
McMaster University, Hamilton, ON, Canada

Degree (year): *Doctor of Philosophy* (2011)

Department: Chemical Engineering

Supervisor: Dr. Prashant Mhaskar

Number of pages: xviii | 122

To my parents, *Shamim* and *Kazi*, for their sacrifices, support, and love

Abstract

This thesis considers the problems of modelling and control of batch processes, a class of finite duration chemical processes characterized by their absence of equilibrium conditions and nonlinear, time-varying dynamics over a wide range of operating conditions. In contrast to continuous processes, the control objective in batch processes is to achieve a non-equilibrium desired end-point or product quality by the batch termination time. However, the distinguishing features of batch processes complicate their control problem and call for dedicated modelling and control tools.

In the initial phase of this research, a predictive controller based on the novel concept of reverse-time reachability regions (RTRRs) is developed. Defined as the set of states from where the process can be steered inside a desired end-point neighbourhood by batch termination subject to input constraints and model uncertainties, an algorithm is developed to characterize these sets at each sampling instance *offline*; these characterizations subsequently play an integral role in the control design. A key feature of the resultant controller is that it requires the online computation of only the immediate control action while guaranteeing reachability to the desired end-point neighbourhood, rendering the control problem efficiently solvable even when using the nonlinear process model. Moreover, the use of RTRRs and one-step ahead type control policy embeds important fault-tolerant characteristics into the controller.

Next, we address the problem of the unavailability of reliable and computationally manageable first-principles-based process models by developing a new data-based modelling approach. In this approach, local linear models (identified via latent variable regression techniques) are combined with weights (arising from fuzzy c-means clustering) to describe global nonlinear process dynamics. Nonlinearities are captured through the appropriate combination of the different models while the linearity of the individual models prevents against a computationally expensive predictive controller. This modelling approach is also generalized to account for time-varying dynamics by incorporating online learning ability into the model, making it adaptive. This is accomplished by developing a probabilistic recursive least squares (PRLS) algorithm for updating a subset of the model parameters.

The data-based modelling approach is first used to generate data-based reverse-time reachability regions (RTRRs), which are subsequently incorporated in a new predictive controller. Next, the modelling approach is applied on a complex nylon-6,6 batch poly-

merization process in order to design a trajectory tracking predictive controller for the key process outputs. Through simulations, the modelling approach is shown to capture the major process nonlinearities and closed-loop results demonstrate the advantages of the proposed controller over existing options. Through further simulation studies, model adaptation (via the PRLS algorithm) is shown to be crucial for achieving acceptable control performance when encountering large disturbances in the initial conditions.

Finally, we consider the problem of *direct* quality control even when there are limited quality-related measurements available from the process; this situation typically calls for *indirectly* pursuing the control objective through trajectory tracking control. To address the problem of unavailability of online quality measurements, an inferential quality model, which relates the process conditions over the entire batch duration to the final quality, is required. The accuracy of this type of quality model, however, is sensitive to the prediction of the future batch behaviour until batch termination. This “missing data" problem is handled by integrating the previously developed data-based modelling approach with the inferential model in a predictive control framework. The key feature of this approach is that the causality and nonlinear relationships between the future inputs and outputs are accounted for in predicting the final quality and computing the manipulated input trajectory. The efficacy of the proposed predictive control design is illustrated via simulations of the nylon-6,6 batch polymerization process with a different control objective than considered previously.

Acknowledgements

I would first like to express my appreciation and thanks to Dr. Prashant Mhaskar for his guidance and support. He was the catalyst for both my interest in the control field and my pursuit of a graduate degree, and I consider myself very fortunate to have been under his direction. I have thoroughly enjoyed collaborating with him the last 4 years, and our time together has been an invaluable learning experience. Any successes I have had during my graduate studies are directly attributable to Dr. Mhaskar's patience, insight, and inspiration, and his influence will undoubtedly play a role in any future successes I may have.

I would also like to thank my mother (Shamim Ara Begum) and father (Kazi Salam) for their unwavering support in anything I have pursued. I am grateful for the personal sacrifices they have made to provide me an environment where I could succeed. This work is as much a product of their efforts as mine.

I am grateful to Dr. Christopher Swartz and Dr. Saeid Habibi for their service on my doctoral committee. Their questions, comments, and suggestions during our meetings have enhanced the quality of this research. I would also like to express my gratitude to Lynn Falkiner, Kathy Goodram, and Nanci Cole, for their handling of all things administrative and always answering all my questions, as redundant and unnecessary they may have been.

Finally, I would like to acknowledge the students of the McMaster Advanced Control Consortium (MACC). I was lucky to interact with a number of people in the MACC, and I acknowledge all those who have offered their encouragement, assistance, and friendship. The following students deserve special recognition. To my friends, Rahul Gandhi, Shailesh Patel, Zhiwen Chong, and Richard Mastragostino, I am grateful for our invaluable discussions from which I always learned something new or which sparked my curiosity in a new area of control. This was a great group to be a part of both academically and personally.

Siam Aumi

McMaster University
November 2011

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

Lower Case Symbols

$\boldsymbol{b}_j$	Inner relationship coefficients in PLS regression
$\boldsymbol{c}_\ell$	Model/ℓ-th cluster centre point in the data-based modelling approach/fuzzy c-means clustering
$\hat{\boldsymbol{c}}_z$	Ellipsoid centre point of the empirical RTRR estimate at sampling instance z
$\tilde{\boldsymbol{c}}_z$	Ellipsoid centre point of the robust RTRR estimate at sampling instance z
$\boldsymbol{d}$	Translation vector in $T(\cdot)$
$\boldsymbol{e}_j$	Inner relationship residuals in PLS regression
f	Weighting exponent parameter in fuzzy c-means clustering
$\boldsymbol{f}(\cdot)$	Vector function of the differential equations for the state variables
$\boldsymbol{g}(\cdot)$	Vector measurement function for the output variables
$\boldsymbol{h}(\cdot)$	Vector measurement function for the quality variables
$\boldsymbol{k}[k]$	RLS gain vector at sampling instance k for updating all models in the data-based modelling approach
$\boldsymbol{k}_\ell[k]$	PRLS gain vector at sampling instance k for updating model ℓ in the data-based modelling approach
ℓ	Model index in the data-based modelling approach
n_u	Lags in the inputs in an ARX model
n_y	Lags in the outputs in an ARX model
$\boldsymbol{p}_j$	Loadings of the regressor matrix for the j-th principal component
$\boldsymbol{q}$	Quality variables
$\boldsymbol{q}_j$	Loadings of the response matrix for the j-th principal component in PCA or PLS regression
$\boldsymbol{r}_j$	Scores of the response matrix on the j-th principal component in PCA or PLS regression
t_0	Batch start/initial time
t_{end}	Batch termination time
t_{fault}	Fault occurrence time
t_{repair}	Fault rectification/repair time

$\boldsymbol{t}_j$ Scores of the regressor matrix on the j-th principal component in PCA or PLS regression

$u_{i,\max,\text{fail}}$ Maximum allowable u_i during failure period

$u_{i,\min,\text{fail}}$ Minimum allowable u_i during failure period

$\boldsymbol{u}$ Process input variables

$\boldsymbol{u}_{\max}$ Maximum allowable process input variables

$\boldsymbol{u}_{\min}$ Minimum allowable process input variables

$v[k]$ Residual at sampling instance k in the data-based modelling approach

$v_\ell[k]$ Residual at sampling instance k for model ℓ in the data-based modelling approach

$\boldsymbol{v}$ Measurement noise

$\boldsymbol{w}$ Model uncertainties

$\boldsymbol{w}_{\max}$ Maximum allowable model uncertainties

$\boldsymbol{w}_{\min}$ Minimum allowable model uncertainties

$\boldsymbol{x}$ Process state variables

$\boldsymbol{x}_{\text{des}}$ Desired end-point

$\boldsymbol{x}_{\text{f}}[k]$ Row vector of future data with respect to sampling instance k in LV-MPC modelling

$\boldsymbol{x}_{n,\text{ub}}$ Point on an unit ball

$\boldsymbol{x}_{\text{p}}[k]$ Row vector of past data with respect to sampling instance k in LV-MPC modelling

$\tilde{\boldsymbol{x}}[k]$ Concatenated lagged outputs and inputs at sampling instance k

$\boldsymbol{y}$ Process output (measurable) variables

$\boldsymbol{y}_{\text{p}}$ Response vector corresponding to one of the outputs sorted sample-wise

$\boldsymbol{y}_{\text{ref}}[k]$ Reference/set-point trajectories for the output variables at sampling instance k

z $:= (t_{\text{end}} - t)/\delta$ Sampling instance index defined starting from batch termination

Upper Case Symbols

$\mathbf{A}_{\text{FCM}}$ Norm inducing matrix in fuzzy c-means clustering

$D_{\ell,i}$ Euclidean distance between point i and the ℓ-th cluster centre in fuzzy c-means clustering

$\mathbf{E}_{(\cdot)}$ Residuals in PLS regression

$\mathbf{H}$ Positive-definite, symmetric rotation matrix in $T(\cdot)$

$\mathbf{I}$ Identity matrix

$J_{(\cdot)}$ Objective function in various optimization problems

L Number of clusters/local linear models in the data-based modelling approach

L_E Lagrangian term in an end-point-based MPC design

$\mathbf{L}$ Extended Luenberger observer gain matrix

M Lags in LV-MPC

M_E Mayer term in an end-point-based MPC design

N_{disc} Number of combinations of the inputs following their discretization for RTRR generation

N_t Effective memory length in RLS estimation

N_{obs} Number of lagged output-input vectors (with n_y and n_u lags) or state-input vectors (with 1 lag in the states and inputs) that can be constructed from a given database

N_{ub} Number of predetermined points on the surface of an unit ball used for empirical RTRR generation

P Prediction horizon in a MPC design/leads in LV-MPC

$\mathbf{P}$ Loadings of the regressor matrix in PCA or PLS regression

$\mathbf{P}[k]$ Co-variance matrix of the ARX models' coefficients at sampling instance k in the data-based modelling approach

$\mathbf{P}_{\text{c}}$ Partition of the loading matrix corresponding to the inputs in LV-MPC modelling

$\mathbf{P}_{\ell}[k]$ Co-variance matrix of the ℓ-th ARX model's coefficients at sampling instance k in the data-based modelling approach

$\mathbf{P}_{\text{f}}$ Partition of the loading matrix corresponding to the future data (with respect to sampling instance k) in LV-MPC modelling

$\mathbf{P}_{\text{p}}$ Partition of the loading matrix corresponding to the past data (with respect to sampling instance k) in LV-MPC modelling

$\hat{\mathbf{P}}_z$ Positive-definite, symmetric ellipsoid matrix for the empirical RTRR estimate at sampling instance z

$\tilde{\mathbf{P}}_z$ Positive-definite, symmetric ellipsoid matrix for the robust RTRR estimate at sampling instance z

$\mathbf{Q}$ Loadings of the response matrix in PLS regression

$\mathbf{Q}_{\text{end}}$ Qualities at batch termination in a batch database

$\mathbf{R}$ Scores/projections of the response matrix in PLS regression

$S(0,1)$ Unit ball in $\mathbb{R}^n$

$T(\cdot)$ Affine transformation in $\mathbb{R}^n$

$\mathbf{T}$ Scores/projections of the regressor matrix in PCA or PLS regression

$\mathbf{U}$ Membership matrix in fuzzy c-means clustering

$\mathbf{U}_{\text{disc}}$ Discretized inputs for RTRR generation

$V(\cdot)$ Time-varying least squares criterion

$\mathbf{V}^{-1}$ Positive-definite, symmetric ellipsoid matrix after an affine transformation of $S(0,1)$

$\mathbf{X}_b$ Data for batch b in LV-MPC modelling

$\mathbf{X}_{\text{p}}$ Regressor matrix with the columns corresponding to $\begin{bmatrix}\tilde{\boldsymbol{x}} & 1\end{bmatrix}$ sorted sample-wise

$\mathbf{X}_{\text{PVT}}$ Process variable trajectories in a batch database

$\tilde{\mathbf{X}}$ Lagged output-input or state-input space to be clustered in the data-based modelling approach

$Y_i^{[k]}$ Measurement sequence of output y_i up to sampling instance k

$\mathbf{Y}_{\text{p}}$ Response matrix with the columns corresponding to the outputs sorted sample-wise

Z_0 Number of measurements related to the initial conditions in a batch database

$\mathbf{Z}_0$ Initial conditions in a batch database

Greek Symbols

$\boldsymbol{\alpha}_j$	ARX model coefficient vector for the lagged outputs
$\boldsymbol{\alpha}$	State transition matrix in an ARX model when there are full state measurements
$\boldsymbol{\beta}_j$	ARX model coefficient vector for the lagged inputs
$\boldsymbol{\beta}$	Input matrix in an ARX model when there are full state measurements
δ	Sampling period, sample time, or controller execution period
γ	ARX model bias
$\boldsymbol{\gamma}$	Bias vector in an ARX model when there are full state measurements
λ	Forgetting factor
$\boldsymbol{\Lambda}$	Inferential quality model
$\mu_{i,\ell}$	Membership of the i-th observation to the ℓ-th cluster in fuzzy c-means clustering
$\boldsymbol{\theta}$	ARX model coefficients
$\boldsymbol{\theta}_\ell[k]$	ARX model coefficients of ℓ-th local linear model at sampling instance k in the data-based modelling approach
$\boldsymbol{\Theta}$	All local linear models in the data-based modelling approach
$\omega_\ell[k]$	Model ℓ's weight at sampling instance k in the data-based modelling approach
$\boldsymbol{\Omega}$	Vertically stacked $\mathbf{X}_b$ matrices in LV-MPC modelling
$\boldsymbol{\Psi}$	Regressor matrix in the data-based modelling approach
σ_y^2	Measurement variance of output y_i
$\boldsymbol{\Pi}$	Positive-definite, diagonal weighting matrix in MPC designs
$\boldsymbol{\psi}[k]$	Known vector/matrix at sampling instance k in the data-based modelling approach
$\boldsymbol{\Xi}$	Positive-definite, diagonal weighting matrix in MPC designs
$\zeta_\ell[k]$	Posterior probability that a plant measurement originated from plant dynamics representable by model ℓ in the data-based modelling approach

Script Symbols

$\mathcal{B}(\boldsymbol{x}_{\text{des}})$	Desired end-point neighbourhood centred around $\boldsymbol{x}_{\text{des}}$
$\mathcal{N}\left(x;\, \mu, \sigma^2\right)$	Value of x on a normal distribution with mean μ and variance σ^2
$\mathcal{R}(t)$	RTRR at time t
$\mathcal{R}_z$	Discrete RTRR at sampling instance z
$\tilde{\mathcal{R}}_z$	Discrete robust RTRR at sampling instance z
$\mathcal{U}$	$:= \{\boldsymbol{u} \mid \boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}\}$ Admissible/allowable inputs
$\mathcal{W}$	$:= \{\boldsymbol{w} \mid \boldsymbol{w}_{\min} \leq \boldsymbol{w} \leq \boldsymbol{w}_{\max}\}$ Allowable model uncertainties
$\boldsymbol{\mathcal{X}}_z$	Matrix representing a point set estimate of $\mathcal{R}_z$ (each column contains a point)

Other

$\text{diag}\{\cdot\}$	Diagonal matrix where $\{\cdot\}$ is an ordered list of the diagonal elements
$\mathbf{E}\{\cdot\}$	Expected value operator
$\Pr\{\cdot\}$	Probability of an event
$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	Euclidean space of dimension n

$\mathbb{R}^{n\times m}$ Matrix of dimensions $n \times m$

$\|\boldsymbol{x}\|$ $:= \sqrt{\boldsymbol{x}'\boldsymbol{x}}$ Euclidean or l_2 norm of a vector $\boldsymbol{x}$

$\|\boldsymbol{x}\|_{\Xi}$ $:= \sqrt{\boldsymbol{x}'\Xi\boldsymbol{x}}$ Weighted Euclidean norm of a vector $\boldsymbol{x}$ with respect to a positive-definite matrix Ξ

Abbreviations

ARX Auto-regressive exogenous

ELO Extended Luenberger observer

FTCS Fault tolerant control structures

GAMS General algebraic modelling system

ILC Iterative learning control

IPOPT Interior point optimizer

ITAE Integral of time-weighted absolute error

LV-MPC Latent variable model predictive control

MPC Model predictive control

MSV Mean sum of the variances

NIPALS Nonlinear iterative partial least squares

NLP Nonlinear program

ODE Ordinary differential equation

OLS Ordinary least squares

PCA Principal component analysis

PCR Principal component regression

pdf Probability density function

PID Proportional-integral-derivative

PLS Partial least squares

PRBS Pseudo-random binary signal

PRLS Probabilistic recursive least squares

PWA Piece-wise affine

RLS Recursive least squares

RMSE Root mean squared error

RTRR Reverse-time reachability region

SPC Statistical process control

SPE Squared prediction error

TSK Takagi, Sugeno, and Kang

CHAPTER 1

Introduction

1.1 BATCH PROCESSES

Batch (and semi-batch[1]) processes constitute a class of chemical processes that play an important role in the production and processing of a wide range of value-added products. Specialized sectors of the chemical industry operate exclusively in the batch modes. Additionally, batch processes may serve as start-up/intermediate steps in continuous processing units. For these cases, conditions at batch termination can ultimately dictate process performance upon the transition to continuous mode of operation (e.g. see [1]).

A typical batch process (see Figure 1.1) consists of the following steps:

1. Charging the reactor with a recipe of raw materials whose properties are usually recorded.
2. Processing under controlled conditions for a finite duration of time (the batch termination time) during which the process inputs are varied according to a specified control policy and measurements are collected.
3. Discharging the final product and performing a range of quality measurements (which are recorded) on a product sample to determine if the final product meets required specifications.

The final product quality is dependent on the initial conditions (i.e., raw material properties), the process variable trajectories (and their cumulative effects) over the batch duration,

[1] A semi-batch process is a special class of batch process in which material may be fed (i.e., a fed-batch process) or removed during the process. The terms, batch and semi-batch, will be used interchangeably for the remainder of this thesis.

and the ability of the control policy to reject disturbances. The appeal of batch processes (over continuous processes) is the flexibility to achieve a wide range of end-point/terminal conditions, which do not have to be equilibrium conditions, by changing the initial conditions and process variable trajectories. This flexibility is particularly important in the specialty product industries. Key examples include the production of certain bio-chemicals (e.g., ethanol) and polymers (e.g., nylon-6,6) as well as numerous pharmaceutical products.

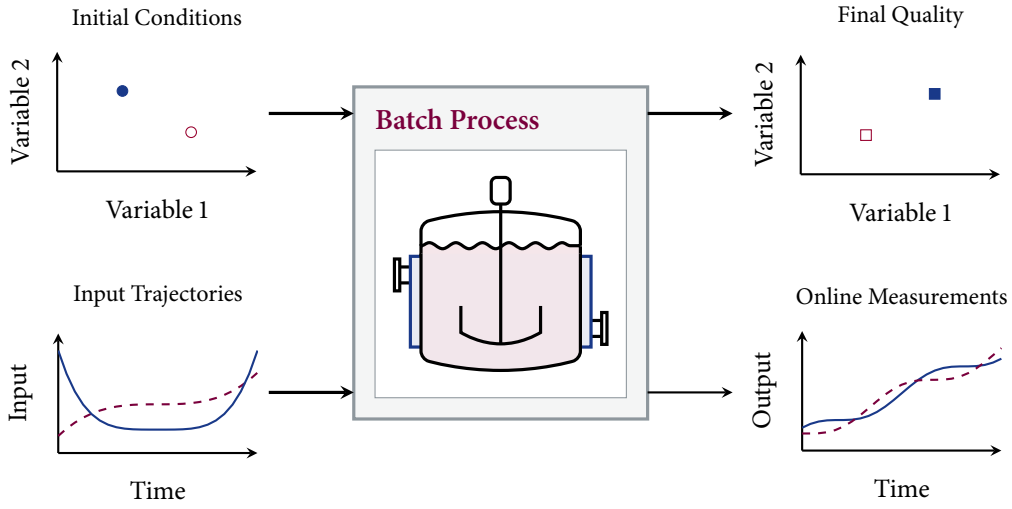


Figure 1.1: Schematic of a batch process

Mathematically, the class of batch processes considered in this work can be described by the general model form shown below.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \\ \boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) + \boldsymbol{v} \\ t &\in [t_0, t_{\text{end}}] \end{aligned} \tag{1.1}$$

where $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ is a vector of the physical states of the process and $\boldsymbol{y} \in \mathbb{R}^{p\times 1}$ denotes a vector of noise corrupted output (measurable) variables with $\boldsymbol{v}$ representing the measurement noise. The vector, $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$, denotes the constrained inputs to the process, taking values in a non-empty convex subset, $\mathcal{U}$, of $\mathbb{R}^m$ where $\mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^{m\times 1} \mid \boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}\}$ with $\boldsymbol{u}_{\min} \in \mathbb{R}^{m\times 1}$ and $\boldsymbol{u}_{\max} \in \mathbb{R}^{m\times 1}$ denoting the minimum and maximum allowable $\boldsymbol{u}$ (respectively). The vector, $\boldsymbol{w} \in \mathcal{W} \subset \mathbb{R}^q$, collects any bounded, possibly time-varying model uncertainties where $\mathcal{W} = \{\boldsymbol{w} \in \mathbb{R}^{q\times 1} \mid \boldsymbol{w}_{\min} \leq \boldsymbol{w} \leq \boldsymbol{w}_{\max}\}$ with $\boldsymbol{w}_{\min} \in \mathbb{R}^{q\times 1}$ and $\boldsymbol{w}_{\max} \in \mathbb{R}^{q\times 1}$ denoting the minimum and maximum realizations of $\boldsymbol{w}$. The times, t_0 and t_{end}, denote the initial time and batch termination times, respectively. The vector function, $\boldsymbol{f}(\cdot) : \mathbb{R}^n \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}^n$, contains ordinary differential equations (ODEs) for the state

variables (row-wise) and $\boldsymbol{g}(\cdot) : \mathbb{R}^n \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}^p$ is the measurement function for the outputs.

1.2 BATCH PROCESS CONTROL

The primary control objective in batch processes is to reach a specified product quality, which typically corresponds to a non-equilibrium point, by batch termination. In the past, batch-to-batch operation entailed implementing predetermined input trajectories that were either optimized offline, determined through data-mining, or historically yielded on-spec product[2]. Consistent results were achieved through precise sequencing and automation of all the stages in the batch operation. This type of *open-loop* operation policy, however, made the final product quality susceptible to disturbances encountered during the process and/or in the initial conditions (i.e., from raw material impurities). Motivated by the increased demands of consistently producing high quality products, numerous batch-to-batch and within-batch control strategies have been adopted.

The idea behind batch-to-batch control is to improve the batch recipe and operating trajectories for the upcoming batch using data collected from previously completed batches in an attempt to bring the new batch's quality closer to the specified value. This approach, however, represents an entirely offline strategy and lacks any real-time feedback mechanism for rejecting disturbances encountered *during* batch evolution. This motivates the use of real-time, **within-batch** control approaches, which is the focus of this research.

The within-batch control problem is complicated by many of the distinguishing features of batch processes. These include a finite duration of operation and the absence of equilibrium conditions coupled with strong nonlinear and time-varying dynamics over a wide range of operating conditions. In contrast, continuous processes are characterized by operation around a steady-state with a relatively narrower range of operating conditions. The literature on within-batch control strategies is extensive at this point[3]. The selection of an appropriate strategy is largely dictated by the availability and properties of a process model and the observability of the process. For instance, in many industrial batch processes, real-time measurements of the quality variables are unavailable or the quality variables are not observable from the available process measurements. Under these circumstances, a within-batch control strategy must rely on inferential techniques to estimate the quality for direct quality control or pursue the control objective indirectly in some fashion.

[2] In any case, these *open-loop* input trajectories incorporated the desired end-point properties in some fashion but did so offline.

[3] Many techniques will be reviewed throughout the course of this thesis.

1.2.1 Model Predictive Control

One control method that has been the foundation for many within-batch control strategies (including those proposed in this work) is model predictive control (MPC) [2]. MPC is a centralized control approach where the process dynamics and interactions can be accounted for when computing the control action through the use of a process model.

In MPC (see Figure 1.2), the sequence of steps shown below is executed at each sampling instance once a new measurement becomes available. It is repeated at each sampling instance in order to account for any information obtained from the newly available measurement.

1. The process model is initialized at the new plant measurement. This represents a feedback mechanism to account for plant-model mismatch. When using a state-space process model, this step calls for an appropriately designed state estimator.

2. An optimization problem is solved in which:

   a) The process model is used to predict the future outputs over a prediction horizon (denoted by P in Figure 1.2) for candidate input trajectories.

   b) An input trajectory is computed that minimizes an objective function while satisfying any constraints. The most common examples of constraints are input constraints that arise from the physical limitations of control actuators. An example objective function is a quadratic function of the predicted process outputs' deviations from their corresponding set-points, summed over the prediction horizon.

3. The first element of the computed input trajectory is implemented on the process.

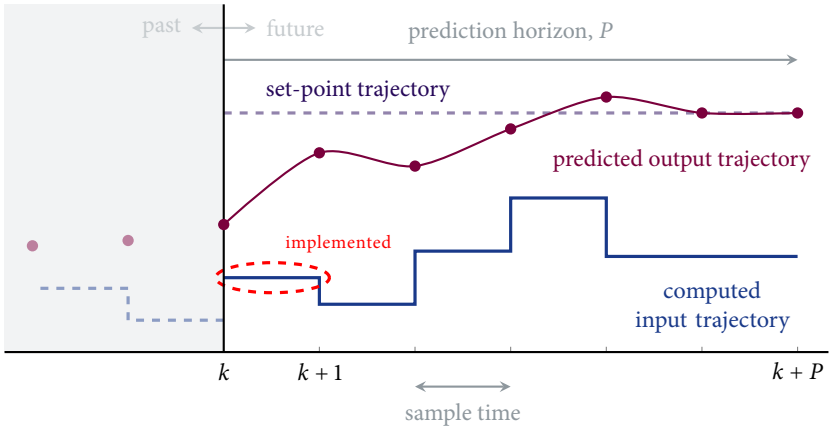


Figure 1.2: Illustration of the basic idea behind MPC. The schematic shows some of the key trajectories at the end of the MPC calculation.

The underlying model in a MPC formulation may be the first-principles, usually nonlinear model[4], a linearized version of this model, or a linear, data-based model. For many batch processes, the rigorous, first-principles model is unavailable or unreliable as it is difficult to fully characterize all the chemistry, mixing, and heat transfer phenomena occurring in the process. Additionally, even when a first-principles model is available, many of the simplifying assumptions made during its derivation can break down in practice, it may become too difficult to maintain, or the resulting MPC formulation may be too computationally expensive for real-time application.

In the literature and in practice, linear models[5] (first-principles-based or data-based) have been more popular for MPC formulations, partly because the resulting MPC optimization problem is computationally tractable for real-time application and the dynamics of *continuous* processes can be reasonably approximated by a linear model for control purposes [3, 4]. The latter is true because continuous processes tend to operate in a relatively narrow range of conditions around a steady-state operating point. A similar approximation of linear dynamics, however, applied to batch processes will result in poor control performance due to the presence of strong nonlinearities and time-varying dynamics over a much wider range of operating conditions compared to continuous processes. A key contribution of this research is the development of a data-based modelling technique that can capture the nonlinear, time-varying nature of batch dynamics while remaining amenable for real-time MPC applications.

One topic in the MPC literature that has been largely overlooked has been the design of fault tolerant control designs specific to batch processes. In general, 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 and negatively impact the overall process productivity and economy. Batch process productivity is particularly susceptible to faults as there is an emphasis on final product quality, and a fault during a batch may can ruin the entire batch product or invalidate the desirable properties of a control design. While there has been significant work on fault detection and isolation for batch processes (see, e.g., [5–9]), fault tolerant control structures (FTCS) specific to batch processes have received limited research attention. The majority of the extensive research on FTCS for continuous processes cannot be applied to batch processes due to the absence of equilibrium points and fundamental differences in the control objectives between batch and continuous processes. One result from this research is the design of a predictive controller with embedded fault-handling properties specifically from a batch process perspective.

[4] In the literature, this is also sometimes referred to as the deterministic, mechanistic, or fundamental process model.

[5] The linear model may take the form of a state-space or transfer function model with the state-space representation being more convenient for multiple-input-multiple-output systems.

1.3 THESIS OUTLINE

Motivated by the discussion above, in this thesis, we are considering the problem of designing computationally efficient, fault-tolerant predictive controllers for batch processes that are designed to achieve a desired final product quality by batch termination. The rest of this thesis is organized as follows:

Chapter 2: The novel concept of reverse-time reachability regions (RTRRs) is introduced. Defined as the set of states from where the process can be driven to a desired end-point by batch termination subject to input constraints and model uncertainties, an algorithm to mathematically characterize RTRRs *offline* at every sampling instance is given. These characterizations are subsequently used to formulate a computationally efficient, nonlinear MPC design with good fault-handling properties. The effectiveness of the predictive controller is demonstrated in both a faulty and fault-free environment via simulations of a fed-batch reactor process.

Chapter 3: A data-based multi-model approach is developed for modelling batch processes in which multiple local linear models are identified using latent variable regression techniques and combined using an appropriate weighting function that arises from fuzzy c-means clustering. The resulting model is integrated with the previously developed RTRR framework to relax the requirement of a first-principles process model. Specifically, the model is used to generate and characterize data-based or empirical RTRRs that are subsequently incorporated in a MPC design. Simulation results (with and without faults) of a fed-batch reactor process under the proposed RTRR-based design are presented. The data-based modelling methodology is then applied on an industrially relevant nylon-6,6 batch polymerization process in order to design a predictive controller that tracks time-varying set-points of the key measurable process variables.

Chapter 4: The data-based modelling methodology developed in Chapter 3 is generalized to account for time-varying dynamics by incorporating online learning ability into the model, making it adaptive. First, the standard recursive least squares algorithm with a forgetting factor is applied to update the model parameters. To address the drawbacks with this algorithm, namely that it may lead to an unnecessary update of all local linear models, a **probabilistic** recursive least squares estimator (also with a forgetting factor) is developed. The adaptation algorithms are compared by implementing them on the models developed for the nylon-6,6 batch polymerization process in Chapter 3. The adaptive models are then used in the trajectory tracking MPC design in Chapter 3 to demonstrate the benefits of model adaptation.

Chapter 5: Up to this point, the data-based modelling methodology has been integrated with the RTRR framework and used for trajectory tracking control. The former requires extensive process measurements while the latter represents an *indirect* way to achieve quality control that is sensitive to process disturbances. In this chapter, the problem of *direct* quality control with limited process measurements is addressed. To address the problem of unavailability of online quality measurements, an inferential quality model, which relates the process conditions over the entire batch duration to the final quality, is first developed. The accuracy of this type of quality model, however, is sensitive to the prediction of the future batch behaviour until batch termination, which is unknown at a given sampling instance. This "missing data" problem is handled by integrating the previously developed data-based modelling methodology in Chapter 3 with the inferential model in a MPC framework. The efficacy of the proposed predictive control design is illustrated via closed-loop simulations of the nylon-6,6 batch polymerization process with a different control objective than considered previously.

Chapter 6: The contributions of the research are summarized and suggestions for related future work are presented.

REFERENCES

[1] P. Mhaskar and S. Aumi, "Transition from batch to continuous operation in bioreactors: a model predictive control approach and application," *Can. J. Chem. Eng.*, vol. 45, pp. 416–423, 2007.

[2] D. Q. Mayne, "Non-linear model predictive control: challenges and opportunities," in *Non-Linear Model Predictive Control.* F. Allgöwer and A Zheng, Eds., Birkhäuser, Basel, 2000, pp. 23–44.

[3] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[4] J. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Syst. Mag.*, vol. 20, no. 3, pp. 38–52, 2000.

[5] P. Nomikos and J. F. MacGregor, "Monitoring batch processes using multiway principal component analysis," *AIChE J.*, vol. 40, no. 8, pp. 1361–1375, 1994.

[6] A. Cinar, S. J. Parulekar, C. Undey, and G. Birol, *Batch Fermentation: Modeling, Monitoring, and Control*. New York, NY: CRC Press, 2003.

[7] C. Undey, E. Tatara, B. Williams, G. Birol, and A. Cinar, "A hybrid supervisory knowledge-based system for monitoring penicillin fermentation," in *Proc. of American Control Conf.*, vol. 6, Chicago, IL, 2000, pp. 3944–3948.

[8] —, "On-line real-time monitoring of penicillin fermentation.," in *International Symposium on Advanced Control of Chemical Processes*, vol. 6, Pisa, Italy, 2000, pp. 243–248.

[9] C. Undey and A. Cinar, "Statistical monitoring of multistage, multiphase batch processes," *IEEE Control Syst. Mag.*, vol. 22, no. 5, pp. 1361–1375, 2002.

CHAPTER 2

Reverse-time Reachability Region-based Model Predictive Control

The results in this chapter have been published in:

JOURNAL PAPERS:

[1] S. Aumi and P. Mhaskar, "Safe-steering of batch processes," *AIChE J.*, vol. 55, pp. 2861–2872, 2009.

[2] —, "Robust model predictive control and fault-handling of batch processes," *AIChE J.*, vol. 57, pp. 1796–1808, 2010.

REFEREED CONFERENCE PROCEEDINGS:

[3] S. Aumi and P. Mhaskar, "Robust model predictive control and fault-handling of batch processes," in *Proc. of the American Control Conf.*, Baltimore, MD, 2010, pp. 4415–4420.

2.1 INTRODUCTION

The primary control objective in batch processes is to reach a desired product quality by batch termination. Naturally, this calls for control designs that specifically account for the desired end-point product properties in computing the control action such as end-point based MPC. In end-point-based MPC, the MPC optimization problem directly incorporates the desired end-point in the objective function and/or constraints. Despite the significant reduction of computational times of real-time optimization algorithms and increased availability of computational resources, the main impedance to the application of end-point-based MPC designs is the computational demand associated with repeatedly solving the MPC optimization problem. Inherent to the MPC formulation, 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.

A general way to reduce the computational costs of a MPC design is to reduce the complexity of the underlying model. By employing a linear model instead of the first-principles, nonlinear model, the MPC optimization problem can be made convex, provided the constraints and objective function remain convex[1], and there are several solvers available for efficiently solving convex optimization problems such that they can be solved in real-time (e.g. `cvx` [4]). However, due to the strong nonlinearities present in most batch processes, MPC performance with linear models is severely limited. Successive linearization techniques and scheduling of multiple linear models represent some of the workarounds to overcome these performance limitations (see [5] for a review). Recently, an input parameterization strategy, designed specifically with batch processes in mind, has been proposed for reducing the computational cost [6, 7]. In this approach, the batch control objective is first cast as an optimization problem that is solved offline (using a first-principles model), yielding a nominal set of optimal input trajectories. These trajectories are then systematically characterized and only the required adjustments to the nominal inputs for maintaining optimality are computed online (as opposed to the entire trajectory). The limiting assumption in this framework is that the active set of the solution that is determined offline does not change online. However, in practice, with modelling errors and process noise, the true plant optimum (and therefore the active set) can differ considerably from that determined offline.

In addition to the computational cost, another issue with MPC is the uncertainty in the underlying predictive model. Model uncertainties can result in significant discrepancies between the predicted and actual behaviour of the process. For instance, a predictive model integrated forward in time with a nominal realization of the uncertainties 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

[1] If the objective function is quadratic and the constraints are linear (with respect to the decision variables, the inputs), the MPC optimization problem becomes a quadratic program.

properties and/or safety constraints due to inaccurate nominal parameter values. Therefore, incorporating model uncertainties into the control calculation either by modifying the conventional MPC optimization problem (e.g., see [5, 8, 9]) or by reducing biases in state estimates arising from model uncertainties is essential (e.g., see [10–13]) for obtaining acceptable performance. While several end-point-based MPC formulations that explicitly account for model uncertainties using a min-max optimization framework are available (e.g., see [5, 9]), these approaches are often more computationally prohibitive. This is because the control moves are computed by taking into account the worst-case realization of the uncertainties which are computed using a second, embedded optimization problem within the original MPC optimization problem.

The variability in the raw material availability adds another layer of complexity to the batch control problem 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. While there exist MPC designs for *continuous* processes that allow the explicit characterization of the set of initial conditions from where stability is achievable [14–16], these results are not applicable for batch systems because the desired end-point is 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.

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; see e.g. [17]) and reconfiguration-based fault-tolerant control approaches (also called active FTC, see e.g., [18–24]). More recently, the control of nonlinear, 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 in [25]. The safe-parking FTC 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 extensive results for handling faults for continuous processes (including the safe-parking framework in [25]), 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 prevent the direct applicability of much of the research results for continuous processes. For batch processes, the majority of the FTCS are passive, essentially relying on the robustness of the control design to handle faults as disturbances during the failure period (e.g, see [9, 26]). 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 can lead to a missed opportunity to implement control action that could enable achieving the primary control objective after fault repair.

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 fails (i.e., its actuator is “stuck" at its fail-safe value), it is likely that the conventional end-point-based MPC optimization problem becomes infeasible during the faulty period because the desired end-point properties can no longer be reached with the limited available input for the rest of the batch duration. On the other hand, if the fault is repaired sufficiently fast, it may still be possible to reach the desired end-point. However, without the knowledge of the fault repair time, traditional end-point-based MPC approaches (during fault rectification) would dictate computing the input trajectories using the reduced control effort until batch termination (therefore yielding an infeasible solution). By repeatedly applying saturated versions of infeasible input trajectories, the process can be driven to a point from 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. A desirable property in a framework for handling faults in the context of batch systems, therefore, would be one that can identify input trajectories (if they exist) without requiring any prior knowledge of the fault repair time to ensure end-point reachability upon fault repair.

Motivated by these considerations, in this chapter, we consider the problem of designing a computationally efficient, nonlinear MPC design for batch processes subject to input constraints, faults in the control actuators, and model uncertainties. 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

chapter is organized as follows: First, the class of processes considered is presented followed by a review of a conventional end-point-based MPC formulation. Next, we design a reverse-time reachability region-based predictive controller that requires the online computation of only the immediate control move. In doing so, we first introduce the notion of reverse-time reachability regions and propose an algorithm for mathematically characterizing them offline. These characterizations are subsequently used to formulate the MPC optimization problem. Then, after formulating the safe-steering problem, a safe-steering framework is developed that utilizes the MPC design to ensure the process states can be driven inside a desired end-point neighbourhood if the fault is repaired sufficiently fast. Closed-loop simulation results of a fed-batch process subject to actuator failure, model uncertainties, limited availability of measurements, and sensor noise are presented to illustrate the efficacy of the proposed MPC design and the details of the safe-steering framework. Finally, we summarize our results.

2.2 PRELIMINARIES

In this section, the class of batch processes considered is presented followed by a representative formulation of a nonlinear, end-point-based predictive controller.

2.2.1 Process Description

We consider batch processes that can be described by the process description in Equation (1.1). For the results in this chapter, we also assume the following:

- The vector function, $\boldsymbol{f}(\cdot) : \mathbb{R}^n \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}^n$, in Equation (1.1) is continuous on $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})$ and locally Lipschitz in $\boldsymbol{x}$ on $\mathcal{D} \times \mathcal{U} \times \mathcal{W}$, where $\mathcal{D} \subset \mathbb{R}^n$.
- For any $\boldsymbol{u} \in \mathcal{U}$ and $\boldsymbol{w} \in \mathcal{W}$, the solution of the model in Equation (1.1) exists and is continuous $\forall\, t \in [t_0, t_{\text{end}}]$.
- The desired end-point quality can be expressed as a corresponding state vector denoted by $\boldsymbol{x}_{\text{des}}$, which is specified at the process design phase.

Note that for some cases, not all the elements of $\boldsymbol{x}_{\text{des}}$ are explicitly specified at the process design phase. In these cases, the objective may be to maximize or minimize a certain performance objective (i.e., maximize product concentration). Accordingly, for these cases, a nominal optimization problem can be solved *offline* with the appropriate performance objective, and $\boldsymbol{x}_{\text{des}}$ can be taken to be the state vector at t_{end} from the optimal state trajectories.

2.2.2 *End-point-based Model Predictive Control*

In this section, a representative formulation of a shrinking horizon, nonlinear end-point-based predictive controller is presented. This formulation is not meant to generalize all variations of published MPC formulations of this type but only meant to convey the key idea in most existing formulations, which is the computation of the input trajectories from the current time to the end of the batch.

Consider the batch process described by Equation (1.1), the control action at each sampling instance is computed by solving the following dynamic optimization problem:

$$\min_{\boldsymbol{u}(\tau)\in\mathcal{U}} J_E = M_E\left(\boldsymbol{x}(t_0), \tilde{\boldsymbol{x}}(t_{\text{end}}-t), \boldsymbol{x}_{\text{des}}\right) + \int_t^{t_{\text{end}}} L_E\left(\tilde{\boldsymbol{x}}, \boldsymbol{u}\right)\,\mathrm{d}\tau \tag{2.1a}$$

$$\text{subject to: } \tilde{\boldsymbol{x}}(0) = \boldsymbol{x}(t) \tag{2.1b}$$

$$\tilde{\boldsymbol{x}}(t_{\text{end}}-t) = \tilde{\boldsymbol{x}}(0) + \int_0^{t_{\text{end}}-t} \boldsymbol{f}(\tilde{\boldsymbol{x}}, \boldsymbol{u})\,\mathrm{d}\tau \tag{2.1c}$$

$$\tilde{\boldsymbol{x}}(t_{\text{end}}-t) = \boldsymbol{x}_{\text{des}} \tag{2.1d}$$

where $M_E(\cdot)$ and $L_E(\cdot)$ represent the Mayer and Lagrangian terms, respectively. The Mayer term explicitly involves the initial and final conditions while the Lagrangian term is frequently used to implement soft constraints on the control rate or minimize deviations from some nominally optimal state and input trajectories. Equation (2.1b) represents the initialization of the optimization problem at the current process conditions/states and can be understood as the feedback mechanism to account for plant-model mismatch. In the absence of full state measurements, this calls for estimating $\boldsymbol{x}(t)$ using a suitable state estimator[2]. Equation (2.1c) represents the model integration to the end of the batch, and the terminal constraint, Equation (2.1d), specifies that the model should be driven to the desired end-point $\boldsymbol{x}_{\text{des}}$ in the remaining batch time or $t_{\text{end}} - t$. The minimizing control action is directly implemented on the process over the interval $[t, t+\delta)$, where δ is the sampling period, and this procedure is repeated until batch termination.

The evaluation of the objective function, Equation (2.1a), and terminal constraint, Equation (2.1d), necessitates the integration of the nonlinear model and optimization of the inputs up to t_{end} at each sampling instance. Thus, the optimization problem becomes computationally expensive regardless of the optimization strategy (sequential or simultaneous). Note that in the absence of model uncertainties, the solution to the optimization problem is only required at the first sampling instance because the solution at the j-th time step is simply the initial solution trajectory from $(j+1)\delta$ to t_{end} (i.e., the "tail" of the solution - see Figure 2.1).

[2] A variety of state estimators that are capable of handling nonlinearities can be used such as the extended Kalman filter, unscented Kalman Filter, or a moving horizon estimator. A review of these algorithms is beyond the scope of this thesis.

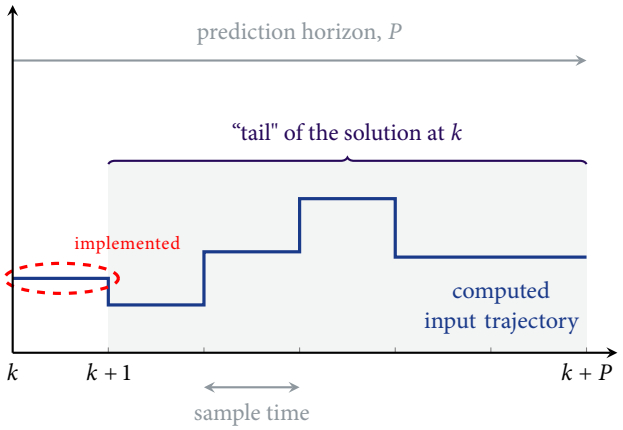


Figure 2.1: Illustration of the "tail" of a MPC optimization problem solution at sampling instance k

With the presence of model uncertainties, the "tail" of the initial solution is no longer a solution for subsequent MPC optimization problems. While 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 uncertainties.

While we present a "nominal" MPC formulation to emphasize the fact that the control calculation at every instance requires the solution of the entire input trajectories, a min-max dynamic optimization problem can, in principle, be employed to handle the problem of uncertainties. The solution to such a min-max problem would be input trajectories from the current time to the end of the batch that minimize (using the inputs as decision variables) the maximum (over all realizations of the uncertainties) value of the objective function. This added layer of optimization renders min-max based MPC approaches for batch processes even more computationally expensive than the nominal end-point-based MPC formulation in Equations (2.1a) to (2.1d) and motivates the development of a computationally efficient and robust nonlinear MPC design for batch processes.

2.3 REVERSE-TIME REACHABILITY REGION-BASED MODEL PREDICTIVE CONTROL

In this section, we present a nonlinear predictive controller for batch processes. The key idea behind this design is to require the computation of only the immediate values of the inputs while ensuring the desired end-point remains attainable throughout the batch. Preparatory to the controller design, we first introduce the notion of **reverse-time reachability regions** (RTRRs), which are essential in the control design and analysis. Initially, we assume no model uncertainties to establish the fundamentals and then define robust RTRRs that explicitly take model uncertainties into account.

2.3.1 Reverse-time Reachability Regions

As previously discussed, the objective in batch processes is to reach a desired end-point, $\boldsymbol{x}_{\text{des}}$, and of interest is the set of states from where $\boldsymbol{x}_{\text{des}}$ can be reached. This set can be expressed in the form of reverse-time reachability regions (RTRRs), which are formally defined below.

Definition 2.1 (Reverse-time Reachability Region): *For the batch process described by Equation* (1.1) *without model uncertainties, the reverse-time reachability region (RTRR) at time t, $\mathcal{R}(t)$, is the set:*

$$\mathcal{R}(t) = \{\boldsymbol{x}_0 \mid \boldsymbol{x}(t_{end}) = \boldsymbol{x}_0 + \int_t^{t_{end}} \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\,\mathrm{d}\tau = \boldsymbol{x}_{des}\ \exists\, \boldsymbol{u}(t) \in \mathcal{U}\ \forall t \in [t, t_{end}]\}$$

The RTRR at time t, $\mathcal{R}(t)$, therefore, consists of all process states from where the process can be steered to $\boldsymbol{x}_{\text{des}}$ by the end of the batch (i.e., in a time $t_{\text{end}} - t$) while satisfying the input constraints. The reason behind naming this set the “reverse-time reachability region" is as follows. Note that a reachability region 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 input constraints. If the “reverse-time" version of the process is considered (i.e., $\dot{\boldsymbol{x}}(t) = -\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$), 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 in turn 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).

While $\mathcal{R}(t)$ is defined allowing for $\boldsymbol{u}(t)$ to take values in $\mathcal{U}$, computation of the RTRRs can only be carried out by discretizing the control action (i.e., subject to a control action held constant for a predefined period of time). Below we define the discrete time version of RTRRs where the control action is held for a time δ at each sampling instance until batch termination.

Definition 2.2 (Discrete Reverse-time Reachability Region): *For the batch process described by Equation* (1.1) *with sampling period, δ, and without model uncertainties, the discrete reverse-time reachability region (RTRR) at time $t = t_{end} - z\delta$, indexed by z, is the set:*

$$\mathcal{R}_z = \{\boldsymbol{x}_0 \mid \boldsymbol{x}(t_{\text{end}}) = \boldsymbol{x}_0 + \int_t^{t_{end}} \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\,\mathrm{d}\tau = \boldsymbol{x}_{des}\ \exists\, \boldsymbol{u}(t) = \{\boldsymbol{u}[i]\} \in \mathcal{U} \ldots$$
$$\forall\, i = 1, \ldots, z\}$$

where $\boldsymbol{u}[i] = \boldsymbol{u}(i\delta)$ and satisfies $\boldsymbol{u}(t) = \boldsymbol{u}[i]\ \forall t \in [i\delta, (i+1)\delta)$.

Generating Reverse-time Reachability Regions

One way to compute $\mathcal{R}_z$ 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 the input constraints

and to include every state for which the optimization has a feasible solution. However, the understanding of these sets as being the reachability regions of the reverse-time version of the process allows their sequential determination (of an estimate) without having to solve optimization problems.

In particular, for a given $\boldsymbol{x}_{\text{des}}$, the reverse-time process model, (i.e., $\dot{\boldsymbol{x}}(t) = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$), can be integrated backwards in time for the duration of δ, holding the value of the inputs constant. Performing this integration for all possible (appropriately discretized) values of the inputs (see Figure 2.2) in turn yields an (under) estimate of $\mathcal{R}_1$ (the fact that the computation yields an underestimate, however, does not negatively impact its use within the controller design). A finer discretization in terms of the inputs naturally yields a better estimate of the RTRR. $\mathcal{R}_2$ can, in turn, be determined by repeating the process for all elements in $\mathcal{R}_1$, and the process repeated to yield the RTRR for the initial time.

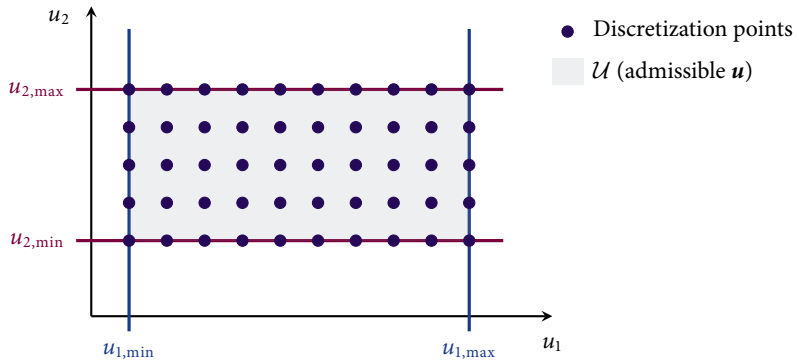


Figure 2.2: Illustration of discretized $\boldsymbol{u}$ between $\boldsymbol{u}_{\min}$ and $\boldsymbol{u}_{\max}$ for 2 inputs.

The computational demands of generating RTRRs increase when computing $\mathcal{R}_2$ compared to $\mathcal{R}_1$ (since $\mathcal{R}_2$ is the set of initial conditions from where a state in $\mathcal{R}_1$ can be reached, compared to $\mathcal{R}_1$, which is the set of initial conditions from where only a single point, $\boldsymbol{x}_{\text{des}}$, can be reached). In general, the increase in the computational demands is related to the increase in size of the RTRRs as we go back in time. However, it is worth noting that the size of these sets does not necessarily grow as fast when going back in time for processes where the desired end-point is an equilibrium point (i.e., continuous processes - see Remark 2.3).

In addition to being dependent on the size of the previously generated RTRR, the computational demand is also generally dependent on the number of process states and inputs. For processes with a high number of states, 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 integration software available in the public domain (see [27]) can be utilized. While a higher 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 inputs. The generation algorithm, however, has an important feature in that integrations of the (reverse-time) model equations may be done independently for different values of the inputs and initial conditions. Accordingly, to alleviate potential computational issues associated with having a high number of inputs, starting from a given RTRR, the integrations may be done independently, which, in turn, implies parallel computing schemes can be readily employed to significantly reduce the computation times.

Pseudo-code on the construct of RTRRs is presented in Algorithm 2.1 to clarify the algorithm described in this section. First, we establish some of the algorithm notations. Let:

- $z := (t_{\text{end}} - t)/\delta = \{0, \dots, Z\}$ index the sampling instances
- N_{disc} be the number of combinations of the inputs following their discretization
- $\mathbf{U}_{\text{disc}} \in \mathbb{R}^{m \times N_{\text{disc}}}$ be a matrix holding the discretized inputs

---

Algorithm 2.1 RTRR GENERATION WITHOUT MODEL UNCERTAINTIES

Require: $\boldsymbol{x}_{\text{des}}, \mathbf{U}_{\text{disc}}, \delta$
  $z \leftarrow 0$
  $\boldsymbol{\mathcal{X}}_z \leftarrow \boldsymbol{x}_{\text{des}}$
  for $z = 0$ **to** $z = Z$ **do**
    $n_z \leftarrow$ number of columns in $\boldsymbol{\mathcal{X}}_z$
    for $i = 1$ **to** n_z **do**
      for $j = 1$ **to** N_{disc} **do**
        $\boldsymbol{x}_z^* \leftarrow i$-th column of $\boldsymbol{\mathcal{X}}_z$
        $\boldsymbol{u}^* \leftarrow j$-th column of $\mathbf{U}_{\text{disc}}$
        $\boldsymbol{x}_{z+1}^* = \boldsymbol{x}_z^* + \int_0^\delta -\boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}^*\right) \mathrm{d}\tau$
        Store $\boldsymbol{x}_{z+1}^*$ in $\boldsymbol{\mathcal{X}}_{z+1}$ (column-wise)
      end for
    end for
  end for
  return $\boldsymbol{\mathcal{X}}_1, \boldsymbol{\mathcal{X}}_2, \cdots, \boldsymbol{\mathcal{X}}_Z$ as point set estimates of $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_Z$

Remark 2.1: The RTRR generation algorithm outlined above describes how RTRRs can be constructed via only integrations of the reverse-time model of the system, $\dot{\boldsymbol{x}}(t) = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$. With the assumptions that $-\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ is continuous on $(\boldsymbol{x}, \boldsymbol{u})$ and is locally Lipschitz in $\boldsymbol{x}$ on $\mathcal{D} \times \mathcal{U}$ (see Section 2.2.1), the continuity of the solutions of $\dot{\boldsymbol{x}}(t) = -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ in terms of the initial conditions and inputs is guaranteed. As a result, these assumptions ensure that RTRRs generated at each sampling instance 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.

Remark 2.2: Existing Lyapunov-based control designs can be very well used in the context of batch process control; however, 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 $\boldsymbol{f}(\boldsymbol{x}_{\text{des}}, 0) = 0$, is not satisfied in the case of batch processes. Note that this cannot be achieved by a coordinate transformation because $\boldsymbol{x}_{\text{des}}$ is simply not an equilibrium point of the process. Of course, a positive-definite function V_c can be defined such that $V_c(\boldsymbol{x}_{\text{des}}) = 0$. The set of states for which $\dot{V}_c$ can be made negative, however, does not form a neighbourhood around $\boldsymbol{x}_{\text{des}}$, which, in turn, precludes the construction of "invariant" sets around $\boldsymbol{x}_{\text{des}}$. 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 2.3: While seemingly conceptually similar, RTRRs in the context of continuous processes are inherently different from those in the context of batch processes. In particular, when considering stabilization to an equilibrium point, the RTRRs, 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, $\mathcal{R}(t_1) \subset \mathcal{R}(t_2)$ when $t_1 < t_2$ (see Figure 2.3).

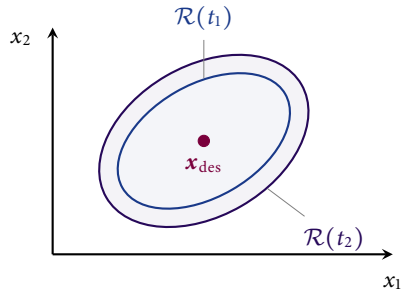


Figure 2.3: Illustration of $\mathcal{R}(t_1) \subset \mathcal{R}(t_2)$ when $t_1 < t_2$ and $\boldsymbol{x}_{\text{des}}$ is an equilibrium point

To understand this, consider the set of states which constitute $\mathcal{R}(t_2)$; there naturally exists a subset of states within $\mathcal{R}(t_2)$ that can be steered to $\boldsymbol{x}_{\text{des}}$ in a time $t_{\text{end}} - t_2$ and simply kept there until t_{end} as $\boldsymbol{x}_{\text{des}}$ is an equilibrium point. The set of points for which this time is equal to $t_{\text{end}} - t_1$ constitutes $\mathcal{R}(t_1)$. In contrast, when $\boldsymbol{x}_{\text{des}}$ is not an equilibrium point, just because a point is in $\mathcal{R}(t_1)$ does not ensure that it is in $\mathcal{R}(t_2)$ since the process cannot be "parked" at $\boldsymbol{x}_{\text{des}}$.

Remark 2.4: The presence of input constraints has significant implications on the ability to control continuous (e.g., see [28]) as well as batch systems. Despite their differences, one common property among null-controllable regions for continuous processes and RTRRs

for batch processes is that they are both dependent only on the process dynamics and input constraints and do not depend on a specific control law.

2.3.2 *Reverse-time Reachability Region-based Model Predictive Controller*

A predictive controller that utilizes explicit characterizations of RTRRs is presented in this section. To this end, consider the process described by Equation (1.1) for which the RTRRs have been characterized for a given $\boldsymbol{x}_{\text{des}}$. The control action at sampling instant $z := (t_{\text{end}} - t)/\delta$ is computed by solving the optimization problem shown below.

$$\min_{\boldsymbol{u}[z] \in \mathcal{U}} J_R \tag{2.2a}$$

$$\text{subject to: } \tilde{\boldsymbol{x}}(0) = \boldsymbol{x}(t) \tag{2.2b}$$

$$\tilde{\boldsymbol{x}}(\delta) = \tilde{\boldsymbol{x}}(0) + \int_0^{\delta} \boldsymbol{f}(\tilde{\boldsymbol{x}}, \boldsymbol{u}[z])\, \mathrm{d}\tau \tag{2.2c}$$

$$\tilde{\boldsymbol{x}}(\delta) \in \mathcal{R}_{z-1} \tag{2.2d}$$

The objective function J_R can be chosen to meet desired performance objectives. For instance, to minimize discrepancies between the state trajectories and some nominally optimal state trajectories, $\boldsymbol{x}_{\text{nom}}$, and prevent against large successive input changes, a possible J_R is:

$$J_R = \int_0^{\delta} \|\tilde{\boldsymbol{x}}(\tau) - \boldsymbol{x}_{\text{nom}}(\tau)\|_{\boldsymbol{\Xi}}^2 \, \mathrm{d}\tau + \|\boldsymbol{u}[z] - \boldsymbol{u}[z+1]\|_{\boldsymbol{\Pi}}^2 \tag{2.3}$$

where the notation, $\|\cdot\|_{\boldsymbol{\Xi}}^2$, refers to the weighted norm, defined by $\|\boldsymbol{x}\|_{\boldsymbol{\Xi}}^2 = \boldsymbol{x}'\boldsymbol{\Xi}\boldsymbol{x}$. The matrices, $\boldsymbol{\Xi}$ and $\boldsymbol{\Pi}$, are positive-definite weighting matrices to trade-off the relative importance of the 2 terms. As evidenced by Equation (2.2d), implementation of the RTRR-based controller necessitates an explicit characterization of RTRRs. By definition, RTRRs take the desired end-point into account; consequently, any existing terminal constraints in end-point-based MPC designs can be replaced with a constraint that requires, at each sampling instance, the process states to remain in the RTRR at the next sampling instance (Equation (2.2d)). The key idea behind this is to maintain the states within RTRRs for the duration of the batch as shown in Figure 2.4. Implications on the reachability guarantees to the desired end-point through this replacement are formalized in Theorem 2.1.

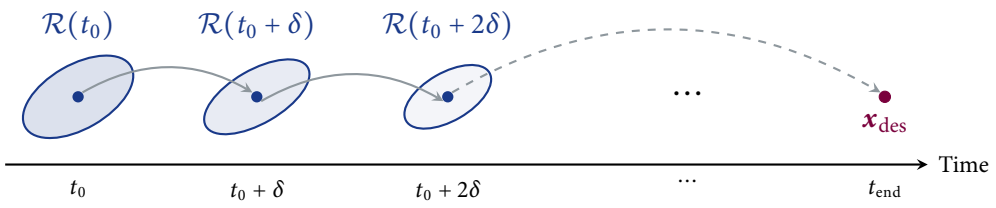


Figure 2.4: Illustration of the key idea behind the RTRR-based MPC design. The RTRR characterizations are shown as ellipsoids for illustrative purposes.

Theorem 2.1: *Consider the batch process described by Equation (1.1) without model uncertainties under the RTRR-based controller in Equations (2.2a) to (2.2d).* ***If and only if*** $\boldsymbol{x}(t_0) \in \mathcal{R}(t_0)$, *the MPC optimization problem remains feasible* $\forall\, t \in [t_0, t_{end}]$ *and* $\boldsymbol{x}(t_{end}) = \boldsymbol{x}_{des}$.

Proof: (Necessity) We first show the **only if** part of the theorem. To this end, consider an initial condition $\boldsymbol{x}(t_0) \notin \mathcal{R}(t_0)$. If the constraint of Equation (2.2d) is feasible and it is implemented in closed-loop, then there exists a sequence of inputs that takes the states to $\boldsymbol{x}_{\text{des}}$ by t_{end}, in turn implying that $\boldsymbol{x}(t_0) \in \mathcal{R}(t_0)$. This argument can be repeated at every sampling instance, eventually leading to the optimization problem in Equations (2.2a) to (2.2d) remaining feasible $\forall\, t \in [t_0, t_{\text{end}}]$ and $\boldsymbol{x}(t_{\text{end}}) = \boldsymbol{x}_{\text{des}}$ **only if** $\boldsymbol{x}(t_0) \in \mathcal{R}(t_0)$.

(Sufficiency) We now show the **if** part of the condition. To this end, consider the case when $\boldsymbol{x}(t_0) \in \mathcal{R}(t_0)$. By definition, there exists a sequence of inputs that takes the states to $\boldsymbol{x}_{\text{des}}$ by t_{end}. For such a sequence of inputs (and the associated state trajectory), this must imply that the state trajectory at $t_0 + \delta$ resides within $\mathcal{R}(t_0 + \delta)$ (invoking the necessity of the condition proved earlier for $\boldsymbol{x}(t_0 + \delta)$). In essence, this implies that there exists a feasible solution to the constraint of Equation (2.2d). This completes the proof of Theorem 2.1.

The statement of Theorem 2.1 essentially formalizes that the existence of the states in RTRRs 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 RTRRs, the desired end-point simply cannot be reached. In other words, the condition of continued existence in successive RTRRs cannot be relaxed because if the states go outside the RTRRs, it is simply not possible (whether using the proposed RTRR-based predictive controller or any other control law) to steer the states back into the RTRRs and then to the desired end-point by the batch termination time.

Remark 2.5: When using Algorithm 2.1, the true RTRRs are estimated as point sets. Depending on the specific process under investigation, the shape and orientation of the point sets may permit different strategies for their explicit characterization technique. In any case, the explicit characterization must be either an exact characterization (which is unlikely) or an *underestimate* of the true set. With a characterization that represents an overestimate, a state vector can be incorrectly identified as belonging to the true RTRR. Hence, the constraint in Equation (2.2d) can be satisfied initially even when the states are not contained in the true RTRR, invalidating the guarantees of successive feasibility of the MPC optimization problem. In contrast, if the explicit characterization is an underestimate (generated appropriately), successive feasibility of the optimization problem can be still guaranteed. This idea is illustrated in Figure 2.5.

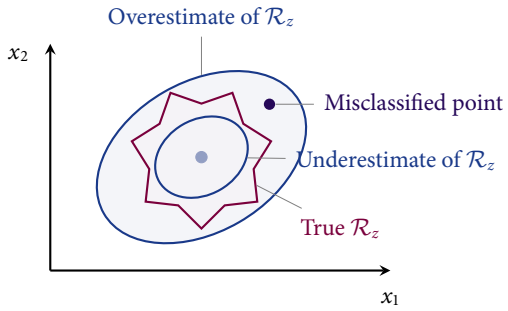


Figure 2.5: Illustration of a misclassified point as a result of an overestimated RTRR. Note that the underestimate prevents against misclassification.

Remark 2.6: With the attainment of the desired end-point achieved via Equation (2.2d), the objective function in the MPC formulation can be utilized to satisfy performance (as shown in Equation (2.3)) or even robustness objectives. Specifically, to enhance disturbance rejection and robustness, the objective function can be used to penalize the Euclidean distance between the process states and the centres of the RTRRs. This would tend to drive the states through the RTRR centres, thereby reducing the chances of disturbances driving the process to a point from where the desired end-point becomes unreachable.

2.4 ROBUST REVERSE-TIME REACHABILITY REGION-BASED MODEL PREDICTIVE CONTROL

The reverse-time reachability regions (RTRRs) in the previous section are estimated by integrating the reverse-time process model; consequently, the shapes and sizes of the estimated regions are sensitive to modelling errors. Due to the possibility of discrepancies between the estimated and true RTRRs in the presence of modelling errors, there is the potential of misclassifying states as being contained within a true RTRR. In such cases, the reachability guarantees provided by the nominal RTRR-based controller do not hold, and its direct application could very likely result in off-spec product and a wasted batch. In this section, we redesign the nominal RTRR-based predictive controller to explicitly account for model uncertainties by incorporating bounds for the uncertainties in the generation of RTRRs.

2.4.1 Robust Reverse-time Reachability Regions

In order to define robust RTRRs, we first note that in batch process control, the implication of model uncertainties is that in general, exact end-point reachability guarantees cannot be made, regardless of the control law. Instead, only reachability to a certain neighbourhood around the desired end-point can be guaranteed.

To understand this, consider a batch process described by Equation (1.1) subject to a predictive controller. At time $t = t_{\text{end}} - \delta$, the MPC optimization problem is solved to compute the inputs that drive the process to the desired end-point in time δ for *nominal* values of the uncertainty, $\boldsymbol{w}_{\text{nom}}$. However, if the same control moves are implemented on the process, there is no guarantee that the process will be driven to $\boldsymbol{x}_{\text{des}}$ because it is unknown if $\boldsymbol{w}_{\text{nom}}$ represents the actual realization of the uncertainties. This is illustrated in Figure 2.6.

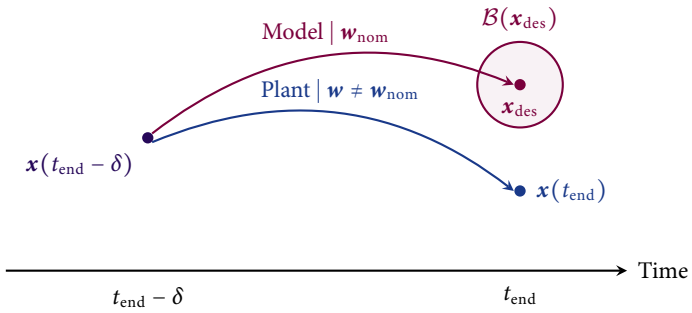


Figure 2.6: Illustration of the effects of uncertainties on batch process control. The vector $\boldsymbol{w}_{\text{nom}}$ denotes the nominal realization of the uncertainties that is assumed for the control calculation, but it may not equal the actual realization $\boldsymbol{w}$.

Based on this argument, a desirable property of a robust MPC design is to guarantee the existence of inputs to drive the states inside a *neighbourhood* around the desired end-point, which is denoted by $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. This neighbourhood can be chosen based on the acceptable level of variance in the desired end-point quality. Accordingly, of interest is the set of states from where $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ can be reached in the presence of model uncertainties while satisfying input constraints. These sets are termed **robust** RTRRs and defined below.

Definition 2.3 (Robust Reverse-time Reachability Region): *For the batch process described by Equation* (1.1) *with sampling period, δ, the robust reverse-time reachability region (RTRR) at time $t = t_{end} - z\delta$, indexed by z, is the set:*

$$\tilde{\mathcal{R}}_z = \{\boldsymbol{x}_0 \mid \boldsymbol{x}(t_{end}) = \boldsymbol{x}_0 + \int_t^{t_{end}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})\, \mathrm{d}\tau \in \mathcal{B}(\boldsymbol{x}_{des})\ \forall \boldsymbol{w}(\tau) \in \mathcal{W}\, \forall \tau \in [t, t_{end}]$$
$$\exists\, \boldsymbol{u}(t) = \{\boldsymbol{u}[i]\} \in \mathcal{U}\, \forall i = 1, \ldots, z\}$$

where $\boldsymbol{u}[i] = \boldsymbol{u}(i\delta)$ and satisfies $\boldsymbol{u}(t) = \boldsymbol{u}[i] \forall t \in [i\delta, (i+1)\delta)$.

Note that for the special case of $z = 0$, $\tilde{\mathcal{R}}_z$ is defined to be $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. Prior to presenting an algorithm to generate robust RTRRs, the existence of these regions must first be established. This is formalized in Theorem 2.2.

Theorem 2.2: *For the batch process described by Equation (1.1), given $\mathcal{W}$ and a non-empty $\tilde{\mathcal{R}}_{z-1}$, there exists a sampling period, δ^*, such that for any $\delta \leq \delta^*$, $\tilde{\mathcal{R}}_z \neq \varnothing$ (i.e., $\tilde{\mathcal{R}}_z$ is non-empty).*

Proof: Consider an element $\boldsymbol{x}_{z-1,\text{nom}}$ in the interior of $\tilde{\mathcal{R}}_{z-1}$ and the point $\boldsymbol{x}_{z,\text{nom}}$ given by[3]:

$$\boldsymbol{x}_{z,\text{nom}} = \boldsymbol{x}_{z-1,\text{nom}} + \int_t^{t+\delta} -\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\text{nom}}, \boldsymbol{w}_{\text{nom}})\, \mathrm{d}\tau$$

where δ is to be determined and $\boldsymbol{u}_{\text{nom}}$ and $\boldsymbol{w}_{\text{nom}}$ are nominal values of the inputs and uncertainties, respectively. It follows that

$$\boldsymbol{x}_{z-1,\text{nom}} = \boldsymbol{x}_{z,\text{nom}} + \int_t^{t+\delta} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\text{nom}}, \boldsymbol{w}_{\text{nom}})\, \mathrm{d}\tau$$

Define:

$$\boldsymbol{x}_{z-1} = \boldsymbol{x}_{z,\text{nom}} + \int_t^{t+\delta} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{\text{nom}}, \boldsymbol{w})\, \mathrm{d}\tau$$

i.e., $\boldsymbol{x}_{z-1}$ is the state vector at $t+\delta$, starting at t from $\boldsymbol{x}_{z,\text{nom}}$ subject to the actual realization of the uncertainties. From the continuity of $\boldsymbol{f}(\cdot)$ on $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})$ and that it is locally Lipschitz in $\boldsymbol{x}$ on $\mathcal{D} \times \mathcal{U} \times \mathcal{W}$, the continuity of solutions of $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})$ from $\boldsymbol{x}_z$ with respect to parameters (and therefore, the uncertainties) follows from [29, Theorem 3.5]. From the proof of [29, Theorem 3.5], it follows that given a desired bound on the discrepancy between the evolution of the nominal and perturbed system (i.e., $\|\boldsymbol{x}_{z-1} - \boldsymbol{x}_{z-1,\text{nom}}\| \leq \rho^*$), there exists a value δ^* such that if the sampling period, $\delta \leq \delta^*$ then it is guaranteed that $\|\boldsymbol{x}_{z-1} - \boldsymbol{x}_{z-1,\text{nom}}\| \leq \rho^*$. Therefore, $\boldsymbol{x}_{z,\text{nom}}$ is an element of $\tilde{\mathcal{R}}_z$, showing $\tilde{\mathcal{R}}_z \neq \varnothing$. This completes the proof of Theorem 2.2.

In the absence of model uncertainties, the existence of non-empty RTRRs is guaranteed simply from the existence of a solution over a finite time. When considering uncertainties, however, the size of the robust RTRRs depends on the size of the desired end-point neighbourhood and also the sampling period in the batch. For example, if we consider a fixed $\mathcal{B}(\boldsymbol{x}_{\text{des}})$, robust RTRRs may cease to exist as we proceed towards the initial time if the given δ is too large. Theorem 2.2 is therefore important in establishing the trade-off between $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ and δ. From a practical perspective, for a specified $\mathcal{B}(\boldsymbol{x}_{\text{des}})$, the result of Theorem 2.2 implies that the sampling period can be used to mitigate the reduction in the size of robust RTRRs as we proceed towards the initial time.

[3] By the interior of $\tilde{\mathcal{R}}_{z-1}$, we mean there exists a ρ^* such that $\mathcal{I}_{z-1} = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}_{z-1}\| \leq \rho^*\} \subset \tilde{\mathcal{R}}_{z-1}$.

Generating Robust Reverse-time Reachability Regions

Next, we develop a methodology to sequentially generate robust RTRR estimates offline. More specifically, starting at $z = 1$, for a given $\mathcal{B}(\boldsymbol{x}_{\text{des}})$, an explicitly characterized estimate of $\tilde{\mathcal{R}}_z$ is identified from where the process can be driven inside $\tilde{\mathcal{R}}_{z-1}$ in the presence of model uncertainties. 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 as show below[4].

$$\tilde{\mathcal{R}}_z \approx \{\boldsymbol{x} \mid \|\boldsymbol{x} - \tilde{\boldsymbol{c}}_z\|^2_{\tilde{\mathbf{P}}_z} \leq 1\} \tag{2.4}$$

where the vector $\tilde{\boldsymbol{c}}_z \in \mathbb{R}^n$ denotes the ellipsoid's centre point, the positive-definite, symmetric matrix $\tilde{\mathbf{P}}_z \in \mathbb{R}^{n\times n}$ defines its size and orientation, and z indexes the batch sampling instances as before. Note that because $z = 0$ corresponds to t_{end}, $\tilde{\boldsymbol{c}}_0 = \boldsymbol{x}_{\text{des}}$ and $\tilde{\mathbf{P}}_0$ is a user defined matrix based on the acceptable variance level of the final product quality.

To determine if an ellipsoid defined by $\left(\tilde{\boldsymbol{c}}_z, \tilde{\mathbf{P}}_z\right)$ is a valid estimate of $\tilde{\mathcal{R}}_z$, we solve, for a given δ, $\mathcal{W}$, and $\tilde{\mathcal{R}}_{z-1}$ estimate defined by $\left(\tilde{\boldsymbol{c}}_{z-1}, \tilde{\mathbf{P}}_{z-1}\right)$, the following multi-level nonlinear program (NLP):

$$\min_{\boldsymbol{x}_0} \; J_1 = 0 \tag{2.5a}$$

$$\text{subject to: } \|\boldsymbol{x}_0 - \tilde{\boldsymbol{c}}_z\|^2_{\tilde{\mathbf{P}}_z} \leq 1 \tag{2.5b}$$

$$J_2 \geq 1 \tag{2.5c}$$

$$\min_{\boldsymbol{u}\in\mathcal{U}} \; J_2 = \|\tilde{\boldsymbol{x}}(\delta) - \tilde{\boldsymbol{c}}_{z-1}\|^2_{\tilde{\mathbf{P}}_{z-1}} \tag{2.5d}$$

$$\text{subject to: } \tilde{\boldsymbol{x}}(\delta) = \boldsymbol{x}_0 + \int_0^\delta \boldsymbol{f}(\tilde{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{w})\, \mathrm{d}\tau \tag{2.5e}$$

$$\max_{\boldsymbol{w}\in\mathcal{W}} \; J_3 = \|\tilde{\tilde{\boldsymbol{x}}}(\delta) - \tilde{\boldsymbol{c}}_{z-1}\|^2_{\tilde{\mathbf{P}}_{z-1}} \tag{2.5f}$$

$$\text{subject to: } \tilde{\tilde{\boldsymbol{x}}}(\delta) = \boldsymbol{x}_0 + \int_0^\delta \boldsymbol{f}(\tilde{\tilde{\boldsymbol{x}}}, \boldsymbol{u}, \boldsymbol{w})\, \mathrm{d}\tau \tag{2.5g}$$

If this NLP is infeasible, we deem the estimate to be a valid robust RTRR. To understand this, consider the different levels of the NLP. For a given initial state within $\mathcal{R}_{z-1}$, $\boldsymbol{x}_0$, the 2 bottom most layers solve the (min-max) robust control problem. In other words, the bottom 2 levels compute the inputs that, for the worst-case realization of the uncertainties, drive the states to the lowest level set of the n-dimensional RTRR ellipsoid at the next sampling instance. The top level problem then searches over all initial conditions within the given $\tilde{\mathcal{R}}_z$ to find (if they exist) initial conditions for which the states at the next sampling instance end up

[4] 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.

being driven outside the robust RTRR at the next sampling instance. If the NLP is feasible, it implies that there are no guarantees that the process starting within the given estimate of the robust RTRR will be driven inside the robust RTRR at the next sampling instance in the presence of uncertainties even when implementing the robust control action. On the other hand, if the problem is infeasible, this implies that for every initial condition in the given robust RTRR, the states are always contained within the robust RTRR at the next sampling instance, even for the worst-case effect of the uncertainties. An infeasible solution therefore represents that a valid robust RTRR estimate has been found.

In principle, one can add another layer to the NLP wherein $\left(\tilde{\boldsymbol{c}}_z, \tilde{\mathbf{P}}_z\right)$ are decision variables and the objective is to maximize the "volume" of the n-dimensional ellipsoid. Even if carried out offline, the determination of the largest robust RTRR ellipsoid would become an unwieldy problem. In this work, we address this problem by appropriately pre-selecting $\left(\tilde{\boldsymbol{c}}_z, \tilde{\mathbf{P}}_z\right)$. In particular, Algorithm 2.1 is first performed for 1 sampling period with points in $\tilde{\mathcal{R}}_{z-1}$ substituted for $\boldsymbol{x}_{\text{des}}$ to yield a point set. That is, the system is reverse-time integrated from all the elements in $\tilde{\mathcal{R}}_{z-1}$ (using nominal values of the uncertainties) and all possible input combinations (after discretization)[5]. Then, a minimum volume enclosing ellipsoid (MVEE) is found that best covers this point set. This ellipsoid can be found by solving a convex optimization problem as shown in [31, Chapter 8]. The resulting ellipsoid is the starting $\left(\tilde{\boldsymbol{c}}_z, \tilde{\mathbf{P}}_z\right)$ for the NLP in Equations (2.5a) to (2.5g). If the NLP is feasible for this ellipsoid, the ellipsoid is scaled down by pre-multiplying the ellipsoid matrix by a coefficient greater than 1, and the problem is resolved until the NLP 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 NLP becomes feasible. The final ellipsoid obtained through this (iterative) procedure then represents the (approximately) largest estimate of the robust RTRR, given the pre-selected orientation and centre point of the ellipsoid. The iterative procedure is shown in Figure 2.7.

Remark 2.7: The problem of determining robust RTRRs cannot be addressed by extending the method for generating nominal RTRRs by reverse-time integrating for discretized values of the uncertainties. The only conclusion that can be drawn for a point in such a set is that there exists a pair of inputs and realization of the uncertainties such that the process can be driven to the RTRR at the next sampling instance. No guarantees can be made for the existence of inputs for *any* allowable realization of the uncertainties. This necessitates the development of the multi-level optimization-based method proposed in this section. Note also that the objective in 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 inputs to drive the process inside the next robust RTRR can be guaranteed.

[5] Algorithms for generating uniformly distributed points inside an ellipsoid are reviewed in [30].

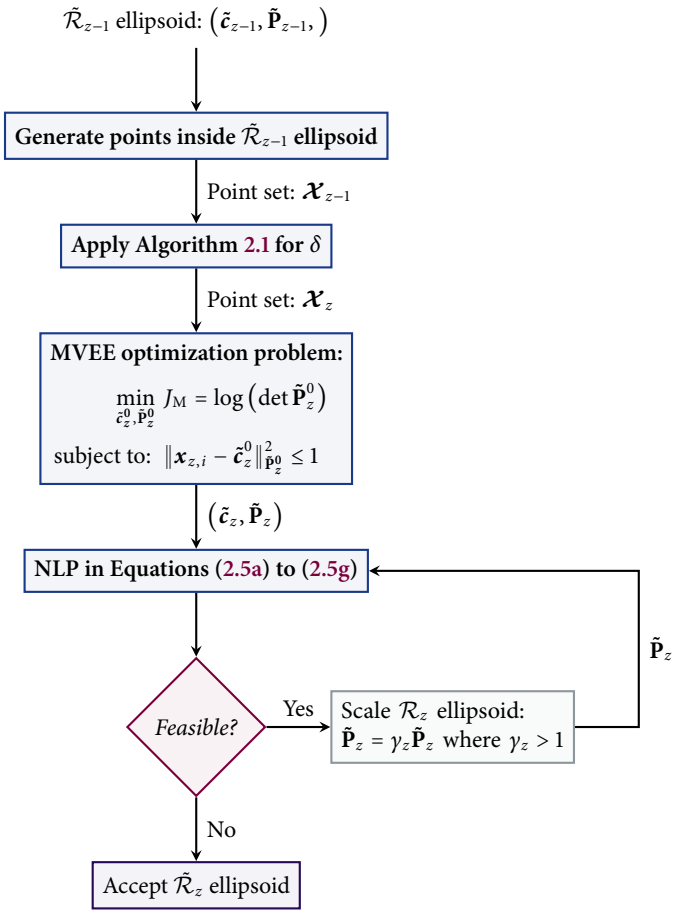


Figure 2.7: Iterative procedure to determine the $\tilde{\mathcal{R}}_z$ ellipsoid estimate

2.4.2 Robust Reverse-time Reachability Region-based Model Predictive Controller

In this section, we present a MPC design that utilizes robust RTRR estimates to steer a batch process inside a desired end-point neighbourhood. As with the RTRR-based formulation, most of the computational burden associated with this design is offline, and the controller is therefore amenable to online implementation. To this end, consider a batch process described by Equation (1.1) for which robust RTRR estimates have been characterized (as ellipsoids).

The control action at sampling instance $z := (t_{\text{end}} - t)/\delta$ is computed by solving the following bi-level NLP:

$$\min_{\boldsymbol{u}[z] \in \mathcal{U}} \; J_{\tilde{R}} \tag{2.6a}$$

$$\text{subject to: } \tilde{\boldsymbol{x}}(0) = \boldsymbol{x}(t) \tag{2.6b}$$

$$\tilde{\boldsymbol{x}}(\delta) = \tilde{\boldsymbol{x}}(0) + \int_0^{\delta} \boldsymbol{f}(\tilde{\boldsymbol{x}}, \boldsymbol{u}[z], \boldsymbol{w}) \, \mathrm{d}\tau \tag{2.6c}$$

$$\|\tilde{\boldsymbol{x}}(\delta) - \tilde{\boldsymbol{c}}_{z-1}\|^2_{\tilde{\mathbf{P}}_{z-1}} \leq 1 \tag{2.6d}$$

$$\max_{\boldsymbol{w} \in \mathcal{W}} \; J_{\boldsymbol{w}} = \|\tilde{\tilde{\boldsymbol{x}}}(\delta) - \tilde{\boldsymbol{c}}_{z-1}\|^2_{\tilde{\mathbf{P}}_{z-1}} \tag{2.6e}$$

$$\text{subject to: } \tilde{\tilde{\boldsymbol{x}}}(\delta) = \tilde{\boldsymbol{x}}(0) + \int_0^{\delta} \boldsymbol{f}(\tilde{\tilde{\boldsymbol{x}}}, \boldsymbol{u}[z], \boldsymbol{w}) \, \mathrm{d}\tau \tag{2.6f}$$

where the objective function, $J_{\tilde{R}}$, can take the form in Equation (2.3). This NLP is formulated in a similar fashion as the bottom 2 levels in the robust RTRR generation NLP in Equations (2.5d) to (2.5g). Following initialization at the current process states (Equation (2.6b)), the worst-case realization of the uncertainties are found from the maximization problem in Equations (2.6e) to (2.6f). The worst-case realization is defined to be the one which drives the states to the highest level set of the $\tilde{\mathcal{R}}_{z-1}$ estimate. For this worst-case realization, the top level searches for inputs that minimize the objective function while ensuring that the states at the next time step are contained within the corresponding robust RTRR estimate. The algorithm used to compute the robust RTRR estimates guarantees the feasibility of this MPC optimization problem with full state feedback. Additionally, by definition, robust RTRRs take into account the requirement to drive the process to a desired end-point neighbourhood. The implications on the guarantees of feasibility and driving the system to a desired end-point neighbourhood are formalized below in Theorem 2.3.

Theorem 2.3: *Consider the batch process described by Equation* (1.1) *under the robust RTRR-based controller in Equations* (2.6a) *to* (2.6f) *with full state feedback. If* $\boldsymbol{x}(t_0) \in \tilde{\mathcal{R}}(t_0)$*, the MPC optimization problem remains feasible for all* $t \in [t_0, t_{end}]$ *and* $\boldsymbol{x}(t_{end}) \in \mathcal{B}(\boldsymbol{x}_{des})$.

Proof: The sufficiency of the condition in Theorem 2.3 can be shown by considering any $\boldsymbol{x}(t_0) \in \tilde{\mathcal{R}}(t_0)$. From the properties of the generation algorithm for $\tilde{\mathcal{R}}(t)$, there exists a set of inputs that take the states inside $\tilde{\mathcal{R}}(t + \delta)$ in a time δ. Repeating this for the duration of the batch implies that the states are driven inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ by t_{end} for all possible realizations of the uncertainties. In essence, this implies that there always exists a feasible solution to the MPC optimization problem in Equations (2.6a) to (2.6f) $\forall\, t \in [t_0, t_{\text{end}}]$ and $\boldsymbol{x}(t_{\text{end}}) \in B(\boldsymbol{x}_{\text{des}})$. This completes the proof of Theorem 2.3.

In Theorem 2.3, the condition, $x(t_0) \in \tilde{\mathcal{R}}(t_0)$, guarantees the existence of a sequence of inputs (via guaranteed feasibility of the robust RTRR-based MPC optimization problem)

to drive the states inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ in the presence of uncertainties. This condition, however, is not a *necessary* condition for driving the states inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ even if we consider exact characterizations of the true robust RTRRs. Consider the case where $\boldsymbol{x}(t_0) \notin \tilde{\mathcal{R}}(t_0)$. While we cannot guarantee the existence of a sequence of inputs to ensure $\boldsymbol{x}(t_{\text{end}}) \in \mathcal{B}(\boldsymbol{x}_{\text{des}})$, this sequence may still exist because it might be possible to drive the states inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ for some realization of the uncertainties (if not for all realizations as required by 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 inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$.

Remark 2.8: One characteristic of batch processes is that the desired end-point, which is based on the values of quality variables at batch termination, typically remains consistent batch-to-batch unless a new product is being manufactured. The main source of variation between batches is usually the initial condition as this is dictated by raw material properties, which are subject to variance depending on their source. The robust RTRR-based controller 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 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.

2.5 SAFE-STEERING FRAMEWORK

In the previous section, we first presented a (robust) RTRR-based predictive controller that was designed with a fault-free assumption. As discussed in Section 2.1, for batch processes, the problem of fault tolerance is fundamentally different than in continuous processes. Specifically, a FTC 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 (robust) RTRR-based MPC design to develop what we call the **safe-steering** framework. First, the safe-steering problem is formulated and then the safe-steering framework is presented.

2.5.1 Problem Definition

For processes described by Equation (1.1), we consider faults in the control actuators under the assumption that upon failure, the available control effort is reduced. 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} as $u_{1,\text{min, fault}} \leq u_1(t) \leq u_{1,\text{max, fault}}\ \forall\, t \in [t_{\text{fault}}, t_{\text{repair}})$ where u_i denotes the i-th component of $\boldsymbol{u}$ and $u_{1,\text{min, fault}}$ and $u_{1,\text{max, fault}}$ denote the minimum and maximum values of u_1 during the fault (respectively)[6].

[6] If t_{fault} is not an integer multiple of the sampling period, it can be taken to be at the upcoming integer multiple, and the safe-steering framework can be implemented as presented in this section.

Reduced control effort corresponds to a situation where $u_{1,\min,\,\text{fault}} > u_{1,\min}$ and $u_{1,\max,\,\text{fault}} < u_{1,\max,\,\text{fault}}$ as shown in Figure 2.8. In the case where an actuator reverts to a completely open or shut position, we have $u_{1,\min,\,\text{fault}} = u_{1,\max,\,\text{fault}}$.

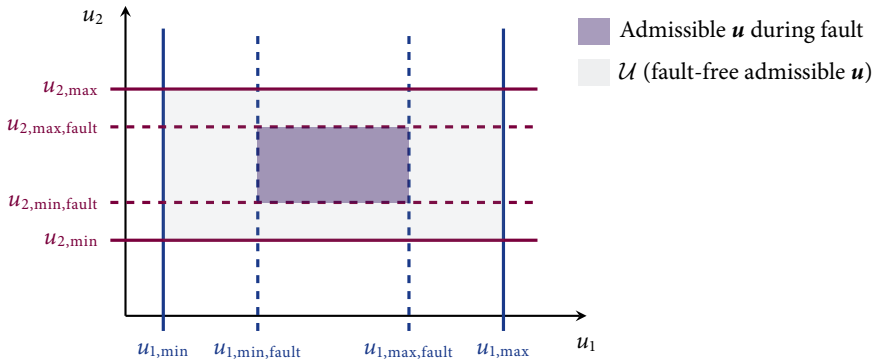


Figure 2.8: Illustration of the reduced available control effort during a fault. Note the smaller area corresponding to the admissible $\boldsymbol{u}$ during the fault.

We define the safe-steering problem as the one of identifying a sequence of the functioning 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 uncertainties that will ensure the process can be driven inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ upon recovery of the full control effort.

2.5.2 *Safe-steering to a Desired End-Point Neighbourhood*

The key idea in the safe-steering problem is to preserve the states within robust RTRRs during the failure period by employing the robust RTRR-based MPC design. By doing so, the robust RTRR-based MPC design is able to drive the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ following fault repair. Note also that the ability to steer the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ after fault repair is dependent on the duration of the fault. To this end, consider a batch process described by Equation (1.1) for which the first control actuator fails at t_{fault} and is repaired at t_{repair}, and the robust RTRR estimates 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 2.4.

Theorem 2.4: *Consider the batch process described by Equation* (1.1) *with* $\boldsymbol{x}(t_0) \in \tilde{\mathcal{R}}(t_0)$ *under the robust RTRR-based controller in Equations* (2.6a) *to* (2.6f)*. If the MPC optimization problem remains feasible* $\forall\, t \in [t_{fault}, t_{repair}]$*, then* $\boldsymbol{x}(t_{end}) \in \mathcal{B}(\boldsymbol{x}_{des})$.

Proof: The proof of this theorem follows from Theorem 2.3. Equating t_{repair} to t_0 results in the satisfaction of the requirements of Theorem 2.3, and therefore, the MPC optimization problem in Equations (2.6a) to (2.6f) continues to remain feasible $\forall\, t \in [t_{\text{repair}}, t_{\text{end}}]$ and $\boldsymbol{x}(t_{\text{end}}) \in \mathcal{B}(\boldsymbol{x}_{\text{des}})$ follows. This completes the proof of Theorem 2.4.

The key idea formally expressed in Theorem 2.4 is that if a fault is repaired sufficiently fast, meaning if there exists an implementable sequence of inputs during the fault repair period and one after fault repair, the robust RTRR-based MPC design finds this sequence via preserving the states within robust RTRRs. The implication of this is that process states at t_{repair} will then belong to $\tilde{\mathcal{R}}(t_{\text{repair}})$, and according to the definition of robust RTRRs, the process can then be driven inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. Therefore, maintaining the process within the robust RTRRs provides a sufficient condition to ensure that $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ can be reached upon fault recovery. In other words, the proposed robust RTRR-based MPC design is able to identify the sequence of inputs during faulty operation (if one exists) that will enable reaching $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ upon fault recovery. In contrast, end-point-based MPC approaches can fail to find this sequence even if it exists. The end-point-based 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 $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ is impossible even after fault recovery.

Theorem 2.4 provides sufficient conditions for fault-tolerant control in batch processes. To address the issue of necessary conditions, we note that if the fault is repaired too late, it can become impossible to preserve the states within 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 that the states at batch termination will be outside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. This is because the states at t_{repair} in this situation could reside in a region for which there exists a specific realization of the uncertainties and corresponding sequence of inputs that can drive the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. This can occur since $\boldsymbol{x}(t_{\text{repair}}) \in \tilde{\mathcal{R}}(t_{\text{repair}})$ is a *sufficient* but not necessary condition for driving the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$.

Remark 2.9: The safe-parking and safe-steering frameworks for continuous and batch systems, respectively, address the problem of how to operate a process during fault recovery, and both frameworks address the kind of faults that prevent the desired plant operation under nominal control laws. Safe-parking handles faults that preclude operation at a nominal equilibrium point while safe-steering addresses the kind of faults that preclude driving the states to a desired end-point. The safe-steering and safe-parking framework both answer the question of how to compute the inputs 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. In the presence of faults that prevent operation at a desired equilibrium point, safe-parking [16, 25] involves transition to a new safe-park (equilibrium) point that allows the transition back to the nominal equilibrium in some optimal way following fault recovery. Prior to considering any optimality criteria regarding the transitions, the safe-parking framework must locate the feasible operating points that allow such tran-

sitions, 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 [14, 32] 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 processes 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 points, the process cannot be “parked", but possibly “steered" in a way that allows for desired end-point reachability if the fault is rectified sufficiently fast. This is achieved in the safe-steering framework via using the proposed robust RTRR-based MPC design.

2.6 SIMULATION EXAMPLE: FED-BATCH REACTOR

In this section, we first consider a fault-free environment and demonstrate the need for accounting for uncertainty when using a RTRR-based control design. We then illustrate the safe-steering framework by considering an actuator failure. To this end, consider a fed-batch reactor where an irreversible, first-order exothermic reaction of the form:

$$A \xrightarrow{k_A} B$$

takes place. The state-space model (derived using regular modelling assumptions) for this process takes the following form:

$$\dot{x}_1(t) = -k_{A0} \exp\left\{\frac{E}{R}\left(\frac{1}{T_R} - \frac{1}{x_2}\right)\right\} x_1 + \frac{u_2(C_{A0} - x_1)}{x_3} \tag{2.7a}$$

$$\dot{x}_2(t) = \frac{w_2(w_1 - x_2)}{\rho C_p x_3} + \frac{u_2(u_1 - x_2)}{x_3} - k_{A0} \exp\left\{\frac{E}{R}\left(\frac{1}{T_R} - \frac{1}{x_2}\right)\right\} \frac{x_1 \Delta H}{\rho C_p} \tag{2.7b}$$

$$\dot{x}_3(t) = u_2 \tag{2.7c}$$

The states are the concentration of reactant A, reactor temperature, and volume, which are denoted by C_A, T, and V (respectively); thus, we have: $\boldsymbol{x} = \begin{bmatrix} C_A & T & V \end{bmatrix}'$. Uncertainties in the inlet temperature, T_{in} (K), and the heat exchanger coefficient, UA (cal/(h $\cdot$ K)), of ±5% around their nominal values were considered. That is, $\boldsymbol{w} = \begin{bmatrix} T_{\text{in}} & UA \end{bmatrix}'$ with bounds defined by $\boldsymbol{w}_{\min} = \begin{bmatrix} 278.35 & 9.50 \times 10^3 \end{bmatrix}'$ and $\boldsymbol{w}_{\max} = \begin{bmatrix} 307.65 & 1.05 \times 10^4 \end{bmatrix}'$. The uncertainty in T_{in} 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 due to the effects of fouling. The inputs were taken to be the heating coil temperature T_{hx} (K), and inlet feed rate, F (L/h); thus, we had $\boldsymbol{u} = \begin{bmatrix} T_{\text{hx}} & F \end{bmatrix}$, with constraints, $\boldsymbol{u}_{\min} = \begin{bmatrix} 0 & 285 \end{bmatrix}'$ and $\boldsymbol{u}_{\max} = \begin{bmatrix} 25 & 400 \end{bmatrix}'$. The physical meaning of the model parameters and their nominal values can be found in Table 2.1.

Table 2.1: Parameters of the fed-batch reactor model in Equations (2.7a) to (2.7c)

Parameter	Description	Value	Unit
k_{A0}	Reaction rate constant at T_R for $A \rightarrow B$ at T_R	0.15	1/h
E	Activation energy for $A \rightarrow B$	10^4	cal/mol
T_R	Reference temperature at which k_{A0} is computed	290	K
C_{A0}	Inlet A concentration	5	mol/L
UA	Heat transfer coefficient × Area	10^4	cal/(h · K)
ρ	Density of the solution and inlet feed	1	kg/L
C_p	Heat capacity of the solution and inlet feed	65	cal/(kg · K)
T_{in}	Temperature of inlet feed stream	293	K
ΔH	Heat of reaction $A \rightarrow B$	−4000	cal/mol
R	Universal gas constant	1.986	cal/(mol · K)

To demonstrate the applicability of the proposed RTRR-based controllers with limited measurements, we considered the case where only noisy measurements of T and V were available, $\boldsymbol{y} = \begin{bmatrix} T & V \end{bmatrix}'$. The noise was assumed to be normally distributed with zero-mean and standard deviations of 0.15 and 0.05 for T and V, respectively. To estimate C_A using these measurements, an extended Luenberger observer (ELO) of the form shown below was used[7].

$$\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{f}(\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{w}_{\text{nom}}) + \mathbf{L}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{2.8a}$$

$$\hat{\boldsymbol{y}} = \mathbf{C}\hat{\boldsymbol{x}} \tag{2.8b}$$

where $\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{C}_A & \hat{T} & \hat{V} \end{bmatrix}'$ and $\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{T} & \hat{V} \end{bmatrix}'$ denote vectors of the estimated states and outputs (respectively), $\mathbf{L}$ is an ELO gain matrix, and $\mathbf{C}$ is given by:

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For this example, the model was successively linearized at the current state estimates, the computed input values, and the nominal realization of the model parameters and uncertainties, and $\mathbf{L}$ was 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 (i, j) element of $\mathbf{A}$ was given by:

$$a_{i,j} = \left.\frac{\partial f_i}{\partial x_j}\right|_{\hat{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{w}_{\text{nom}}}$$

[7] This meant that the observer system in Equations (2.8a) to (2.8b) was simulated along with the plant, and the states of the observer system were used to initialize the controller.

where f_i is the differential equation associated with state i. A schematic of this process is given in Figure 2.9.

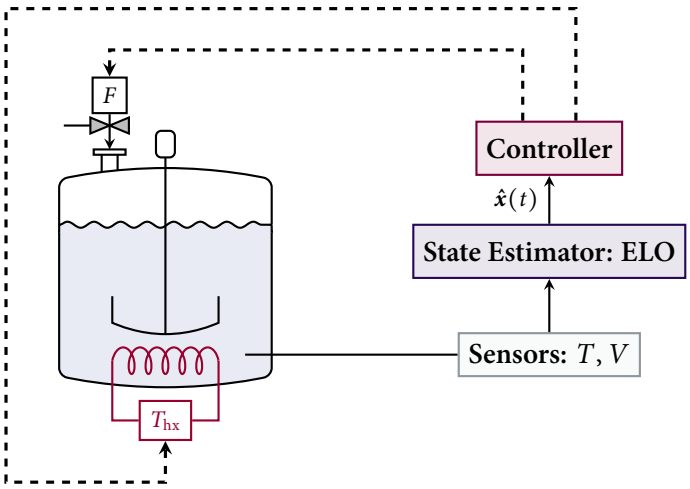


Figure 2.9: Schematic of the fed-batch reactor process

The primary control objective considered was to achieve the desired end-point of $\boldsymbol{x}_{\text{des}} = \begin{bmatrix} 0.1 & 465 & 65 \end{bmatrix}'$. The desired neighbourhood around $\boldsymbol{x}_{\text{des}}$, $\mathcal{B}(\boldsymbol{x}_{\text{des}})$, was chosen to be an ellipsoid with $\tilde{\mathbf{P}}_0 = \text{diag}\left\{10^4, 4, 11.1\bar{1}\right\}$ and $\tilde{\boldsymbol{c}}_0 = \boldsymbol{x}_{\text{des}}$. The total batch time was taken to be $t_{\text{end}} = 0.5$ hours with a sampling period of $\delta = 0.01$ hours.

2.6.1 *Fault-free Closed-loop Results*

To demonstrate the need for accounting for uncertainties, closed-loop simulations were performed using the nominal RTRR-based MPC design in Section 2.3.2 and the robust design in Section 2.4.2 under fault-free conditions. First, for the given $\mathcal{B}(\boldsymbol{x}_{\text{des}})$, bounds of the uncertainties, and input constraints, nominal and robust RTRRs were generated and characterized with ellipsoids for all sampling instances using the algorithm described in Section 2.4.1[8].

The *nominal* RTRR-based MPC formulation could encounter infeasibility (due to the reachability constraint) since the nominal RTRRs were generated assuming no model uncertainties. Additionally, while the robust RTRR-based formulation explicitly accounted for uncertainties, state estimation errors could potentially result in its MPC optimization problem turning infeasible. To avoid this infeasibility problem, the desirable reachability properties of the controllers were achieved through the objective function (as opposed to hard constraints) wherein the deviation between the process states and the centre of the

[8] When generating RTRR ellipsoids without uncertainties, the lowest layer of the NLP in Equations (2.5a) to (2.5g) was removed.

RTRR ellipsoid at the next time step was penalized[9]. The final form of the objective function, which also included a move suppression term, took the following form:

$$J_{\{R,\tilde{R}\}} = \|\tilde{\boldsymbol{x}}(\delta) - \tilde{\boldsymbol{c}}_{z-1}\|^2_{\tilde{\mathbf{P}}_{z-1}} + \|\boldsymbol{u}[z+1] - \boldsymbol{u}[z]\|^2_{\boldsymbol{\Pi}}$$

Using this objective function was useful for ensuring the process states were maintained in a region of the state-space from where the control objective was always within reach.

The tuning parameters, initial conditions, and results for both controllers are summarized in Table 2.2. The initial states and their estimates were both chosen such that they resided in the suitable (nominal or robust) RTRR at t_0.

Table 2.2: Tuning parameters, initial conditions, and results for the nominal and robust RTRR-based MPC designs in a fault-free environment

	Nominal RTRR-based MPC	**Robust RTRR-based MPC**
Move suppression matrix, $\boldsymbol{\Pi}$:	$\text{diag}\{5\times10^{-4}, 9\times10^{-4}\}$	
ELO eigenvalues:	$\{-0.9, -1.05, -1.1\}$	
Initial states, $\boldsymbol{x}(0)$:	$[2.65 \quad 275.16 \quad 59.88]'$	$[2.65 \quad 279.40 \quad 58.48]'$
Initial state estimates, $\hat{\boldsymbol{x}}(0)$:	$[2.68 \quad 276.79 \quad 59.09]'$	$[2.66 \quad 281.02 \quad 58.69]'$
Mean CPU time/MPC calculation:†	0.146 seconds	1.507 seconds
Final states, $\boldsymbol{x}(t_{\text{end}})$:	$[0.19 \quad 466.45 \quad 64.21]'$	$[0.099 \quad 464.84 \quad 64.82]'$
$\|\boldsymbol{x}(t_{\text{end}}) - \tilde{\boldsymbol{c}}_0\|^2_{\tilde{\mathbf{P}}_0}$:	82.16	0.49

† This was computed using the Matlab functions, `tic` and `toc`, on an Intel Quad Core Machine. The MPC optimization problem was solved using the `fmincon` function in Matlab.

For the nominal case, $\boldsymbol{x}(t_{\text{end}})$ corresponded to a level set of 82.16 of $\mathcal{B}(\boldsymbol{x}_{\text{des}})$, which was well outside the desired end-point neighbourhood[10]. On the other hand, the robust RTRR-based controller was able to steer the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ as its final states corresponded to a level set of 0.49. These results indicate the practical importance of explicitly accounting for uncertainties in the controller design. The computation times for both controllers indicate that they would be amenable for real-time implementation. The higher average CPU time per MPC calculation for the robust controller was certainly expected as the MPC optimization problem was more complex; however, this computational trade-off is well worth the savings acquired from achieving an end-point within the desired neighbourhood.

2.6.2 Safe-steering Results

Next, to demonstrate the effectiveness of the safe-steering framework, we considered a fault in a control actuator. Specifically, we considered the scenario where at $t_{\text{fault}} = 0.25$ hours, the

[9] In other words, the hard reachability constraints were made soft constraints.
[10] This follows from the definition of an ellipsoid in Equation (2.4).

heating coil actuator failed. During the failure period, the heating coil could supply limited heat to the reactor, and the heating coil temperature was restricted to 285 K $\leq T_{hx} \leq$ 300 K. At t_{repair} = 0.32 hours, the fault was rectified and full control effort was recovered.

For these simulations, the robust RTRR-based MPC design was bench-marked against a generic end-point-based MPC formulation. For the end-point-based controller, the objective function shown below was used (see Equations (2.1a) to (2.1d)).

$$J_E = \|\tilde{\boldsymbol{x}}(t_{end} - t) - \tilde{\boldsymbol{c}}_0\|^2_{\tilde{\mathbf{P}}_0} + \sum_{i=k}^{K} \|\boldsymbol{u}[i] - \boldsymbol{u}[i-1]\|^2_{\boldsymbol{\Pi}}$$

where k is the current batch sampling instance and K is the sampling instance corresponding to t_{end}. The objective function for the RTRR-based MPC design was kept the same as in the fault-free simulations. Table 2.3 summarizes the tuning parameters, initial conditions, and results from both control designs.

Table 2.3: Tuning parameters, initial conditions, and results for the robust RTRR-based and end-point-based MPC designs in a faulty environment

	RTRR-based MPC	**End-point-based MPC**
Move suppression matrix, $\boldsymbol{\Pi}$:	$\text{diag}\{5 \times 10^{-5}, 5 \times 10^{-6}\}$	
ELO eigenvalues:	$\{-0.9, -1.05, -1.1\}$	
Initial states, $\boldsymbol{x}(0)$:	$[2.65 \quad 282.71 \quad 58.49]'$	
Initial state estimates, $\hat{\boldsymbol{x}}(0)$:	$[2.66 \quad 284.34 \quad 58.70]'$	
Total simulation time:[†]	2.13 minutes	2.32 hours
Final states, $\boldsymbol{x}(t_{end})$:	$[0.10 \quad 464.88 \quad 64.85]'$	$[0.12 \quad 464.66 \quad 64.69]'$
$\|\boldsymbol{x}(t_{end}) - \tilde{\boldsymbol{c}}_0\|^2_{\tilde{\mathbf{P}}_0}$:	0.39	5.99

[†] This was computed using the Matlab functions, `tic` and `toc`, on an Intel Quad Core machine. The MPC optimization problem was solved using the `fmincon` function in Matlab.

From Table 2.3, we first note that the total simulation time required for the robust RTRR-based MPC was significantly shorter compared to the end-point-based MPC design. The reasoning behind reporting the total simulation time as opposed to the CPU time per MPC calculation was that in the end-point-based controller, the number of decision variables changed at each sampling instance due to the shrinking horizon nature of the MPC problem. By comparison, the RTRR-based controller always considered only one step ahead. Simulations of a process with a higher number of states and/or inputs would exhibit an even more substantial difference in the simulation times. Moreover, with additional model uncertainties, wider uncertainty ranges, and the introduction of disturbances into the system, the end-point-based MPC design would require additional computational time because the solution at a given sampling instance would become a poorer initial guess for the next

sampling instance. On the other hand, because the robust RTRR-based MPC formulation accounts for the presence of the uncertainties and its bounds in *offline* calculations (which would certainly increase), the computation time for *online* control calculations would not increase significantly.

The level set of $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ corresponding to $\boldsymbol{x}(t_{\text{end}})$ were 0.38 and 5.99 for the RTRR and end-point-based MPC designs (respectively); thus, in contrast to the RTRR-based MPC design, end-point-based MPC was unable to recover the process following fault repair. The states at batch termination (relative to $\mathcal{B}(\boldsymbol{x}_{\text{des}})$) and input profiles for the two MPC designs are shown in Figure 2.10 and Figure 2.11, respectively.

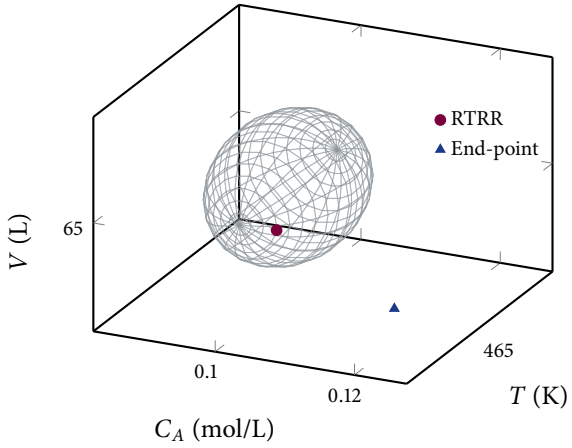


Figure 2.10: States at batch termination from the RTRR and end-point-based MPC designs with a finite duration actuator fault for the fed-batch process

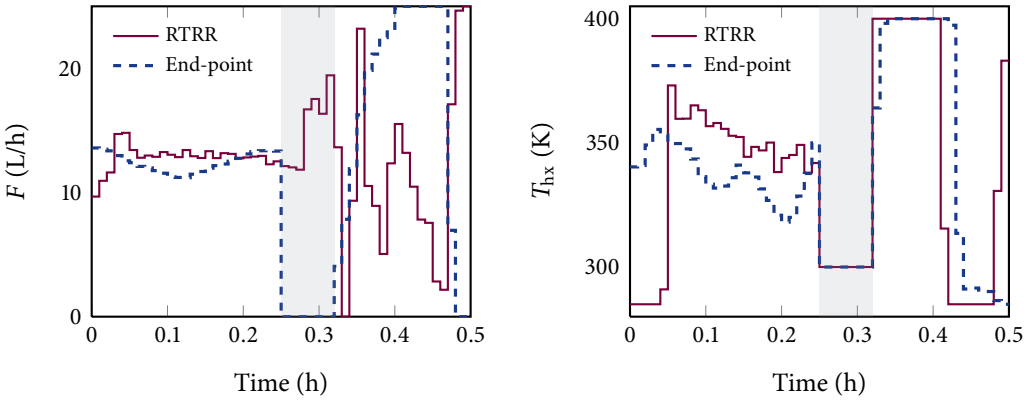


Figure 2.11: Input profiles prescribed by the RTRR and end-point-based MPC designs with with a finite duration actuator fault for the fed-batch process. The failure period is shaded.

During the failure period, the heating coil temperatures prescribed by both controllers were saturated at 300 K (the maximum fault value) while the flow rate trajectories were markedly different. Using only the flow rate, the end-point MPC design was unable to compute a sequence that steered the process inside the desired end-point neighbourhood. Due to the repeated application of a truncated version of these poor input trajectories, the process was driven to a point by t_{repair} from where it could not be steered inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ even after fault recovery. In contrast, the robust RTRR-based MPC design prescribed flow rates during the failure period which maintained the process states from where the batch could be driven inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ upon recovery of the full control effort.

2.7 CONCLUSIONS

In this chapter, we considered the control of batch processes subject to input constraints, model uncertainties, and faults with the objective of reaching a desired end-point neighbourhood. To this end, a computationally efficient, nonlinear robust MPC design based on robust RTRRs was formulated. Prior to the MPC formulation, a multi-level optimization-based algorithm was developed to generate/characterize robust RTRRs as ellipsoids for specified bounds of the model uncertainties, sampling period, and desired end-point neighbourhood. Following the controller design, we considered the problem of finite duration faults in the control actuators that cannot be handled via robust control approaches. Using the robust RTRR-based controller 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 the desired end-point neighbourhood 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 reactor process.

REFERENCES

[1] S. Aumi and P. Mhaskar, "Safe-steering of batch processes," *AIChE J.*, vol. 55, pp. 2861–2872, 2009.

[2] —, "Robust model predictive control and fault-handling of batch processes," *AIChE J.*, vol. 57, pp. 1796–1808, 2010.

[3] —, "Robust model predictive control and fault-handling of batch processes," in *Proc. of the American Control Conf.*, Baltimore, MD, 2010, pp. 4415–4420.

[4] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 1.21*, http://cvxr.com/cvx, Apr. 2011.

[5] D. Q. Mayne, "Non-linear model predictive control: challenges and opportunities," in *Non-Linear Model Predictive Control.* F. Allgöwer and A Zheng, Eds., Birkhäuser, Basel, 2000, pp. 23–44.

[6] B. Srinivasan, S. Palanki, and D. Bonvin, "Dynamic optimization of batch processes: I. Characterization of the nominal solution," *Comp. & Chem. Eng.*, vol. 27, no. 1, pp. 1–26, 2003.

[7] B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki, "Dynamic optimization of batch processes: II. Role of measurements in handling uncertainty," *Comp. & Chem. Eng.*, vol. 27, no. 1, pp. 27–44, 2003.

[8] J. Valappil and C. Georgakis, "State estimation and nonlinear model predictive control of end-use properties in batch reactors.," in *Proc. of the American Control Conf.*, vol. 2, Arlington, VA, 2001, pp. 999–1004.

[9] M. Alamir and I. Balloul, "Robust constrained control algorithm for general batch processes.," *Int. J. Contr.*, vol. 72, pp. 1271–1287, 1999.

[10] T. Crowley and K. Choi, "Experimental studies on optimal molecular weight distribution control in a batch-free radical polymerization processes.," *Chem. Eng. Sci.*, vol. 53, no. 15, pp. 2769–2790, 1998.

[11] J. Dimitratos, C. Georgakis, M. S. El-Aasser, and A. Klein, "Dynamic modeling and state estimation for an emulsion copolymerization reactor.," *Comp. & Chem. Eng.*, vol. 13, no. 1-2, pp. 21–33, 1989.

[12] D. J. Kozub and J. F. MacGregor, "Feedback control of polymer quality in semi-batch copolymerization reactors," *Chem. Eng. Sci.*, vol. 47, no. 4, pp. 929–942, 1992.

[13] P. De Vallière and D. Bonvin, "Application of estimation techniques to batch reactors. II. Experimental studies in state and parameter estimation.," *Comp. & Chem. Eng.*, vol. 13, no. 1-2, pp. 11–20, 1989.

[14] P. Mhaskar, N. H. El-Farra, and P. D. Christofides, "Predictive control of switched nonlinear systems with scheduled mode transitions," *IEEE Trans. Automat. Contr.*, vol. 50, pp. 1670–1680, 2005.

[15] P. Mhaskar, "Robust model predictive control design for fault-tolerant control of process systems.," *Ind. & Eng. Chem. Res.*, vol. 45, pp. 8565–8574, 2006.

[16] M. Mahmood, R. Gandhi, and P. Mhaskar, "Safe-parking of nonlinear process systems: Handling uncertainty and unavailability of measurements," *Chem. Eng. Sci.*, vol. 63, pp. 5434–5446, 2008.

[17] Z. D. Wang, B. Huang, and H. Unbehauen, "Robust reliable control for a class of uncertain nonlinear state-delayed systems," *Automatica*, vol. 35, pp. 955–963, 1999.

[18] L. L. Cheng, K. E. Kwok, and B. Huang, "Closed-loop fault detection using the local approach," *Can. J. Chem. Eng.*, vol. 81, 1101–1108, 2003.

[19] N. Mehranbod, M. Soroush, and C. Panjapornpon, "A method of sensor fault detection and identification," *J. Process Control*, vol. 15, pp. 321–339, 2005.

[20] P. Mhaskar et al., "Integrated fault-detection and fault-tolerant control for process systems," *AIChE J.*, vol. 52, pp. 2129–2148, 2006.

[21] N. H. El-Farra, "Integrated fault detection and fault-tolerant control architectures for distributed processes," *Ind. & Eng. Chem. Res.*, vol. 45, pp. 8338–8351, 2006.

[22] P. Mhaskar, C. McFall, A. Gani, P. D. Christofides, and J. F. Davis, "Isolation and handling of actuator faults in nonlinear systems.," *Automatica*, vol. 44, pp. 53–62, 2008.

[23] N. H. El-Farra and A. Giridhar, "Detection and management of actuator faults in controlled particulate processes using population balance models," *Chem. Eng. Sci.*, vol. 63, 1185–1204, 2008.

[24] A. Armaou and M. A. Demetriou, "Robust detection and accommodation of incipient component and actuator faults in nonlinear distributed processes," *AIChE J.*, vol. 54, pp. 2651–2662, 2008.

[25] R. Gandhi and P. Mhaskar, "Safe-parking of nonlinear process systems," *Comp. & Chem. Eng.*, vol. 32, pp. 2113–2122, 2008.

[26] P. Terwiesch, M. Agarwal, and D. W. T. Rippin, "Batch unit optimization with imperfect modelling: a survey.," *J. Process Control*, vol. 4, pp. 238–258, 1994.

[27] CSE Group at University of California, Santa Barbara, *DASSL: Differential Algebraic System Solver, DASPK2.0: Large Scale Differential Algebraic Equation Solver*, `http://www.cs.ucsb.edu/~cse/software.html`, Nov. 2011.

[28] N. Kapoor and P. Daoutidis, "Stabilization of nonlinear processes with input constraints," *Comp. & Chem. Eng.*, vol. 24, pp. 9–21, 2000.

[29] H. K. Khalil, *Nonlinear Systems*, Third. New York, NY: Prentice Hall, 2002.

[30] H. Sun and M. Farooq, "Note on the generation of random points uniformly distributed in hyper-ellipsoids," in *Proc. of the Fifth International Conf. on Information Fusion*, vol. 1, 2002, pp. 489–496.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*, Seventh. Cambridge, UK: Cambridge University Press, 2004.

[32] P. Mhaskar, N. El-Farra, and P. Christofides, "Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control," *Syst. & Contr. Lett.*, vol. 55, pp. 650–659, 2006.

CHAPTER 3

Integrating Data-based Modelling and Nonlinear Control Tools for Batch Process Control

The results in this chapter are set to appear in or have been published in:

JOURNAL PAPERS:

[1] S. Aumi and P. Mhaskar, "Integrating data-based modelling and nonlinear control tools for batch process control," *AIChE J.*, 2011, (in press). DOI: 10.1002/aic.12720.

[2] S. Aumi, B. Corbett, P. Mhaskar, and T. Clarke-Pringle, "Data-based modelling and control of nylon-6,6 batch polymerization," *IEEE Trans. Control. Sys. Technol.*, 2011, (in press).

REFEREED CONFERENCE PROCEEDINGS:

[3] S. Aumi and P. Mhaskar, "Integrating data-based modelling and nonlinear control tools for batch process control," in *Proc. of the American Control Conf.*, San Francisco, CA, 2011, pp. 2534–2539.

[4] S. Aumi, B. Corbett, and P. Mhaskar, "Data-based modelling and control of nylon-6,6 batch polymerization," in *Proc. of the American Control Conf.*, San Francisco, CA, 2011, pp. 2540–2545.

3.1 INTRODUCTION

Existing batch control approaches (model-based or otherwise) can be classified according to whether the desired end-point is specified directly or indirectly in the control design. For the RTRR-based controller in Chapter 2, the desired end-point was pursued directly because of the assumptions that the final product quality was specified through the states (i.e., $\boldsymbol{x}_{\text{des}}$) and that the states were observable from the process measurements. In some batch processes, however, measurements related to the final end-use quality are unavailable and quality measurements are only made offline after batch completion, making direct control to the desired end-point impractical. For these cases, one response has been to indirectly pursue the control objective through *trajectory tracking* approaches.

In trajectory tracking methods, trajectories for a set of *measurable* process variables related to the end-use quality are generated offline or recalculated at specific time points during the batch by solving a dynamic optimization problem. These trajectories are subsequently tracked using local, model-based controllers or proportional-integral-derivative (PID) controllers, possibly modified with gain-scheduling [5] or feed-forward [6] terms to partially account for process nonlinearities. Even with the improvements proposed in existing literature [5, 6], PID controllers remain inherently based on a decentralized (single-input-single-output) framework that cannot account for interactions between the different control loops, process constraints, and optimality. Explicitly nonlinear model-based tracking controllers have been proposed in the form of feedback linearizing differential geometric controllers [6–8] and model predictive control (MPC) [9–13]. A differential geometric controller takes the form of an algebraic control law that is obtained by appropriately inverting the process model; these controllers, however, can be sub-optimal for a given control objective and are generally incapable of handling process constraints. MPC, on the other hand, as discussed in Chapter 1 is well-suited for handling constraints and optimality.

A common, underlying assumption in the significant literature addressing MPC for batch processes is the availability of an accurate, first-principles-based deterministic process model. In deterministic modelling, differential equations for the process states are derived from first-principles[1] with some parameters in the equations to be determined from experimental data. One of the limitations with the development of these models is the lack of sufficient measurements to uniquely determine the key model parameters, and even when available, many of the simplifying assumptions taken during model development can be violated in specific situations in practice and/or the model is inapplicable for online control applications due to the nature of the equations. For instance, the number of states may be excessive and/or the differential equations overly complex (i.e., discontinuities, etc.) for use in any model-based control design.

[1] This includes conservation equations (i.e., mass and energy balances) and models for reaction kinetics (dictated by the reaction chemistry).

Some of the limitations with deterministic modelling can be overcome through empirical or data-based modelling. Empirical model development imposes a simpler model structure (often linear) on the process dynamics, and the model parameters are subsequently determined entirely from plant/experimental data. Identification experiments (to build empirical models), such as those in which a pseudo-random binary signal (PRBS) is applied on the process, while suitable for model identification at steady-states, are often too expensive to justify for batch processes since they result in expensive wasted batches. For batch processes, most of the identification data takes the form of historical databases, which consist of process variable measurements taken at regular sampling intervals until batch termination for a number of previous batches. Furthermore, batch process dynamics are highly nonlinear and time-varying, making conventional system identification approaches where a single *linear* model is identified, ill-suited for identifying accurate models. The high expenses associated with every batch dictate the need for the development of dedicated modelling tools for batch processes that minimize wasted batches in the model development process and yet provide a model that captures the essential nonlinear and complex nature of the process.

A popular technique to improve the quality of data-based batch process models has been to exploit the availability of measurements in the databases beyond those designated as inputs and outputs. While these measurements are often (auto and/or cross) correlated, they contain important information about the states, implying an accurate model could be identified if they are utilized in the model development. This has motivated the application of latent variable modelling methods, particularly partial least squares (PLS) regression, for identifying batch process models.

Latent variable modelling methods are useful for reducing the dimensions of a large correlated data set into a set of fewer uncorrelated variables. This is achieved by projecting the data set onto subspaces, called latent variable spaces, defined by principal components. In PLS regression, the regressor and response matrices are both projected onto their corresponding latent variable spaces, and the idea is to find the orientation of these two subspaces such that the correlation among them is maximized (see [14] for a tutorial on PLS regression). Although conventionally applied to static (i.e., steady-state) data, by introducing lagged data matrices into the PLS regression algorithm, a dynamic model can be readily obtained (e.g., see [15, 16]). Although latent variable tools are useful for utilizing all the available information in batch databases, the inherent limitation with existing approaches is the assumption of a linear relationship between the latent variables, which is often not valid in batch processes. Some approaches to incorporate nonlinear relationships into the PLS framework include the work in [17–19]; the predictive capability of these models, however, depends on an appropriate choice for the nonlinear mapping (quadratic functions, neural networks, splines, etc.).

One general strategy to describe nonlinear behaviour while retaining the simplicity of linear models is to partition/cluster the training data into a number of different regions,

identify local linear models for each region, and combine them with appropriate weights to describe the global nonlinear behaviour. This idea has been formalized in piece-wise affine (PWA) [20, 21], Takagi, Sugeno, and Kang (TSK) [22], and operating-regime based [15] modelling. Note also that there exist predictive controller designs that can explicitly accommodate multiple linear models (e.g. see [23–25]). In this chapter, we propose a new multi-model approach specific to batch processes that unifies the concepts of auto-regressive exogenous (ARX) modelling, latent variable regression techniques, fuzzy c-means clustering, and multiple local linear models in an integrated framework capable of capturing the nonlinearities and multivariate nature of batch data. The key delineating aspects of this work is the bringing together of the clustering algorithm used to partition the training data, the use of latent variable tools to estimate the model parameters, and the derivation of a generalized continuous weighting function that is entirely data dependent and does not require precise process knowledge. Additionally, the resulting model is readily applicable in a MPC framework.

As discussed in Chapter 2, the ability to handle faults is an intrinsic requirement of the control design for batch processes since a fault can ruin the entire batch. With data-based MPC designs, it also becomes imperative for the model to maintain its validity for a wide range of operating conditions since faults can drive the process significantly away from typical operating conditions. The existing fault-tolerant control structures (FTCS) for batch processes are mostly robust control designs that employ a deterministic model and treat faults as disturbances. However, upon fault occurrence, the final product quality can become unreachable if the fault is not repaired sufficiently fast. Additionally, implementing inputs prescribed by controllers with limited fault-tolerant properties can drive the states to a point from where the final quality becomes permanently unreachable. In response to these issues, we developed the control and safe-steering framework in Chapter 2 that utilized a first-principles model and presented a computationally efficient MPC design that addressed the problem of determining how to utilize functioning inputs during fault rectification to enable desired product properties reachability following fault repair. The proposed design represented a computationally efficient framework that is amenable for integration with appropriately derived data-based models for FTC of batch processes.

Motivated by the above considerations, this chapter considers the problem of designing an integrated framework seamlessly merging data-based models with nonlinear control tools for the control of batch processes. The rest of this chapter is organized as follows: First, the class of processes considered is presented followed by reviews of the key concepts required to understand the modelling technique, namely ARX modelling, PLS regression, and fuzzy c-means clustering. We also review latent variable MPC, which was proposed in [26] and later serves as a basis of comparison for the simulations. Next, a framework is presented for developing a data-based model for a batch process that makes use of all

existing measurements in a preexisting database and captures the nonlinearities of the process. The resulting model is then incorporated within the RTRR-based MPC and safe-steering frameworks. Specifically, an algorithm is presented to generate RTRRs using the data-based model for their subsequent use within a RTRR-based MPC design. Then, simulation results (fault-free and faulty) of a fed-batch reactor process subject to the data-based RTRR-based MPC design are presented. This is followed by the presentation of simulation results of a nylon-6,6 batch polymerization process wherein the control objective is trajectory tracking and the data-based modelling technique is used to develop the models used in a generic trajectory tracking controller. Finally, we summarize our results.

3.2 PRELIMINARIES

In this section, we first describe the class of batch processes considered. Then, we give an overview of auto-regressive exogenous (ARX) modelling, a popular technique for developing linear input-output models, and then illustrate how latent variable regression tools, such as partial least squares (PLS) regression, can be used within the ARX modelling framework to utilize all available measurements (beyond those designated as the inputs and outputs). Then, we review fuzzy c-means clustering, a key concept used in the data-based modelling framework. Finally, we review latent variable MPC (LV-MPC), which was originally developed in [26] and is later used as a point of comparison in our simulation studies.

3.2.1 *Process Description*

We consider batch processes described by the model in Equation (1.1), which is restated below for convenience.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \\ \boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) + \boldsymbol{v} \\ t &\in [t_0, t_{\text{end}}] \end{aligned} \tag{3.1}$$

Unlike in Chapter 2, no additional assumptions are made regarding the continuity of ODEs.

3.2.2 *Auto-regressive Exogenous Models*

In auto-regressive exogenous (ARX) modelling, the outputs at a specific sampling instance depend linearly on the previous process conditions defined by the process outputs and inputs. Mathematically, the ARX model prediction for output i, $\hat{y}_i$ is defined as:

$$\hat{y}_i[k] = \sum_{j=1}^{n_y} \boldsymbol{\alpha}_j' \boldsymbol{y}[k-j] + \sum_{j=1}^{n_u} \boldsymbol{\beta}_j' \boldsymbol{u}[k-j] + \gamma \tag{3.2}$$

where $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{n_y}\}$ and $\{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{n_u}\}$ are model coefficient vectors, γ is a bias term, which is important if the model is not identified around a steady-state, and n_y and n_u denote the number of lags in each output and input variable (respectively) and define the model order. For notation simplicity, Equation (3.2) is shown with the same number of lags in each output and input variable (i.e., all output and input variables are lagged n_y and n_u times, respectively).

To facilitate the estimation of the ARX model coefficients, Equation (3.2) can be rewritten in vector form as follows:

$$\hat{y}_i[k] = \boldsymbol{\theta}' \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} \tag{3.3}$$

where

$$\boldsymbol{\theta} := \begin{bmatrix} \boldsymbol{\alpha}_1' & \cdots & \boldsymbol{\alpha}_{n_y}' & \boldsymbol{\beta}_1' & \cdots & \boldsymbol{\beta}_{n_u}' & \gamma \end{bmatrix}'$$

is a vector of model coefficients and

$$\bar{\boldsymbol{x}}[k] := \begin{bmatrix} \boldsymbol{y}'[k-1] & \cdots & \boldsymbol{y}'[k-n_y] & \boldsymbol{u}'[k-1] & \cdots & \boldsymbol{u}'[k-n_u] \end{bmatrix}'$$

is a vector of lagged concatenated outputs and inputs. Given plant data, a response vector, $\boldsymbol{y}_\mathrm{p}$, and regressor matrix, $\mathbf{X}_\mathrm{p}$, can be constructed with columns corresponding to $y_i[k]$ and $\begin{bmatrix} \bar{\boldsymbol{x}}'[k] & 1 \end{bmatrix}$ (respectively) by sorting the data sample-wise. The model coefficients can be subsequently estimated using ordinary least squares (OLS) as shown below.

$$\hat{\boldsymbol{\theta}}' = \left(\mathbf{X}_\mathrm{p}'\mathbf{X}_\mathrm{p}\right)^{-1}\mathbf{X}_\mathrm{p}'\boldsymbol{y}_\mathrm{p}$$

However, for cases when the process data is highly correlated and/or co-linear, the co-variance matrix, $\mathbf{X}_\mathrm{p}'\mathbf{X}_\mathrm{p}$, will be nearly singular (rank deficient), leading to imprecise model coefficient estimates with large variances. This problem is particularly pertinent in batch data since many of the columns in $\mathbf{X}_\mathrm{p}$ can be auto correlated (correlated with each other at the same sampling instance) because they describe the same underlying phenomena in the process. Moreover, the process variables can also be cross correlated (correlated with each other and other variables at different sampling instances) when the data is collected under closed-loop conditions (e.g., see [27–30] for explicit system identification methods for closed-loop data), which introduces relationships between the process outputs and previous inputs[2]. The

[2] While closed-loop data can cause numerical problems, unlike in continuous processes, there can be sufficient information in such data to identify acceptable input-output models. This is because in batch processes, set-points vary over a wide range of operating conditions throughout the batch and the inputs are adjusted in response, keeping the process persistently excited.

sensitivity of the variance to correlated and co-linear data in OLS regression can be addressed using latent variable regression techniques such as partial least squares (PLS) regression.

3.2.3 Partial Least Squares Regression

When using partial least squares (PLS) regression in the context of ARX modelling, the coefficients for all p outputs are estimated simultaneously. This calls for constructing a p column response matrix, $\mathbf{Y}_\mathrm{p}$, in which the columns correspond to the outputs sorted sample-wise. Given the regressor and response matrices, in PLS modelling, the variables (columns) in $\mathbf{X}_\mathrm{p}$ and $\mathbf{Y}_\mathrm{p}$ are projected onto orthogonal subspaces of A-pairs of latent variables. Each pair of latent variables accounts for a certain percentage of the variance in the regressor and response matrices. Mathematically, PLS regression consists of decomposing $\mathbf{X}_\mathrm{p}$ and $\mathbf{Y}_\mathrm{p}$ as the sum of the outer products of a score and loading vector as follows:

$$\mathbf{X}_\mathrm{p} = \sum_{j=1}^{A} \boldsymbol{t}_j \boldsymbol{p}_j' + \mathbf{E}_x = \mathbf{T}\mathbf{P}' + \mathbf{E}_x \tag{3.4a}$$

$$\mathbf{Y}_\mathrm{p} = \sum_{j=1}^{A} \boldsymbol{r}_j \boldsymbol{q}_j' + \mathbf{E}_y = \mathbf{R}\mathbf{Q}' + \mathbf{E}_y \tag{3.4b}$$

where $\boldsymbol{t}_j$ and $\boldsymbol{r}_j$ are the input and output scores representing the projections of the variables in $\mathbf{X}_\mathrm{p}$ and $\mathbf{Y}_\mathrm{p}$ on their subspaces, $\boldsymbol{p}_j$ and $\boldsymbol{q}_j$ define the orientation of the corresponding subspaces, the matrices, $\mathbf{T}$, $\mathbf{P}$, $\mathbf{R}$, and $\mathbf{Q}$, contain their corresponding vectors, and $\mathbf{E}_{(\cdot)}$ denotes residual matrices. The noise reduction property of PLS regression stems from the idea that the lesser principal components are typically a consequence of measurement and process noise and therefore can be discarded during the regression.

Because it is desired to obtain a useful relationship between the original data matrices, $\mathbf{X}_\mathrm{p}$ and $\mathbf{Y}_\mathrm{p}$, the two matrices are linked by an inner relation between their scores of the form:

$$\boldsymbol{r}_j = \boldsymbol{b}_j \boldsymbol{t}_j + \boldsymbol{e}_j, \quad \text{for } j \in [1, A]$$

where $\boldsymbol{b}_j$ are the coefficients and $\boldsymbol{e}_j$ are the residuals of the inner relationship. In naive PLS algorithms, PLS is performed as follows. The 2 matrices are decomposed using principal component analysis (PCA) and then the inner relationship coefficients are computed using linear regression. The flaw with this approach is that because the orthogonal subspaces for both matrices are computed independently, the inner relationship can be weak. Thus, in common PLS algorithms, such as nonlinear iterative partial least squares (NIPALS), the subspace orientation and scores for both matrices are determined *simultaneously* so as to maximize the correlation between $\mathbf{X}_\mathrm{p}$ and $\mathbf{Y}_\mathrm{p}$ and therefore obtain the optimal fit for the inner relationship. The properties and steps of the NIPALS algorithms can be found in [14] with the rigorous mathematical details available in [31]. The final result from PLS regression

is a linear model between $\mathbf{X}_\mathrm{p}$ and $\mathbf{Y}_\mathrm{p}$ where the coefficients are functions of the scores and loadings from the matrix decompositions.

Remark 3.1: When there are only a few output variables, and they are known to be fairly uncorrelated, a viable alternative to PLS regression is principal component regression (PCR). In PCR,

1. PCA is performed on the regressor matrix, $\mathbf{X}_\mathrm{p}$, yielding $\mathbf{T}$ and $\mathbf{P}$.
2. OLS regression is subsequently performed between $\mathbf{T}$ and each output variable, yielding a linear model for each output.

Thus, the variables in the regressor matrix in OLS regression are essentially replaced by new ones (the scores) with better properties (orthogonality) that also span the original space. The orthogonality property improves the numerical properties of the required inversion during OLS regression. Additionally, by leaving out the unimportant principal components in the PCA step, PCR retains the noise reduction properties of PLS and other latent variable modelling methods.

Remark 3.2: Although ARX PLS/PCR models are capable of describing high order linear systems, they can still be poor representations of inherently nonlinear processes, owing to the assumption of linearity between the latent variables. Additionally, the lesser latent variables in a PLS/PCR model for highly nonlinear systems can contain important information about the nonlinearities and therefore cannot be discarded. In response to this, PLS algorithms have been expanded to incorporate nonlinearities by modifying the inner relationship between the scores while retaining the useful statistical properties of the linear PLS modelling approach [17–19]. The ability of these approaches to capture the nonlinear behaviour is, however, contingent on the appropriate choice of the nonlinear mapping.

3.2.4 Fuzzy c-Means Clustering

An important step in the proposed multi-model approach in Section 3.3 is to locate operating points around which individual local linear models are identified. One approach to find this set of operating points is to partition the historical batch database into a number of clusters (i.e., a group of points in the database that are mathematically similar). Subsequently, a corresponding linear model can be identified for each cluster. For the current work, we employ **fuzzy** c**-means** clustering (see for [32] a review) to partition the database.

Let

$$\bar{\mathbf{X}} := \begin{bmatrix} \bar{\boldsymbol{x}}_1 & \cdots & \bar{\boldsymbol{x}}_i & \cdots & \bar{\boldsymbol{x}}_{N_\mathrm{obs}} \end{bmatrix}$$

be a $[(p \times n_y) + (m \times n_u)] \times N_{\text{obs}}$ matrix where each column is a different instance of concatenated lagged outputs and inputs from the database and N_{obs} is the number of lagged vectors that can be constructed from a given database. The lagged output-input space in $\bar{\mathbf{X}}$ can be partitioned into L different clusters using fuzzy clustering, which assigns each sample, $\bar{\boldsymbol{x}}_i$, a degree of belonging to a cluster $\ell \in [1, L]$. The partition information can be represented by a membership matrix, $\mathbf{U} = \{\mu_{\ell,i}\} \in \mathbb{R}^{L \times N_{\text{obs}}}$, where each row contains the membership information for the ℓ-th cluster for all N points. In fuzzy clustering, the elements in $\mathbf{U}$ must satisfy the following conditions [33]:

$$\mu_{\ell,i} \in [0,1], \quad \text{for } \ell \in [1, L], i \in [1, N_{\text{obs}}] \tag{3.5a}$$

$$\sum_{\ell=1}^{L} \mu_{\ell,i} = 1, \quad \text{for } i \in [1, N_{\text{obs}}] \tag{3.5b}$$

$$0 < \sum_{i=1}^{N_{\text{obs}}} \mu_{\ell,i} < N_{\text{obs}}, \quad \text{for } \ell \in [1, L] \tag{3.5c}$$

Equation (3.5b) requires that the total membership of each observation, which ranges from 0 to 1 (Equation (3.5a)), equals 1. The majority of fuzzy clustering algorithms is based on minimizing the total variance in the data from cluster centres. Mathematically, this idea is expressed by minimizing the following (nonlinear) objective function [32, 34] (the so-called c-means functional):

$$J_{\text{FCM}} = \sum_{i=1}^{N_{\text{obs}}} \sum_{\ell=1}^{L} \mu_{\ell,i}^{f} D_{\ell,i}^{2} \tag{3.6}$$

where $D_{\ell,i} := \|\bar{\boldsymbol{x}}_i - \boldsymbol{c}_\ell\|$ denotes the Euclidean distance between point i and the ℓ-th cluster centre and $\boldsymbol{c}_\ell \in \mathbb{R}^{[(p \times n_y)+(m \times n_u)] \times 1}$ denotes the ℓ-th cluster's centre, which has to be determined for $\ell \in [1, L]$. The weighting exponent parameter, f, determines the fuzziness of the clusters with $f = 1$ implying hard, non-overlapping partitions. For this work, (as is typically the case) we choose $f = 2$.

The partition matrix elements, $\mu_{\ell,i}$, and cluster centres, $\boldsymbol{c}_\ell$, that minimize Equation (3.6) and satisfy the constraints in Equations (3.5a) to (3.5c) have been shown to be (for $f > 1$) [32, 34]:

$$\mu_{\ell,i} = \frac{1}{\sum_{j=1}^{L} (D_{\ell,i}/D_{j,i})^{2/(f-1)}} \tag{3.7}$$

and

$$\boldsymbol{c}_\ell = \frac{\sum_{i=1}^{N_{\text{obs}}} \mu_{i,\ell}^{f} \bar{\boldsymbol{x}}_i}{\sum_{i=1}^{N_{\text{obs}}} \mu_{i,\ell}^{f}} \tag{3.8}$$

From Equation (3.8), it can be seen that the centre point of each cluster is also the mean of all the points, weighted by their membership degrees. Note that Equations (3.7) to (3.8) constitute a set of nonlinear equations, which can be solved using successive iterations [35], with the iterations terminated when changes in the membership matrix between two iterations become smaller than some predefined tolerance. As this is a nonlinear set of equations, this procedure can possibly terminate at a point that is not a true solution; therefore, the procedure is usually repeated numerous times starting from different initial memberships, and the results are selected for the instance that yields the minimum objective function value in Equation (3.6). The steps of the iterative algorithm are shown in Algorithm 3.1.

Algorithm 3.1 FUZZY c-MEANS CLUSTERING

Require: $\bar{\mathbf{X}}$, ε (termination tolerance), L, and f

$j \leftarrow 0$

Initialize membership function matrix randomly to $\mathbf{U}^{(j)}$

repeat

$j \leftarrow j+1$

Compute the cluster centres:

$$\boldsymbol{c}_\ell^{(j)} = \frac{\sum_{i=1}^{N_{\text{obs}}} \left(\mu_{i,\ell}^{(j-1)}\right)^f \bar{\boldsymbol{x}}_i}{\sum_{i=1}^{N_{\text{obs}}} \left(\mu_{i,\ell}^{(j-1)}\right)^f}$$

Compute the Euclidean distances:

$$D_{\ell,i}^2 = \|\bar{\boldsymbol{x}}_i - \boldsymbol{c}_\ell^{(j)}\|^2, \quad \text{for } \ell \in [1, L], i \in [1, N_{\text{obs}}]$$

Update the membership function matrix:

$$\mu_{\ell,i}^{(j)} = \frac{1}{\sum_{k=1}^{L} (D_{\ell,i}/D_{k,i})^{2/(f-1)}}, \quad \text{for } \ell \in [1, L], i \in [1, N_{\text{obs}}]$$

until $\|\mathbf{U}^{(j)} - \mathbf{U}^{(j-1)}\| < \varepsilon$

return $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_L$

For the case of $f = 2$, Equation (3.7) reduces to the following form:

$$\mu_{\ell,i} = \frac{\|\bar{\boldsymbol{x}}_i - \boldsymbol{c}_\ell\|^{-2}}{\sum_{\ell=1}^{L} \|\bar{\boldsymbol{x}}_i - \boldsymbol{c}_\ell\|^{-2}} \tag{3.9}$$

In view of this, in fuzzy clustering, the degree of $\bar{\boldsymbol{x}}_i$ belonging to cluster ℓ is essentially taken to be inversely proportional to the squared distance between the point and cluster centre, $\boldsymbol{c}_\ell$ (i.e., $\mu_{\ell,i} \propto \|\bar{\boldsymbol{x}}_i - \boldsymbol{c}_\ell\|^{-2}$), which is then normalized across all clusters.

In the above discussion on fuzzy c-means clustering, points that are mathematically "similar" according to the Euclidean 2-norm are clustered, resulting in (overlapping) *spherical* clusters. One way to influence the cluster shapes is to use a norm-inducing matrix, $\mathbf{A}_{\text{FCM}}$, in computing $D^2_{\ell,i}$ as shown below.

$$D^2_{\ell,i} = \|\bar{\boldsymbol{x}}_i - \boldsymbol{c}_\ell\|^2_{\mathbf{A}_{\text{FCM}}}$$

In the discussion above, $\mathbf{A}_{\text{FCM}} = \mathbf{I}$, which gives the standard Euclidean norm. In the simulation examples, to account for the different variances (i.e., from having different units) in the directions of the coordinate axes of $\bar{\mathbf{X}}$, $\mathbf{A}_{\text{FCM}}$ was specified to be a diagonal matrix comprised of the inverse variance of each variable as its elements:

$$\mathbf{A}_{\text{FCM}} = \text{diag}\left\{(1/\sigma_i^2)\right\}, \quad i \in \left[1, \left[(p \times n_y) + (m \times n_u)\right]\right]$$

where σ_i denotes the standard deviation of variable i. While the Euclidean norm induces spherical clusters, this choice for $\mathbf{A}_{\text{FCM}}$ generated ellipsoidal clusters with the axes of the ellipsoids parallel to the coordinate axes (i.e., it induced a diagonal norm on $\mathbb{R}^{[(p\times n_y)+(m\times n_u)]}$). To accommodate other types of non-spherical clusters, extensions of fuzzy c-means clustering that consider different weighted norms (i.e., the Mahalanobis norm) or different shapes (e.g., see [36, 37]) can be utilized.

The number of clusters is another essential parameter in fuzzy c-means clustering. Well-defined criteria (based on the cluster geometry) to iteratively refine the number of clusters have been presented in [38–40]. To evaluate the goodness of the final fuzzy partitions, many validation measures have also been introduced (e.g., see [40, 41]) with the most popular being the Xie-Beni index [41], which is a ratio of the total within-cluster variance to the separation of the cluster centres (and therefore should be minimal for the best partition). In this work, as described in Section 3.3, we iteratively refined the number of clusters based on how well an independent validation data set was predicted. Thus, there was a balancing of the number of clusters and prediction error from the final model.

Remark 3.3: From the definition of $\bar{\mathbf{X}}$ in this section, the dimension of the space required to be clustered is $(p \times n_y) + (m \times n_u)$, which can be prohibitively high. The dimensionality problem was addressed in this work by first projecting the variables in $\bar{\mathbf{X}}$ onto a lower dimensional subspace or latent variable space using PCA and subsequently clustering the resulting latent variable or score space. The resulting loading matrix from PCA, $\mathbf{P}$, can be used to relate the original cluster space variables to the latent variables according to: $\mathbf{T} = \bar{\mathbf{X}}\mathbf{P}$ where $\mathbf{T}$ denotes the projections of each row in $\bar{\mathbf{X}}$ onto the subspace (i.e., the scores). Typically, a much lower number of principal components (compared to $(p \times n_y) + (m \times n_u)$) is required to completely characterize $\bar{\mathbf{X}}$ since $\bar{\mathbf{X}}$ can include many lagged variables and

therefore correlations among the columns. In short, PCA is a natural choice to sift through extraneous data and identify the core dimensionality of the dynamics prior to clustering.

3.2.5 Latent Variable Model Predictive Control

In [26], a modelling and control approach is presented for tracking batch process variables called latent variable MPC (LV-MPC). In this section, we give a brief overview of the LV-MPC modelling procedure and show how it can be used to formulate a trajectory tracking predictive controller. Note that in this work, we consider a simple version of LV-MPC as an example of a prominent batch modelling and control approach only to benchmark our proposed approach.

In the LV-MPC modelling approach, lag and lead parameters for a dynamic PCA model are the key user inputs. Suppose a lag and lead of M and P sampling instances (respectively) are chosen. All the information from batch b (of B total batches) is collected in the matrix $\mathbf{X}_b$ in which each row corresponds to a specific sampling instance, denoted generally by k, and is comprised of the following vector:

$$\mathbf{X}_b[k] = \begin{bmatrix} \boldsymbol{x}_\mathrm{p}[k] & \boldsymbol{x}_\mathrm{f}[k] \end{bmatrix}$$

where:

$$\boldsymbol{x}_\mathrm{p}[k] = \begin{bmatrix} \boldsymbol{y}'[k-M] & \cdots & \boldsymbol{y}'[k] & \boldsymbol{y}'_\mathrm{ref}[k-M] & \cdots & \boldsymbol{y}'_\mathrm{ref}[k+P] & \boldsymbol{u}'[k-M] & \cdots & \boldsymbol{u}'[k-1] \end{bmatrix}$$
$$\boldsymbol{x}_\mathrm{f}[k] = \begin{bmatrix} \boldsymbol{y}'[k+1] & \cdots & \boldsymbol{y}'[k+P] & \boldsymbol{u}'[k] & \cdots & \boldsymbol{u}'[k+P] \end{bmatrix}$$

where $\boldsymbol{y}_\mathrm{ref}[k]$, $\boldsymbol{y}[k]$, and $\boldsymbol{u}[k]$ denote the output reference, output, and input vectors at sampling instance k. Thus, each row in $\mathbf{X}_b$ is partitioned into a vector of known and unknown variables at k, denoted by $\boldsymbol{x}_\mathrm{p}[k]$ and $\boldsymbol{x}_\mathrm{f}[k]$ (respectively). Note that the vector of known variables includes the *future* reference trajectories since the reference trajectories for the batch duration are known before the batch run begins.

The matrices for all previous batches are then stacked vertically in $\boldsymbol{\Omega}$ as shown below.

$$\boldsymbol{\Omega} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_B \end{bmatrix}$$

Next, $\boldsymbol{\Omega}$ is mean centred and scaled to unit variance and then decomposed using PCA, yielding the following relationships:

$$\hat{\boldsymbol{\Omega}} = \mathbf{T}\mathbf{P}'$$
$$\mathbf{T} = \hat{\boldsymbol{\Omega}}\mathbf{P}$$

where $\hat{\boldsymbol{\Omega}}$ is the model prediction of $\boldsymbol{\Omega}$, and $\mathbf{T}$ and $\mathbf{P}$ denote the score and loading matrices.

Based on all the known information at any point during the batch, the PCA model extracted from $\boldsymbol{\Omega}$ can be used for simultaneously computing the current control action and the future input and output behaviour (up to P time steps). To this end, the loading matrix is partitioned into corresponding blocks as follows:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_\mathrm{p}' & \mathbf{P}_\mathrm{f}' \end{bmatrix}'$$

where $\mathbf{P}_\mathrm{p}$ and $\mathbf{P}_\mathrm{f}$ correspond to $\boldsymbol{x}_\mathrm{p}$ and $\boldsymbol{x}_\mathrm{f}$, respectively. For a new batch to maintain the same correlation structure as the historical database at k (that is, for the new batch to remain statistically consistent with previous batches), we require:

$$\boldsymbol{x}_\mathrm{f}[k] = \hat{\boldsymbol{t}}'[k]\mathbf{P}_\mathrm{f}'$$

where $\hat{\boldsymbol{t}}[k]$ is the score vector of the new batch. One way to compute the unknown $\boldsymbol{x}_\mathrm{f}[k]$ vector is to treat it as missing data and employ the PCA missing data estimation algorithm [42]. According to this algorithm, we have (for details, refer to [26, 42]):

$$\boldsymbol{x}_\mathrm{f}[k] = \boldsymbol{x}_\mathrm{p}[k]\mathbf{P}_\mathrm{p}\left(\mathbf{P}_\mathrm{p}'\mathbf{P}_\mathrm{p}\right)^{-1}\mathbf{P}_\mathrm{f}' \tag{3.10}$$

Since $\boldsymbol{x}_\mathrm{f}[k]$ includes the current control action, Equation (3.10) represents the closed form LV-MPC control law. As in PID control, input constraints can be imposed by "clipping" the prescribed inputs appropriately.

To explicitly account for input constraints or to incorporate a move suppression factor, the inputs can be alternatively computed by solving the following optimization problem.

$$\min_{\hat{\boldsymbol{t}}[k]} J_\mathrm{LV} = \left\|\hat{\boldsymbol{t}}'[k] - \left[\boldsymbol{x}_\mathrm{p}[k]\mathbf{P}_\mathrm{p} + \boldsymbol{x}_\mathrm{f}[k]\mathbf{P}_\mathrm{f}\right]\right\|_{\boldsymbol{\Xi}}^2 + \left\|\boldsymbol{u}[k] - \boldsymbol{u}[k-1]\right\|_{\boldsymbol{\Pi}}^2 \tag{3.11a}$$

$$\text{subject to: } \boldsymbol{x}_\mathrm{f}[k] = \hat{\boldsymbol{t}}'[k]\mathbf{P}_\mathrm{f}' \tag{3.11b}$$

$$\boldsymbol{u}_\mathrm{min}' \leq \hat{\boldsymbol{t}}'[k]\mathbf{P}_\mathrm{c}' \leq \boldsymbol{u}_\mathrm{max}' \tag{3.11c}$$

The decision variables in this problem are the elements of the desired score vector, $\hat{\boldsymbol{t}}[k]$, which are related to the future outputs and inputs by Equation (3.11b). In Equation (3.11c), the matrix $\mathbf{P}_\mathrm{c}$ refers to the elements of $\mathbf{P}_\mathrm{f}$ which correspond specifically to the inputs. This

constraint ensures the computed inputs are maintained between $\boldsymbol{u}_{\min}$ and $\boldsymbol{u}_{\max}$. The first term in Equation (3.11a) attempts to minimize the distance between the desired score vector and the score vector obtained by projecting all the known and predicted variables of the current batch (i.e., $\boldsymbol{x}_{\mathrm{p}}$ and $\boldsymbol{x}_{\mathrm{f}}$) onto the latent variable subspace. When $\boldsymbol{\Xi} = \mathbf{I}$, this distance is commonly referred to as the "squared prediction error" (SPE).

For the closed form LV-MPC control law in Equation (3.10), the SPE is forced to be zero, implying the implicitly computed score vector lies exactly on the latent variable subspace. In contrast, when computing $\hat{\boldsymbol{t}}[k]$ using the optimization problem, there is flexibility for the score vector to lie off the model plane such that the input constraints are satisfied and large successive input changes are penalized (according to the second term in Equation (3.11a)). In using this optimization problem or Equation (3.10), note that the inputs are not explicitly "optimized" to meet the desired set-point trajectories. Instead, the set-point information is implicitly incorporated in the model, and the inputs are computed using the PCA missing data estimation algorithm.

Remark 3.4: In addition to M and P, the number principal components retained in the PCA model is another important user input. In our case, the number of principal components was specified such that 99% of the variation in $\boldsymbol{\Omega}$ was explained by the model. The number of lags and leads were selected based on the "reconstruction error" in the input predictions as explained in [26]. Specifically, for a given number of validation batches, the PCA model was used to predict the input moves at each time step (after M time steps) using Equation (3.10). The number of lags and leads were chosen such that they minimized the average sum of absolute errors (over the batch duration) between the predicted and database (i.e., actual) inputs.

Remark 3.5: During the first M sampling instances of a new batch, the complete past information data vector, $\boldsymbol{x}_{\mathrm{f}}$, which is needed for solving Equation (3.10), is unavailable. In this work, during this time period, the inputs prescribed by a tightly tuned PI controller was implemented on the process. Another alternative is to simply implement the nominal inputs during this period. A similar missing information problem occurs around batch termination. Specifically, during the end of the batch, future $\boldsymbol{y}_{\mathrm{ref}}[k]$ vectors are required up to P sampling instances to complete the $\boldsymbol{x}_{\mathrm{p}}$ vector. For $P > 1$, this calls for set-point information beyond the batch termination time. One solution for this case (as suggested in [26]) is to simply assume the last elements in the original $\boldsymbol{y}_{\mathrm{ref}}[k]$ vectors hold after the batch duration. This amounts to assuming the unknown output reference trajectories remain consistent with the last known reference trajectory trends.

Remark 3.6: The data arrangement in the LV-MPC modelling approach is commonly referred to as the "variable-wise unfolding" of batch data. The end result of this approach is a single, "average" PCA model of the batch, implying just 1 correlation structure of the batch

process variables for the batch duration. In other words, there is an inherent assumption associated with “variable-wise" unfolding of batch data of a *constant* correlation structure for the batch duration. However, this assumption often does not hold for batches that proceed in distinct multiple phases during which the correlations among the process variables can change substantially. In the proposed multi-model approach, by clustering the batch database appropriately prior to the model fitting, we essentially capture the different phases (if any) of the process and identify their corresponding models.

3.3 DATA-BASED MODEL DEVELOPMENT

In this section, we present the multi-model approach and explain the model development process. Assuming a database of previous batches exists, the main identification steps involve:

1. Clustering the $\bar{\mathbf{X}}$ space (or the score space after decomposing $\bar{\mathbf{X}}$ using PCA) of the database using fuzzy c-means clustering
2. Using linear regression (i.e., OLS regression, PLS regression, or PCR) to identify ARX models around the cluster centre points

In the final model form, the local linear models are combined with weights to describe the global nonlinear dynamics. Mathematically, this idea is expressed as follows:

$$\hat{y}_i[k] = \sum_{\ell=1}^{L} \omega_\ell[k] \left(\sum_{j=1}^{n_y} \boldsymbol{\alpha}'_{\ell,j} \boldsymbol{y}[k-j] + \sum_{j=1}^{n_u} \boldsymbol{\beta}'_{\ell,j} \boldsymbol{u}[k-j] + \gamma_\ell \right) \tag{3.12a}$$

$$= \sum_{\ell=1}^{L} \omega_\ell[k] \boldsymbol{\theta}'_\ell \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} \tag{3.12b}$$

where $\bar{\boldsymbol{x}}[k]$ denotes a vector of lagged concatenated outputs and inputs (as before), $\omega_\ell[k]$ is the weight given to model ℓ of the L total models, and $\boldsymbol{\alpha}_{\ell,j}$, $\boldsymbol{\beta}_{\ell,j}$, and γ_ℓ define the ℓ-th ARX model. The vector, $\boldsymbol{\theta}_\ell$, stores ℓ-th model's coefficients. Using the following definitions,

$$\boldsymbol{\Theta} := \begin{bmatrix} \boldsymbol{\theta}'_1 & \cdots & \boldsymbol{\theta}'_\ell & \cdots & \boldsymbol{\theta}'_L \end{bmatrix}' \tag{3.13}$$

$$\boldsymbol{\psi}[k] := \begin{bmatrix} \omega_1[k] \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} & \cdots & \omega_\ell[k] \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} & \cdots & \omega_L[k] \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} \end{bmatrix}' \tag{3.14}$$

Equation (3.12a) can be rewritten in the common least squares vector form:

$$\hat{y}_i[k] = \boldsymbol{\psi}'[k] \boldsymbol{\Theta} \tag{3.15}$$

If the weights corresponding to the training data are computed independently from the model coefficients, Equation (3.12a) is linear with respect to the model coefficients, reducing the system identification problem to a regression problem. Since all the local models can potentially contribute during prediction, it is important to identify them simultaneously. To facilitate this regression, a regressor matrix with columns corresponding to $\boldsymbol{\psi}[k]$ and a response vector corresponding to $y_i[k]$ are constructed, denoted by $\boldsymbol{\Psi}$ and $\boldsymbol{y}_{\mathrm{p}}$, respectively. Next, a linear regression technique is used to estimate the coefficients, $\boldsymbol{\Theta}'$. Note that when it is desired to estimate models for *multiple* outputs simultaneously (i.e., using PLS regression), a response matrix (instead of a vector), $\mathbf{Y}_{\mathrm{p}}$, has to be constructed with the columns corresponding to the different outputs, and $\boldsymbol{\Theta}$ is a matrix as opposed to a vector.

With full state measurements, $n_y = n_u = 1$ is a natural choice for all the outputs (states) and inputs in the ARX model in Equation (3.2), reducing it to the state-space model described by:

$$\hat{\boldsymbol{x}}[k] = \boldsymbol{\alpha}\boldsymbol{x}[k-1] + \boldsymbol{\beta}\boldsymbol{u}[k-1] + \boldsymbol{\gamma} \tag{3.16a}$$

$$= \begin{bmatrix} \boldsymbol{\alpha} & \boldsymbol{\beta} & \boldsymbol{\gamma} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}[k-1] \\ \boldsymbol{u}[k-1] \\ 1 \end{bmatrix} \tag{3.16b}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{n\times n}$, $\boldsymbol{\beta} \in \mathbb{R}^{n\times m}$, and $\boldsymbol{\gamma} \in \mathbb{R}^{n\times 1}$ define the ARX model. Accordingly, for the multi-model approach with full state measurements, the matrix to be clustered, $\bar{\mathbf{X}}$, consists of state and input measurements; that is, $\bar{\boldsymbol{x}}[k] = \begin{bmatrix} \boldsymbol{x}'[k-1] & \boldsymbol{u}'[k-1] \end{bmatrix}'$ and the final model takes the form shown below.

$$\hat{\boldsymbol{x}}[k] = \sum_{\ell=1}^{L} \omega_\ell[k] \begin{bmatrix} \boldsymbol{\alpha}_\ell & \boldsymbol{\beta}_\ell & \boldsymbol{\gamma}_\ell \end{bmatrix} \begin{bmatrix} \boldsymbol{x}[k-1] \\ \boldsymbol{u}[k-1] \\ 1 \end{bmatrix} \tag{3.17}$$

Intuitively, from the process description in Equation (3.1), the weights placed on the local linear models should depend on the current value of the states and inputs since they define the process dynamics. In other words, the local models should be weighted according to the current process conditions. In the absence of state measurements, a combination of lagged outputs and inputs can be used to infer the current process conditions. In this work, to determine the weights for the training data, the normalized fuzzy clustering membership function in Equation (3.9) is used; thus, for data point i in the training data, we have $\omega_\ell[i] = \mu_{\ell,i}$, and in general, we have:

$$\omega_\ell[k] = \frac{\|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{\sum_{\ell=1}^{L} \|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}} \tag{3.18}$$

Because the membership function quantifies the degree to which a lagged output-input vector belongs to each cluster, it is also indicative of which local models should be given more weight than the others. For instance, if a lagged output-input vector nearly coincides with a specific cluster's centre point, the local linear model corresponding to that cluster should be given most of the weight. This is consistent with Equation (3.18) as the membership function value corresponding to that cluster will be close to 1 while for the remaining clusters, the membership function value will be near 0.

The key model parameters that have to be specified in this modelling approach are the lags in the output and inputs variables, n_y and n_u, and the number of clusters for L. In this work, we iterated over different combinations of these parameters and selected the combination which minimized the root mean squared prediction error (RMSE) when predicting back an *independent* validation data set. The RMSE in the i-th output was defined to be:

$$\text{RMSE}_i \triangleq \sqrt{\frac{1}{B}\frac{1}{K}\sum_{b=1}^{B}\sum_{k=1}^{K}\left[\hat{y}_i^{(b)}[k] - y_i^{(b)}[k]\right]^2} \tag{3.19}$$

where b indexes the batch number, and B and K are the number of validation batches and sampling instances in each batch (respectively). When PLS regression was used to estimate the model parameters, an additional loop for determining the "optimum" number of principal components to retain was also included. Generally, retaining a high number of principal components will result in a very low residuals for the training data, but the predictive capabilities of the model will be reduced due to over-fitting. This is because with an excessive number of principal components, the model begins to fit the random noise element in the data.

The iterative procedure for balancing the number of model parameters with the prediction error is shown below as Algorithm 3.2. The necessary user inputs are summarized in Table 3.1.

Table 3.1: Input parameters for Algorithm 3.2

Input	Description
$n_{y,\min}$	Minimum number of lags in the outputs
$n_{y,\max}$	Maximum number of lags in the outputs
$n_{u,\min}$	Minimum number of lags in the inputs
$n_{u,\max}$	Maximum number of lags in the inputs
$L_{\min}$	Minimum number of clusters
$L_{\max}$	Maximum number of clusters
PLS flag	Flag indicating if PLS regression will be used - ("yes" or "no")

Algorithm 3.2 COMPUTATION OF THE MODEL PARAMETERS

Require: $n_{y,\min}$, $n_{y,\max}$, $n_{u,\min}$, $n_{u,\max}$, $L_{\min}$, $L_{\max}$, and PLS flag
  for $i = n_{y,\min}$ to $n_{y,\max}$ **do**
    for $j = n_{u,\min}$ to $n_{u,\max}$ **do**
      for $\ell = L_{\min}$ to $L_{\max}$ **do**
        Construct $\boldsymbol{\Psi}$ with i and j lags and ℓ local linear models (see Equation (3.14))
        if PLS Flag = "yes" **then**
          Construct $\mathbf{Y}_{\mathrm{p}}$
          $N_{\mathrm{pc,max}} \leftarrow$ Number of columns in $\boldsymbol{\Psi}$
          for $n = 1$ to $N_{\mathrm{pc,max}}$ **do**
            $\boldsymbol{\Theta}_{i,j,n,\ell} \leftarrow$ PLS Regression with: $\left(\mathbf{Y}_{\mathrm{p}}, \boldsymbol{\Psi}, n\right)$
            $\mathrm{RMSE}_{i,j,\ell,n} \leftarrow$ RMSE in validation data with $\boldsymbol{\Theta}_{i,j,\ell,n}$
          end for
        else if PLS Flag = "no" **then**
          Construct $\boldsymbol{y}_{\mathrm{p}}$
          $\boldsymbol{\Theta}_{i,j,\ell} \leftarrow$ OLS regression/PCR with: $\left(\boldsymbol{y}_{\mathrm{p}}, \boldsymbol{\Psi}\right)$
          $\mathrm{RMSE}_{i,j,\ell} \leftarrow$ RMSE in predicted validation data with $\boldsymbol{\Theta}_{i,j,\ell}$
        end if
      end for
    end for
  end for
  if PLS Flag = "yes" **then**
    return $\boldsymbol{\Theta}$ corresponding to minimum $\mathrm{RMSE}_{i,j,\ell,k}$
  else if PLS Flag = "no" **then**
    return $\boldsymbol{\Theta}$ corresponding to minimum $\mathrm{RMSE}_{i,j,\ell}$
  end if

Remark 3.7: A key difference between this modelling approach and the PWA framework in [20] is the clustering algorithm used to partition the training data. In PWA modelling, the clustering algorithm, k-means, induces artificial boundaries between the partitions (i.e., hard or crisp clusters) and only samples belonging to a specific partition can contribute in determining its corresponding model. In contrast, fuzzy c-means clustering permits adjacent clusters to overlap, and surrounding data around each cluster plays a role in determining the cluster's model. This becomes important for accurately modelling periods of transition in the process when it is evolving from one operating region to another (i.e., one cluster to another) or when an output-input combination is encountered that belongs to many clusters with varying degrees. PWA models also use a discrete weighting function wherein only one model from the bank of models is used for prediction. By comparison, in this multi-model approach, multiple models can simultaneously contribute in coming up with a prediction through the continuous weighting function, resulting in overall smoother predictions. The discrete model selection feature in the PWA framework also negatively impacts its use in any optimization-based control design by requiring the solution of an optimization problem

that includes continuous (the control action) as well as discrete (the choice of the model) variables (i.e., a mixed integer problem).

Remark 3.8: The clustering algorithm and weighting function are also the important differentiating features of the method described in this section from TSK and operating-regime-based modelling. Specifically, in the proposed approach, the entire model input space and possible correlations among the variables are considered during the clustering step and therefore in generating the weighting function. In TSK modelling, each model input variable is clustered separately and potential interactions/correlations among the variables are ignored. Moreover, there is no systematic way to choose the weighting function form in TSK modelling. The weighting function selection is also ambiguous in operating-regime-based modelling as it is derived from an understanding of the system mechanism and is therefore problem specific [15]. An appropriate set of measurable process variables with which to compute the weights also have to be first correctly identified. The proper identification of these variables may be difficult (if not impossible) for complex batch processes. Even after suitably identifying these variables, the weighting function form is essentially obtained through a trial and error process where several candidate forms are attempted.

3.4 EMPIRICAL REVERSE-TIME REACHABILITY REGION-BASED MODEL PREDICTIVE CONTROL

Reverse-time reachability regions (RTRRs) were used in Chapter 2 to design predictive controllers for batch processes with useful reachability and fault-tolerant characteristics. However, the currently available algorithm for generating RTRRs requires a first-principles process model, which, in many cases, may be unavailable. In this section, assuming full state measurements, we present a methodology to generate and characterize RTRRs using the data-based modelling approach developed in the previous section. Then, we formulate a MPC design that utilizes these characterizations.

3.4.1 Empirical Reverse-time Reachability Regions

Due to unavoidable discrepancies between a process and its empirical model, instead of considering exact reachability to a desired end-point, we consider reachability to a desired end-point neighbourhood (as before), $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. We define a data-based/empirical version of a RTRR as the set of states from where the data-based/empirical process model can be driven inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ by batch termination. The formal definition of an empirical RTRR is stated below.

Definition 3.1 (Empirical Discrete Reverse-time Reachability Region): *For the batch process described by Equation* (3.1) *with sampling period δ, which has been modelled using the form*

in Equation (3.17), the empirical reverse-time reachability region(RTRR) at time $t = t_{end} - z\delta$*, indexed by z, is the set:*

$$\hat{\mathcal{R}}_z = \{\boldsymbol{x}[0] \mid \boldsymbol{x}[k] = \sum_{\ell=1}^{L} \omega_\ell[k] \begin{bmatrix} \boldsymbol{\alpha}_\ell & \boldsymbol{\beta}_\ell & \boldsymbol{\gamma}_\ell \end{bmatrix} \begin{bmatrix} \boldsymbol{x}[k-1] \\ \boldsymbol{u}[k-1] \\ 1 \end{bmatrix} \text{ for } k = 1, \ldots, z$$
$$\exists\, \boldsymbol{u}[k] \in \mathcal{U} \text{ such that } \boldsymbol{x}(t_{end}) \in \mathcal{B}(\boldsymbol{x}_{des})\}$$

where $\boldsymbol{u}[k] = \boldsymbol{u}(k\delta)$ *which satisfies* $\boldsymbol{u}(t) = \boldsymbol{u}[k]\ \forall t \in [k\delta, (k+1)\delta)$.

Generating Empirical Reverse-time Reachability Regions

In formulating an empirical RTRR-based predictive controller, explicit characterizations of the RTRRs are required. In this work, as before, we choose ellipsoids to mathematically express empirical RTRR estimates as follows:

$$\hat{\mathcal{R}}_z \approx \{\boldsymbol{x} \mid \|\boldsymbol{x} - \hat{\boldsymbol{c}}_z\|_{\hat{\mathbf{P}}_z} \leq 1\} \tag{3.20}$$

where $\hat{\boldsymbol{c}}_z \in \mathbb{R}^{n\times 1}$ denotes the ellipsoid's centre point and the positive-definite, symmetric matrix $\hat{\mathbf{P}}_z \in \mathbb{R}^{n\times n}$ defines its size and orientation. Note that because $z = 0$ corresponds to t_{end}, $\hat{\boldsymbol{c}}_0 = \boldsymbol{x}_{\text{des}}$ and $\hat{\mathbf{P}}_0$ is a user defined matrix based on the acceptable variance level of the final product quality.

An equivalent representation of an ellipsoid was used in this work in which the ellipsoid is expressed as the image of an unit ball under an affine transformation. That is, consider the unit ball in $\mathbb{R}^n$:

$$S(0,1) := \left\{\boldsymbol{x} \mid \|\boldsymbol{x}\|^2 \leq 1\right\}$$

and the affine transformation:

$$T(\boldsymbol{x}) := \mathbf{H}\boldsymbol{x} + \boldsymbol{d}$$

where $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a positive-definite, symmetric rotation matrix and $\boldsymbol{d} \in \mathbb{R}^n$ is a translation vector. Applying the affine transformation to a point on the unit ball, we have:

$$\boldsymbol{z} = \mathbf{H}\boldsymbol{x} + \boldsymbol{d} \rightarrow \boldsymbol{x} = \mathbf{H}^{-1}(\boldsymbol{z} - \boldsymbol{d})$$

An ellipsoid can then be expressed through an affine transformation of the unit ball:

$$T(S(0,1)) = \left\{\boldsymbol{z} \parallel \|\mathbf{H}^{-1}(\boldsymbol{z} - \boldsymbol{d})\|^2 \leq 1\right\} = \left\{\boldsymbol{z} \parallel \|\boldsymbol{z} - \boldsymbol{d}\|^2_{\mathbf{V}^{-1}} \leq 1\right\} \tag{3.21}$$

where $\mathbf{V} = \mathbf{H}\mathbf{H}' \in \mathbb{R}^{n\times n}$ is a positive-definite, symmetric matrix. Thus, from Equation (3.21), defining $\mathbf{H}_z$ and $\boldsymbol{d}_z$ is equivalent to defining the ellipsoid parameters, $\hat{\mathbf{P}}_z$ and $\hat{\boldsymbol{c}}_z$, in Equation (3.20).

Starting at $z = 1$, an ellipsoidal estimate of $\hat{\mathcal{R}}_z$ is identified from where the model states can be driven inside the ellipsoidal estimate of $\hat{\mathcal{R}}_{z-1}$. This procedure is repeated until an empirical RTRR ellipsoid is identified at every sampling instance. Given the RTRR ellipsoid parameters at $z-1$ and N_{ub} (predetermined) points (generated from a uniform distribution) on the surface of an unit ball denoted by $\left\{\boldsymbol{x}_{1,\text{ub}}, \ldots, \boldsymbol{x}_{n,\text{ub}}, \ldots, \boldsymbol{x}_{N_{\text{ub}},\text{ub}}\right\}$, the following NLP is solved to determine the ellipsoid parameters, $\mathbf{H}_z$ and $\boldsymbol{d}_z$ (and therefore $\hat{\mathbf{P}}_z$ and $\hat{\boldsymbol{c}}_z$):

$$\min_{\mathbf{H}_z, \boldsymbol{d}_z, \boldsymbol{u}_n \in \mathcal{U}} \; J_1 = \det \mathbf{H}_z \tag{3.22a}$$

$$\text{subject to: } \boldsymbol{x}_n = \mathbf{H}_k \boldsymbol{x}_{n,\text{ub}} + \boldsymbol{d}_k, \quad \text{for } n \in [1, N_{\text{ub}}] \tag{3.22b}$$

$$\boldsymbol{x}_{\text{next}} = \sum_{\ell=1}^{L} \omega_\ell \begin{bmatrix} \boldsymbol{\alpha}_\ell & \boldsymbol{\beta}_\ell & \boldsymbol{\gamma}_\ell \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_n \\ \boldsymbol{u}_n \\ 1 \end{bmatrix} \tag{3.22c}$$

$$\omega_\ell = \frac{\left\| \begin{bmatrix} \boldsymbol{x}_n' & \boldsymbol{u}_n' \end{bmatrix}' - \boldsymbol{c}_\ell \right\|^{-2}}{\sum_{\ell=1}^{L} \left\| \begin{bmatrix} \boldsymbol{x}_n' & \boldsymbol{u}_n' \end{bmatrix}' - \boldsymbol{c}_\ell \right\|^{-2}} \tag{3.22d}$$

$$\left\| \boldsymbol{x}_{\text{next}} - \hat{\boldsymbol{c}}_{z-1} \right\|^2_{\hat{\mathbf{P}}_{z-1}} \leq 1 \tag{3.22e}$$

$$\mathbf{H}_z = \mathbf{L}_z \mathbf{L}_z' \tag{3.22f}$$

The *independent* decision variables in this NLP are the ellipsoid parameters, $\mathbf{H}_z$ and $\boldsymbol{d}_z$, and N_{ub} control moves corresponding to the N_{ub} initial conditions on the surface of the ellipsoid. The NLP is formulated to maximize the volume of the current RTRR ellipsoid while ensuring for N_{ub} uniformly distributed points on the surface of this ellipsoid, there exists a control action (as prescribed by a predictive controller using the data-based model) that can drive the ellipsoid surface point inside the next RTRR's ellipsoid. Equation (3.22b) represents the affine transformation of the N_{ub} unit ball points into the ellipsoid surface points. Equation (3.22f) represents the Cholesky decomposition of $\mathbf{H}_z$, where $\mathbf{L}_z \in \mathbb{R}^{n\times n}$ is a lower triangular matrix, and ensures $\mathbf{H}_z$ is positive-definite and symmetric. Note that ascertaining the feasibility of the optimization problem for the N_{ub} surface points does not guarantee the feasibility of all points on the surface, or for that matter, for the internal points. While the nonlinear and non-convex nature of the optimization problem prevents such guarantees, in practice this conclusion can be reached by choosing a sufficiently large N_{ub}. To ensure that the N_{ub} chosen is sufficiently large, in this work, N_{ub} was increased until changes in the solution were below a predefined tolerance.

To further verify that a control action exists to drive the states inside the next RTRR for the internal points of the ellipsoid, the NLP defined earlier in Equations (2.5a) to (2.5g) was

solved after substituting the data-based model for the first-principles model, $\hat{\mathbf{P}}_{(\cdot)}$ and $\hat{\boldsymbol{c}}_{(\cdot)}$ for $\tilde{\mathbf{P}}_{(\cdot)}$ and $\tilde{\boldsymbol{c}}_{(\cdot)}$, and removing the bottom-most layer.

3.4.2 Empirical Reverse-time Reachability Region-based Model Predictive Controller

In this section, using the ellipsoidal characterizations of the empirical RTRRs, we formulate a MPC design to steer a batch process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. To this end, consider a batch process described by Equation (3.1) for which empirical RTRR estimates have been characterized for a given δ and $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. The control action at sampling instance $z := (t_{\text{end}} - t)/\delta$ is computed by solving the following NLP:

$$\min_{\boldsymbol{u}[i]\in\mathcal{U}} J_{\hat{R}} = \sum_{i=1}^{P} \|\hat{\boldsymbol{x}}[i] - \hat{\boldsymbol{c}}_{z-i}\|^2_{\hat{\mathbf{P}}_{z-i}} + \|\boldsymbol{u}[i] - \boldsymbol{u}[i-1]\|^2_{\boldsymbol{\Pi}} \tag{3.23a}$$

$$\text{subject to: } \hat{\boldsymbol{x}}[0] = \boldsymbol{x}(t) \tag{3.23b}$$

$$\hat{\boldsymbol{x}}[k] = \sum_{\ell=1}^{L} \omega_\ell[k] \begin{bmatrix} \boldsymbol{\alpha}_\ell & \boldsymbol{\beta}_\ell & \boldsymbol{\gamma}_\ell \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}[k-1] \\ \boldsymbol{u}[k-1] \\ 1 \end{bmatrix}, \quad \text{for } k \in [0, P] \tag{3.23c}$$

$$\omega_\ell[k] = \frac{\left\| \begin{bmatrix} \hat{\boldsymbol{x}}'[k-1] & \boldsymbol{u}'[k-1] \end{bmatrix}' - \boldsymbol{c}_\ell \right\|^{-2}}{\sum_{\ell=1}^{L} \left\| \begin{bmatrix} \hat{\boldsymbol{x}}'[k-1] & \boldsymbol{u}'[k-1] \end{bmatrix}' - \boldsymbol{c}_\ell \right\|^{-2}} \tag{3.23d}$$

The objective function, $J_{\hat{R}}$, is formulated to minimize variations in the control moves and maintain the process states inside the empirical RTRRs over the prediction horizon, P. The relative importance of the two terms in $J_{\hat{R}}$ can be traded off using the move suppression matrix, $\boldsymbol{\Pi}$. The predictive model, specifically the nonlinear weighting function, makes this optimization problem a NLP, which can potentially be too computationally expensive for real-time application. However, this nonlinearity, while capturing the process dynamics much better than a single linear model, is likely to be much less severe compared to nonlinearities typically found in first-principles-based deterministic models. Consequently, the optimization problem should remain efficiently solvable even for moderate values of P.

Due to the unavoidable plant-model mismatch, the proposed MPC formulation does not offer any guarantees regarding the reachability of the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. In particular, even if one were to impose a constraint in the MPC formulation requiring the states to go inside the next RTRR's ellipsoid, the feasibility of the constraint (guaranteed if the current states are in the corresponding RTRR) would not guarantee that the states would be inside the RTRR at the next sampling instance. Yet, the determination of the RTRRs, specifically the ellipsoid matrices, provide useful weighting matrices to penalize state deviations to enforce the states to never significantly diverge from conditions where $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ can be reached. This also results in important fault-tolerant characteristics as discussed next.

If a fault occurs during batch operation, in the absence of any knowledge of the fault repair time, the only meaningful control objective is to take control action at the current time such that if full control effort were to be restored at the next sampling instance, reachability to $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ can be achieved. The RTRR-based MPC design, by trying to preserve the states within RTRRs during the fault repair period, implements exactly this control objective. In contrast, end-point-based predictive controllers try to achieve a (potentially) inherently unachievable objective - that of driving the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ subject to the reduced control effort, and in doing so, could drive the process to a point from where $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ is unreachable even after fault repair.

Remark 3.9: In order to generate empirical RTRRs in the state-space, the database must include state measurements. If the number of states is known and a deterministic process model is available but overly complex for online applications, the states can be back calculated offline from the database measurements using a variety of state estimation tools (i.e., a moving horizon estimator or an extended Kalman filter). The resulting states can be used to populate the database, and a state-space model of the form in Equation (3.17) can be developed. This model will capture nonlinearities, is more amenable to online applications, and is usable for generating empirical RTRR estimates (and a corresponding RTRR-based MPC design).

3.5 SIMULATION EXAMPLES

In this section, 2 simulation examples are presented. The first one illustrates the details of the proposed modelling approach and RTRR-based control design subject to varying initial conditions, time-varying uncertainties, and faults. Next, the modelling approach is applied to a process with limited measurements in order to identify models for the key process outputs. These models are subsequently used to design a trajectory tracking controller.

3.5.1 Fed-batch Reactor

In this section, a data-based model of a fed-batch process is extracted from an artificially generated historical database using the proposed modelling approach. The resulting model is utilized to design an empirical RTRR-based predictive controller. To this end, consider a fed-batch reactor where a series of reactions of the form:

$$2A \xrightarrow{k_1} B \xrightarrow{k_2} 3C$$

take place. The state-space process model is available in [43, pp. 625 – 627] and reproduced below.

$$\dot{x}_1(t) = r_1 + \frac{C_{A,\text{in}} - x_1}{x_5} u_1 \tag{3.24a}$$

$$\dot{x}_2(t) = -\frac{1}{2} r_1 + r_2 - \frac{x_2}{x_5} u_1 \tag{3.24b}$$

$$\dot{x}_3(t) = -3r_2 - \frac{x_3}{x_5} u_1 \tag{3.24c}$$

$$\dot{x}_4(t) = \frac{w_2(u_2 - x_4) + C_{A,\text{in}} C_{p,A} u_1 (w_1 - x_4) + \left(\Delta H_1 r_1 + \Delta H_2 r_2\right) x_5}{\left(x_1 C_{p,A} + x_2 C_{p,B} + x_3 C_{p,C}\right) x_5 + N_{\text{cat}} C_{p,\text{cat}}} \tag{3.24d}$$

$$\dot{x}_5(t) = u_1 \tag{3.24e}$$

where the reaction rates, r_1 and r_2, are given according to:

$$r_1 = -k_1 x_1 = -k_{10} \exp\left(\frac{E_1}{R}\left(\frac{1}{T_1} - \frac{1}{x_4}\right)\right) x_1$$

$$r_2 = -k_2 x_2 = -k_{20} \exp\left(\frac{E_2}{R}\left(\frac{1}{T_2} - \frac{1}{x_4}\right)\right) x_2$$

The state vector is $\boldsymbol{x} = \begin{bmatrix} C_A & C_B & C_C & T & V \end{bmatrix}'$ where C_A (mol/L), C_B (mol/L), and C_C (mol/L) denote the concentrations of species A, B, and C (respectively), and T (K) and V (L) denote the reactor temperature and volume (respectively). The inputs were taken to be the inlet feed rate, F (L/h), and heating coil temperature, T_{hx} (K), $\boldsymbol{u} = \begin{bmatrix} F & T_{\text{hx}} \end{bmatrix}'$. The input constraints were $\boldsymbol{u}_{\min} = \begin{bmatrix} 0 & 288 \end{bmatrix}'$ and $\boldsymbol{u}_{\max} = \begin{bmatrix} 20 & 360 \end{bmatrix}'$. In all the simulations, the vector of model uncertainties was $\boldsymbol{w} = \begin{bmatrix} T_{\text{in}} & UA \end{bmatrix}'$ where T_{in} (K) is the inlet temperature and UA (cal/(h · K)) is the heat exchanger coefficient. To simulate disturbances, T_{in} was stochastically varied throughout the duration of each batch around its nominal value in the range 295 – 305 K. For the heat exchanger coefficient, at the start of each batch, UA was assigned a value in the range $28,620 - 30,349$ cal/(h · K) and then decreased exponentially to simulate fouling. The physical meaning of the model parameters and their nominal values are shown in Table 3.2.

Table 3.2: Parameters for the fed-batch reactor model in Equations (3.24a) to (3.24e)

Parameter	Description	Value	Unit
$C_{A,\text{in}}$	Inlet A concentration	4	mol/L
T_{in}	Inlet temperature	300	K
UA	Heat transfer coefficient × Area	30,000	cal/(h · K)
$C_{p,A}$	Heat capacity of species A	30	cal/(mol · K)
$C_{p,B}$	Heat capacity of species B	60	cal/(mol · K)
$C_{p,C}$	Heat capacity of species C	20	cal/(mol · K)
$C_{p,\text{cat}}$	Heat capacity of catalyst	35	cal/(mol · K)
N_{cat}	Amount of catalyst	100	mol
ΔH_1	Heat of reaction for $A \rightarrow \frac{1}{2}B$	$-6,500$	cal/mol A
ΔH_2	Heat of reaction for $B \rightarrow 3C$	8,000	cal/mol B
k_{10}	Reaction rate constant at T_1 for $A \rightarrow \frac{1}{2}B$	1.05	1/h
k_{20}	Reaction rate constant at T_2 for $B \rightarrow 3C$	0.05	1/h
T_1	Reference temperature at which k_{10} is computed	340	K
T_2	Reference temperature at which k_{20} is computed	300	K
E_1	Activation energy for $A \rightarrow \frac{1}{2}B$	9,900	cal/mol
E_2	Activation energy for $B \rightarrow 3C$	7,000	cal/mol
R	Universal gas constant	1.986	cal/(mol · K)

The primary control objective considered was to drive the process inside an ellipsoidal neighbourhood around $\hat{\boldsymbol{c}}_0 = \boldsymbol{x}_{\text{des}} = \begin{bmatrix} 2.752 & 1.601 & 0.8422 & 365.756 & 112.425 \end{bmatrix}'$. The ellipsoid matrix was specified as $\hat{\mathbf{P}}_0 = \text{diag}\{25, 400, 100, 0.04, 1\}$. The batch termination time, t_{end}, was taken to be 1 hour with a sampling period of $\delta = 0.025$ hours. The control performance was assessed by the level set of the desired end-point neighbourhood ellipsoid corresponding to $\boldsymbol{x}(t_{\text{end}})$ with a value of less than 1 indicating $\boldsymbol{x}(t_{\text{end}}) \in \mathcal{B}(\boldsymbol{x}_{\text{des}})$.

Data-based Model Development

A database of 40 batches was generated (using the state-space model) with 10 batches set aside as the validation batches. With 2 inputs, reference trajectories of C_B and T (see Figure 3.1) were chosen to be tracked by manipulating F and T_{hx} (respectively) using 2 PI controllers.

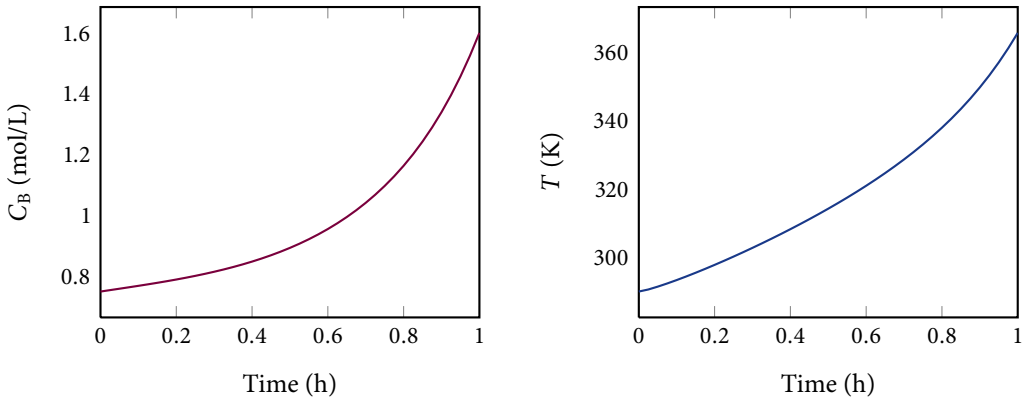


Figure 3.1: Reference C_B and T profiles for the fed-batch process. These profiles were tracked using 2 PI controllers when generating the database.

Both PI controllers were tightly tuned for 1 set of initial conditions and fixed for the remaining 39 batches. The tuning criteria was to minimize the integral of time-weighted absolute error (ITAE) while achieving reasonably smooth input trajectories. For a more realistic representation of plant data, sensor noise was also considered. The range of initial conditions and sensor noise levels are summarized in Table 3.3.

Table 3.3: Simulation parameters used for database generation for the fed-batch process

State	Range of initial conditions	Sensor noise
C_A	4.861 – 5.106 mol/L	±2.2% of original signal
C_B	0.692 – 0.801 mol/L	±2.8% of original signal
C_C	0.446 – 0.539 mol/L	±2.1% of original signal
T	280.931 – 300.057 K	0.10 standard deviation
V	97.952 – 101.959 L	0.10 standard deviation

Given the database, Algorithm 3.2 was carried out with PLS regression. Because full state measurements were assumed, all lags were set to 1. The number of clusters was varied from $L_{\min} = 10$ to $L_{\max} = 100$. The lowest RMSE was obtained with $L = 20$ clusters and 142 principal components[3]. Figure 3.2 illustrates the predictive capabilities of the final data-based model[4] for a set of initial conditions and input trajectories in the validation data set. The temperature range in the figure is significantly larger compared to the concentration ranges because its initial value (and values for the batch duration) was an order of magnitude greater than all the concentrations. As a result, the prediction errors for the concentrations are far

[3]Note that the number of columns in $\mathbf{\Psi}$ (the regressor matrix) was 160 because there were 5 states, 2 inputs, and a bias term (8 total terms) for 20 local models

[4]Note that the volume, V, has been omitted.

more noticeable. Overall, the multi-model approach was able to capture the major process nonlinearities.

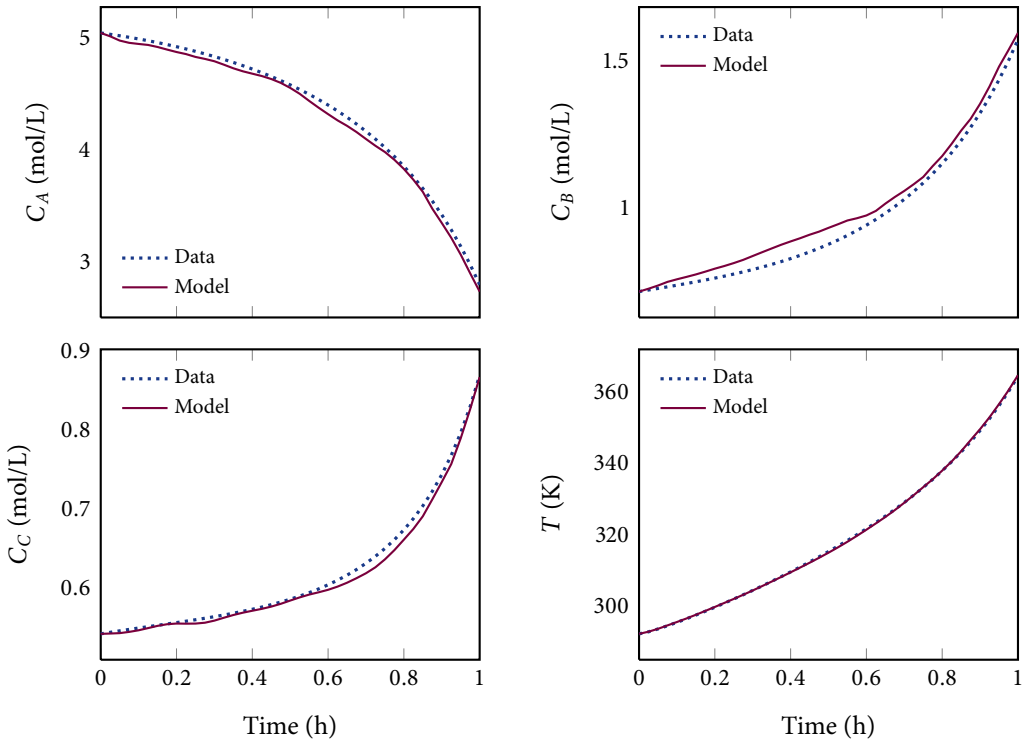


Figure 3.2: Comparison of the data-based model's outputs with the corresponding trajectories in the validation data for the fed-batch process

Closed-loop Results

Next, the RTRR-based MPC design proposed in Section 3.4.2 was implemented, and the control performance was compared with PI control. Closed-loop simulations were performed for 10 new initial conditions, which were all within the empirical RTRR ellipsoid at the initial time. The PI controller tunings were set to those used during database generation. The MPC tuning parameters were set as follows: $\mathbf{\Pi} = \text{diag}\{0.01, 0.005\}$ and $P = 18$. The control performance is summarized in Table 3.4. The RTRR-based MPC design was able to drive the process inside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ for all the initial conditions whereas PI control failed in more than half of the cases. In Figure 3.3, a representative set of closed-loop profiles is presented.

Table 3.4: Final $\mathcal{B}(\boldsymbol{x}_{\text{des}})$ level sets from PI control and the RTRR-based MPC with no faults for the fed-batch process

Initial condition:	1	2	3	4	5	6	7	8	9	10
PI control:	2.48	7.22	0.44	1.97	2.23	0.81	0.68	4.05	0.46	1.19
RTRR-based MPC:	0.70	0.94	0.24	0.93	0.16	0.16	0.026	0.088	0.15	0.19

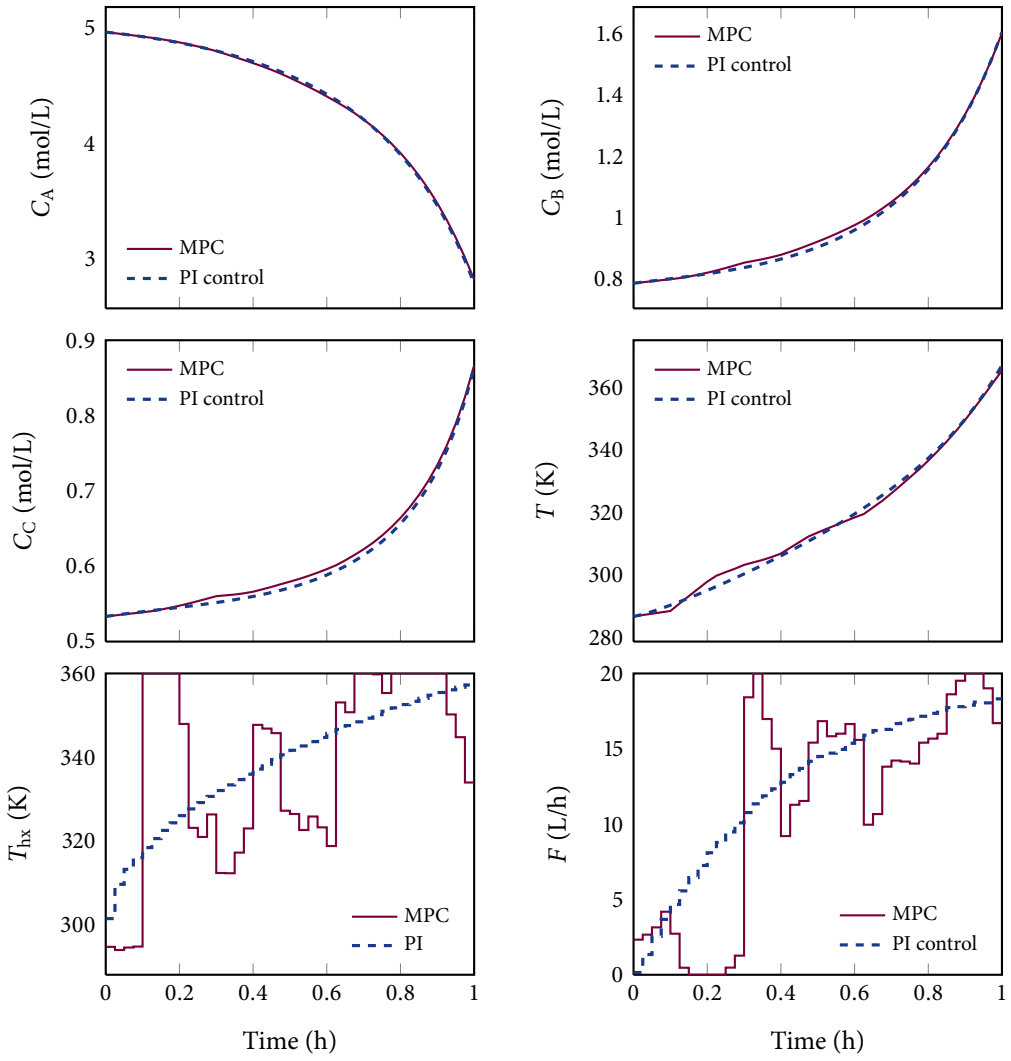


Figure 3.3: Representative state and input profiles from PI control and RTRR-based MPC with no faults for the fed-batch process

As evidence that the RTRR-based MPC optimization problem is efficiently solvable despite being a NLP, we note that with $P = 18$, the longest CPU time required time to solve the NLP was 0.45 seconds using GAMS with IPOPT as the solver on Intel Quad Core machine.

To demonstrate the fault-tolerance of the RTRR-based MPC design, we considered faults in both control actuators and compared the performance of the MPC design with PI control. Starting from $\boldsymbol{x}(t_0) = \begin{bmatrix} 4.98 & 0.76 & 0.54 & 289.49 & 100.46 \end{bmatrix}'$, we considered the scenario where at $t_{\text{fault}} = 0.25$ hours, the actuators associated with both inputs failed, and their maximum values were reduced to $\boldsymbol{u}_{\max} = \begin{bmatrix} 10 & 310 \end{bmatrix}'$ (from $\begin{bmatrix} 20 & 360 \end{bmatrix}'$). At $t_{\text{repair}} = 0.45$ hours, the faults were rectified and full control effort was recovered. The closed-loop profiles for this case are shown in Figure 3.4.

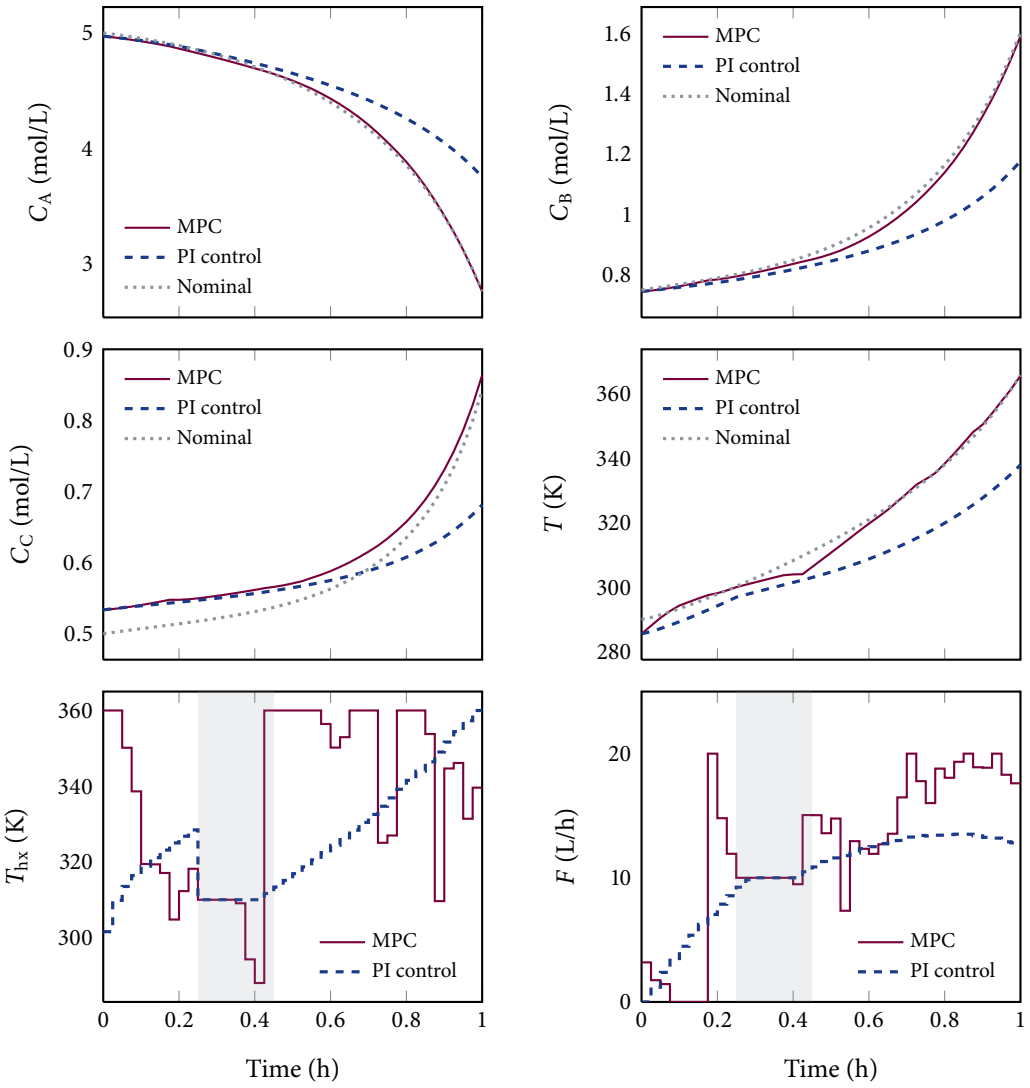


Figure 3.4: State and input profiles from PI control and RTRR-based MPC with finite duration actuator failures for the fed-batch process. The failure period is shaded.

The batch was driven to $\boldsymbol{x}(t_{\text{end}}) = \begin{bmatrix} 3.65 & 1.24 & 0.71 & 339.08 & 110.76 \end{bmatrix}'$ with PI control, corresponding to a final level set of 104.39, which was well outside $\mathcal{B}(\boldsymbol{x}_{\text{des}})$. On the other hand, the final level set for the RTRR-based controller was 0.17. Note that during the failure period, the prediction horizon in the RTRR-based MPC design was reduced from $P = 18$ to $P = 1$ to avoid having to assume the failure situation any longer than necessary. The PI controller prescribed the heat exchanger temperature to remain saturated during the failure period whereas the RTRR-based controller prescribed a more meaningful input trajectory towards the latter stages of the fault. As a result, the RTRR-based controller was able to recover the process after the fault and essentially began to track the nominal state trajectories which terminated at the desired end-point. Also note that the states at the onset of the fault differed for the 2 controllers since the inputs prescribed up to t_{fault} were different.

3.5.2 Nylon-6,6 Polymerization

In this section, we apply the data-based modelling approach on a complex, nonlinear nylon-6,6 batch polymerization process to extract models for the key output variables. Subsequently, we employ these models in a trajectory tracking predictive controller and compare its tracking performance against PI control and LV-MPC (see Section 3.2.5).

While there exist several chemical routes to produce nylon-6,6 polymer, for this work, we focused on its production by the amidation of adipic acid and hexamethylenediamine (HMD) in a batch reactor. In this polymerization, the reactor is initially charged with molten adipic acid and HMD (from an evaporator) in approximately stoichiometric (1 : 1) proportions. The reaction model is summarized by the following equations.

$$\textit{Degradation} \quad \mathrm{C} \rightarrow \mathrm{SE} + \mathrm{W} \tag{3.25a}$$

$$\mathrm{L} \rightarrow \mathrm{SE} + \mathrm{A} \tag{3.25b}$$

$$\textit{Polyamidation} \quad \mathrm{A} + \mathrm{C} \leftrightarrows \mathrm{L} + \mathrm{W} \tag{3.25c}$$

where A is an amine end group, C is a carboxyl end group, W is a water molecule, L is a polymer link, and SE is a (non-reactive) stabilized end group. The polymerization reaction is treated as a second order, reversible reaction of a-a/b-b type that is commonly described in terms of functional groups for simplicity (see [44]). During the polymerization reaction (given by Equation (3.25c)), amine end groups (A) in HMD or the polymer chain react with carboxylic end groups (C) on either the adipic acid or polymer chain, forming a polymer link (L) and a water molecule (W). The degradation reactions, Equations (3.25a) to (3.25b), are considered due to their effect on the reaction mixture temperature. In order to meet the typical desired end-use quality, a high extent of reaction (over 99%) is required, which, in turn, calls for shifting the polymerization reaction towards completion by vaporizing water and then venting the vaporized water. Consequently, the polymerization is typically carried out in an autoclave reactor equipped with a steam jacket for providing the heat needed for vaporization (and reaction) and a valve for venting vaporized water.

The polymerization occurs in 3 main phases as described below.

Heating phase: The vent valve is closed to prevent the loss of volatile HMD, and heat is supplied through the steam jacket, driving the polymerization reaction. After a certain extent of reaction, the valve is opened, initiating the boiling phase.

Boiling phase: Excess water is removed, which is important for achieving high molecular weight of the final polymer. After venting water for an appropriate amount of time, the vent is closed, and the finishing phase begins.

Finishing phase: The vent remains closed and the final quality characteristics of the polymer are developed.

To illustrate the proposed modelling and control approach, we utilized the mathematical model of this process in [44]. The modelling assumptions (and their explanations), parameter values, and kinetic relationships are available in [44, 45] and omitted here for brevity. The final state-space model, which takes the form shown in Equation (3.1), consists of 9 coupled ODEs with the state vector comprised of the molar amounts of each functional group and evaporated HMD, the reaction medium mass, temperature, and volume, and reactor pressure.

One difference between the model in [44] and the one used in this work is that we did not neglect the reactor pressure dynamics. In [44], the reactor pressure is treated as a process *input* due to the assumption of fast dynamics whereas we appended the model equations with an ODE for the pressure that is equal to the product of a (negative) gain term and the vent rate. In other words, we considered the reactor pressure as an additional state compared to the model in [44] and modelled it using a simple linear, first order ODE with respect to the vent rate: $\frac{dP}{dt} = Kv$ where K is the negative gain. In this way, the reactor pressure was treated as a control variable that was influenced by the vent rate, and the control problem (to be discussed shortly) was a multiple-input-multiple-output problem.

The process outputs, $\boldsymbol{y}$, were taken to be the reaction mixture temperature, T (K), and reactor pressure, P (psi). The temperature and pressure measurements were corrupted by normally distributed, zero-mean white noise with standard deviations of 0.16 K and 0.17 psi, respectively. The inputs, $\boldsymbol{u}$, were taken to be the steam jacket pressure, P_j (psi), and vent rate, v (kg/h). Thus, the output and input vectors were defined as follows: $\boldsymbol{y} = \begin{bmatrix} T & P \end{bmatrix}'$ and $\boldsymbol{u} = \begin{bmatrix} P_j & v \end{bmatrix}'$. The physical limitations in the process design were assumed to impose the following input constraints: $\boldsymbol{u}_{\min} = \begin{bmatrix} 700 & 0 \end{bmatrix}'$ and $\boldsymbol{u}_{\max} = \begin{bmatrix} 1800 & 2000 \end{bmatrix}'$. The duration of the batch was 3 hours with a sampling period of 60 seconds. A schematic of the process is shown in Figure 3.5.

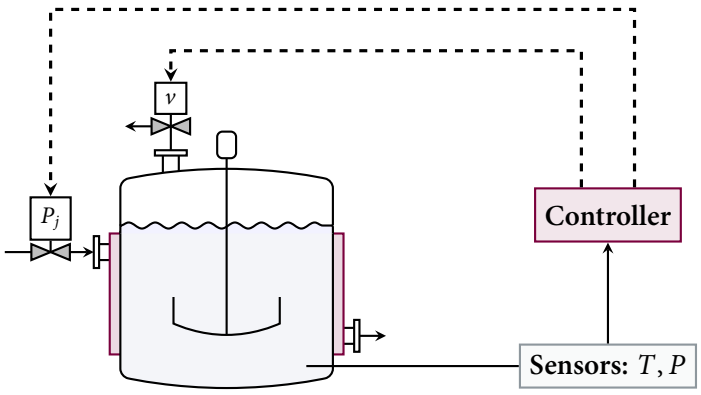


Figure 3.5: Schematic of the nylon-6,6 batch polymerization process

Nylon-6,6 polymer quality is defined by its molecular weight, MW, and residual amide concentration, R_{NH_2}. Accordingly, the control objective is to achieve target values of these 2 quality variables. However, these variables are rarely measured online, making direct control to desired qualities impractical. Instead, the product quality is usually monitored through secondary process variable measurements, such as the temperature and reactor pressure. Thus, a common control strategy has been to track reference trajectories of these measurable variables that are obtained through offline optimization, from historical batch data, or from high level controllers which periodically re-optimize the trajectories in response to encountered disturbances (i.e., mid-course corrections as shown in [46]). In [44], industrially popular tracking control strategies specific to this process are evaluated in terms of their ability to produce the desired product qualities when encountering common disturbances. For this work, we chose to track trajectories of the reaction medium temperature, T, and reactor pressure, P, by manipulating the steam jacket pressure, P_j and vent rate, v. We assumed reference trajectories for T and P, denoted by T_{ref} and P_{ref} (respectively), were identified appropriately in some fashion, and Figure 3.6 presents these specific trajectories.

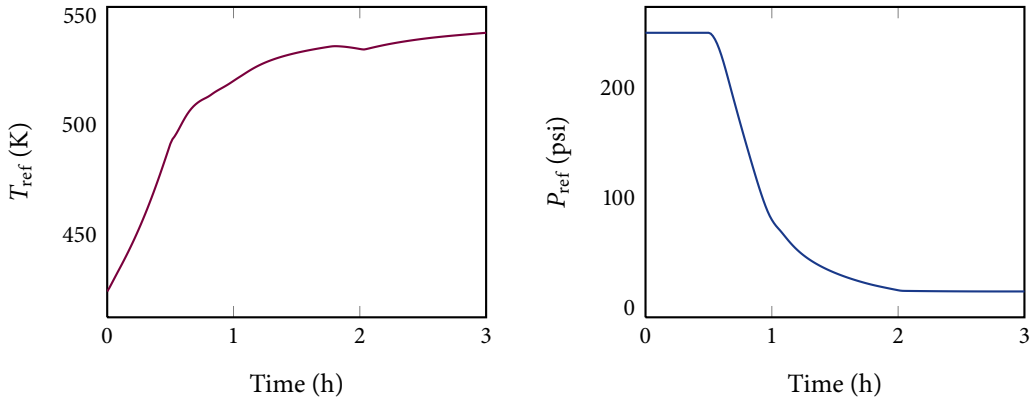


Figure 3.6: Reference T and P profiles for the nylon-6,6 batch polymerization process. These trajectories are required to be tracked in the control objective.

The trajectories in Figure 3.6 were assumed to yield a nominal desired polymer quality at batch termination. However, even under perfect tracking during a *new* batch, there is no guarantee that the desired quality will be met because unavoidable disturbances encountered during the batch can effectively alter the relationship between the quality and the process outputs. Thus, the reference trajectories may no longer yield the desired polymer quality, and they essentially have to be “re-optimized" in some fashion. The development of an inferential quality model, which can be used to predict the final product quality from the process outputs and inputs and then the subsequent integration of this model within a control design is outside the scope of this chapter but later addressed in Chapter 5. In this chapter, we specifically focus on the complexities associated with the trajectory tracking problem.

Data-based Model Development

Data-based models for the 2 outputs are developed in this section from an artificially generated batch database comprised only of input and output measurements. The database was generated by simulating the state-space model 15 times from different initial conditions with 5 batches reserved as the validation data set. To mimic a typical industrial batch database, which consists mostly of "successful" batches, the reference profiles in Figure 3.6 were tracked using 2 PI controllers for the database generation. For the control loop pairing, the vent rate was used to track the reactor pressure while the steam jacket pressure was used to track the reaction mixture temperature. Both controllers were tightly tuned for 1 set of initial conditions and fixed for the 14 remaining batches. The tuning goal was to minimize the ITAE while attaining acceptable input trajectories.

Because the reactor pressure dynamics were significantly faster than the temperature dynamics, there was a weak correlation between the 2 outputs. Consequently, individual models for the outputs were identified with PCR as opposed to a single PLS model that predicted both outputs simultaneously. The model identification procedure in Algorithm 3.2 was carried out for T and P separately. The minimum number of lags considered was 0, which meant the variable was not included in the model, and the maximum number was 2. The clusters were varied from $L_{\min} = 2$ to $L_{\min} = 20$. The lag structure and number of clusters, L, for the 2 outputs that yielded the lowest RMSE values are tabulated in Table 3.5.

Table 3.5: Final lag structures, number of clusters, L, and RMSE values of the data-based models for the nylon-6.6 batch polymerization process

	Lags					
Output	T	P	P_j	v	L	**RMSE**
T	1	0	1	1	5	1.65
P	0	1	0	1	1	0.18

From Table 3.5, the reactor pressure was not used in predicting the temperature, and its dynamics were best captured with a single linear, first order model. These results were consistent with the fundamental process model; the significantly faster pressure dynamics led to a decoupling of the pressure from the other states (i.e., the pressure did not influence any of the other states and vice versa), and the pressure ODE was simply the product of a constant gain term and the vent rate (a linear, first order model). Note that despite the decoupling of the outputs, the control problem cannot be decomposed into 2 single-input-single-output problems because the vent rate affected both outputs. Another observation from Table 3.5 is that the lag structure for the 5 temperature models corresponds to a first order model between the outputs and inputs. One explanation for this behaviour is the assumption of the

same lag structure for all the local models. With this assumption, using all first order models minimized the possibility of over-fitting, and in this case, yielded the lowest RMSE values.

In Figure 3.7, we compare database trajectories with the data-based model for a random set of initial conditions in the validation data set. Overall, the multi-model approach captured the major nonlinearities and provided relatively reliable predictions of both outputs.

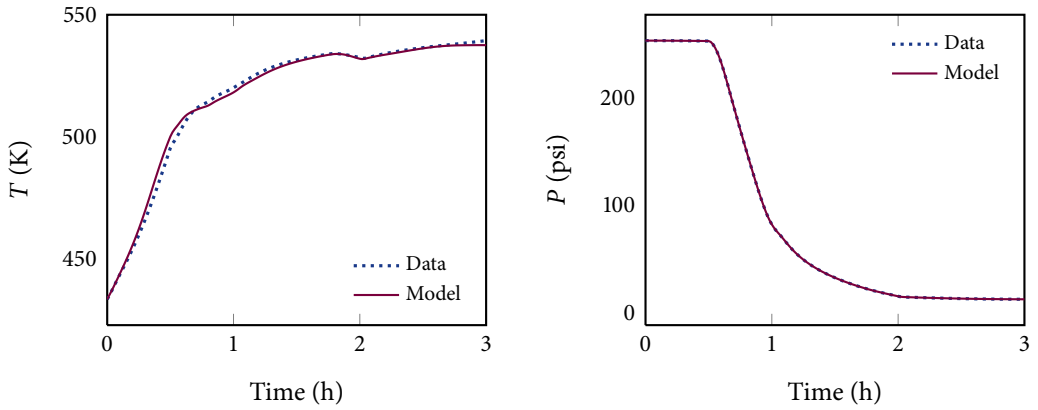


Figure 3.7: Comparison of the data-based models' outputs with the corresponding trajectories in the validation data for the nylon-6,6 batch polymerization process

Closed-loop Results

In this section, we use the models identified in the previous section to design a trajectory tracking controller, and then benchmark its performance against PI control and LV-MPC. For the proposed trajectory tracking controller, the control action at each sampling instance was computed by solving the NLP shown below.

$$\min_{\boldsymbol{u}[k]\in\mathcal{U}} J_T = \sum_{k=1}^{P} \|\hat{\boldsymbol{y}}[k] - \boldsymbol{y}_{\text{ref}}[k]\|_{\boldsymbol{\Xi}}^2 + \|\boldsymbol{u}[k] - \boldsymbol{u}[k-1]\|_{\boldsymbol{\Pi}}^2 \tag{3.26a}$$

$$\text{subject to: } \hat{\boldsymbol{y}}[0] = \boldsymbol{y}(t) \tag{3.26b}$$

$$\text{Equation (3.15) for } y_1 = T \text{ and } y_2 = P, \quad \text{for } k \in [1, P] \tag{3.26c}$$

The first term in the objective function penalized discrepancies between the predicted output, $\hat{\boldsymbol{y}}$, and the reference, $\boldsymbol{y}_{\text{ref}}$, trajectories over the prediction horizon[5], P, and the second term penalized the control rate. The positive-definite matrices, $\boldsymbol{\Xi}$ and $\boldsymbol{\Pi}$, traded-off the relative importance of the output and input performances. Equation (3.26c) states that the data-based models were the underlying predictive models in the MPC formulation, and Equation (3.26b) represents the initialization of the optimization problem at the plant conditions.

[5]Because this MPC formulation is a shrinking horizon optimization problem (as is the case for batch processes), the prediction horizon must be appropriately shortened when necessary so as not to exceed the batch duration.

Closed-loop simulations for 10 new initial conditions were performed using the proposed trajectory tracking MPC design, and the performance was compared against PI control and the LV-MPC approach that was reviewed in Section 3.2.5. All initial conditions were ensured to be within the range of initial conditions in the training data. All controllers were tuned once for a specific set of initial conditions and left unchanged for the remainder of the simulations to avoid confounding the results with tuning. The tuning parameters for the proposed MPC and LV-MPC designs are presented in Table 3.6.

Table 3.6: Tuning parameters for the proposed MPC and LV-MPC designs during closed-loop simulations of the nylon-6,6 batch polymerization process

	Proposed MPC	**LV-MPC**
$\boldsymbol{\Xi}$	diag $\{2.75, 27.5\}$	diag $\{1, \ldots, 1\}$
$\boldsymbol{\Pi}$	diag $\{0.02, 0.02\}$	$\mathbf{0}$
M (lags)	–	5
P	12	10

Note that with $P = 12$, the proposed MPC design was efficiently solvable; the average CPU time required time to solve the MPC optimization problem (as reported by the Matlab functions, `tic` and `toc`) was 0.69 seconds (using GAMS with IPOPT as the solver on Intel Quad Core machine). For LV-MPC, the closed form control law in Equation (3.10) (after clipping for input constraints) yielded better performance compared to solving the optimization problem in Equations (3.11a) to (3.11c), and the corresponding results are shown in this section. The results from all 3 controllers are summarized in Table 3.7 in terms of the ITAE.

Table 3.7: Tracking performance with PI control, the proposed MPC design, and the LV-MPC design for 10 new initial conditions for the nylon-6,6 batch polymerization process

	Temperature ITAE			*Pressure ITAE*		
Initial Condition	**PI**	**Proposed MPC**	**LV-MPC**	**PI**	**Proposed MPC**	**LV-MPC**
1	9.10	2.18	8.34	3.08	1.01	22.22
2	10.18	1.51	7.00	3.46	1.22	13.91
3	3.33	1.33	5.80	5.02	1.47	28.75
4	4.95	1.76	7.76	2.71	2.03	20.93
5	12.99	2.98	10.53	1.93	1.43	15.33
6	6.99	1.21	6.82	10.29	1.04	9.51
7	13.29	2.72	9.38	5.18	1.57	29.53
8	3.14	1.19	6.54	1.20	0.860	18.44
9	14.63	1.90	8.89	1.43	1.31	15.73
10	4.91	1.89	7.50	4.61	1.58	27.62
Average ITAE:	8.35	1.87	7.86	3.89	1.35	20.20

On average, for temperature tracking, both predictive controllers were superior to the PI controller. The proposed MPC design offered a significant advantage of approximately 78% with LV-MPC yielding a 6% average improvement. For pressure tracking, the proposed MPC design yielded the most desirable results followed by the PI controller then LV-MPC. In all simulations, the proposed predictive controller outperformed PI control and LV-MPC for both temperature and pressure tracking.

One explanation for the poor results obtained using LV-MPC was the database used to develop the PCA model. For the proposed modelling approach and LV-MPC modelling, an identical database of closed-loop PI runs was used. However, in generating the databases used to develop the PCA models in [26], dither signals were added on top of all the inputs to help meet identifiability conditions. This was not required in the proposed modelling approach. Another possible explanation is that the correlation structure determined by the PCA model did not hold for the new initial conditions due to strong nonlinearities and/or there were significant changes in the correlation structure as the batch proceeded (see Remark 3.6). The key point of the simulations presented though is to the show the superior performance of the proposed MPC design compared to PI control strategies and to benchmark it against a "simple" implementation of LV-MPC (not necessarily the best implementation of LV-MPC).

A representative set of closed-loop profiles is presented in Figure 3.8. For this set of initial conditions, the ITAEs for the proposed predictive controller improved on the PI controller by 77% and 26% and on the LV-MPC design by 72% and 91% for temperature and pressure tracking (respectively). Overall, the simulation results demonstrated the advantages of implementing the proposed trajectory tracking predictive controller over PI control and LV-MPC.

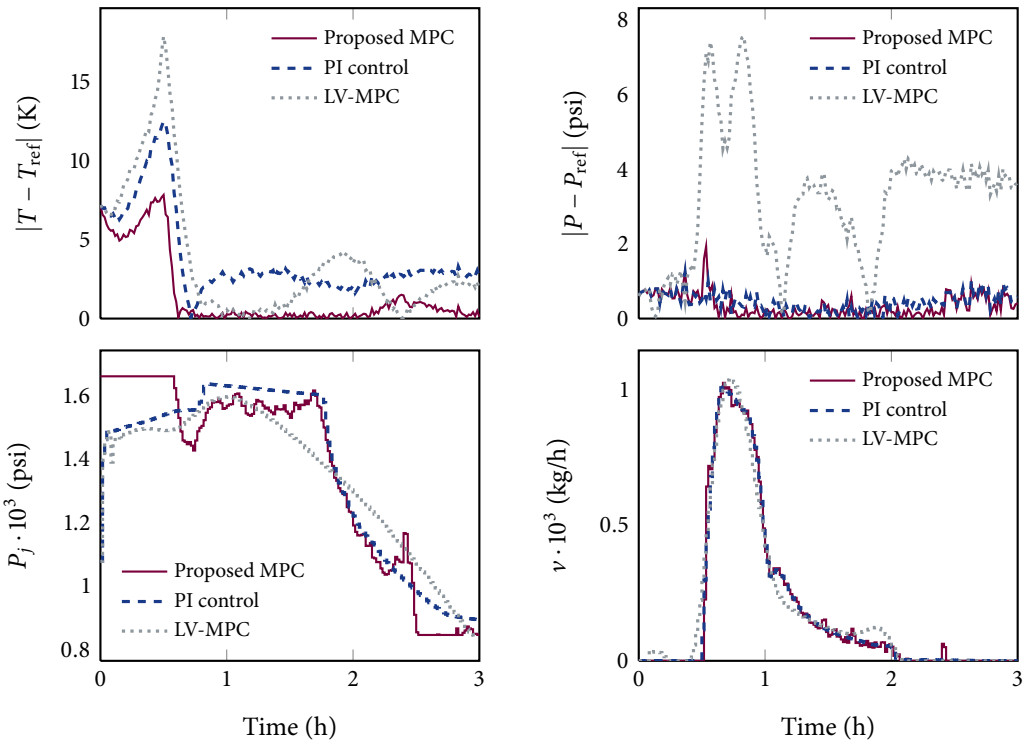


Figure 3.8: Representative tracking error and input profiles from PI control, the proposed MPC design, and the LV-MPC design for the nylon-6,6 batch polymerization process

3.6 CONCLUSIONS

In this chapter, we addressed the problem of uniting empirical/data-based modelling approaches with nonlinear control tools for the control of batch processes. In the proposed modelling approach, we exploited the availability of historical batch data, the simplicity of local linear models, the data extraction capabilities of PLS/PCR, and the use of appropriate clustering and weighting techniques in conjunction with multiple models to capture the nonlinearities of batch processes. The resulting model from this approach was employed to generate empirical RTRRs, which were subsequently incorporated in a predictive control design. The efficacy of the RTRR-based MPC design and superior performance, as well as fault-handling ability, compared to PI control was demonstrated through a fed-batch reactor simulation example.

The data-based modelling approach was also applied to develop models for use in a trajectory tracking MPC design for a nylon-6,6 batch polymerization process with limited measurements. The resulting models were used to develop a predictive controller for tracking reference trajectories of the key process outputs, namely the reaction mixture temperature and reactor pressure. Closed-loop simulation results (subject to noise and disturbances in the

feed conditions) clearly demonstrated the advantages of using the proposed control design over PI control and a simple implementation of LV-MPC.

REFERENCES

[1] S. Aumi and P. Mhaskar, "Integrating data-based modelling and nonlinear control tools for batch process control," *AIChE J.*, 2011, (in press). DOI: 10.1002/aic.12720.

[2] S. Aumi, B. Corbett, P. Mhaskar, and T. Clarke-Pringle, "Data-based modelling and control of nylon-6,6 batch polymerization," *IEEE Trans. Control. Sys. Technol.*, 2011, (in press).

[3] S. Aumi and P. Mhaskar, "Integrating data-based modelling and nonlinear control tools for batch process control," in *Proc. of the American Control Conf.*, San Francisco, CA, 2011, pp. 2534–2539.

[4] S. Aumi, B. Corbett, and P. Mhaskar, "Data-based modelling and control of nylon-6,6 batch polymerization," in *Proc. of the American Control Conf.*, San Francisco, CA, 2011, pp. 2540–2545.

[5] R. Cardello and K. Y. San, "Application of gain scheduling to the control of batch bioreactors," in *Proc. of the American Control Conf.*, Minneapolis, MN, 1987, pp. 682–686.

[6] T. Clarke-Pringle and J. F. MacGregor, "Nonlinear adaptive temperature control of multi-product, semi-batch polymerization reactors," *Comp. & Chem. Eng.*, vol. 21, no. 12, pp. 1395–1409, 1997.

[7] C. Kravaris and C.-B. Chung, "Nonlinear state feedback synthesis by global input/output linearization," *AIChE J.*, vol. 33, no. 4, pp. 592–603, 1987.

[8] C. Kravaris and M. Soroush, "Synthesis of multivariable nonlinear controllers by input/output linearization," *AIChE J.*, vol. 36, no. 2, pp. 249–264, 1990.

[9] G. Gattu and E. Zafiriou, "Nonlinear quadratic dynamic matrix control with state estimation," *Ind. & Eng. Chem. Res.*, vol. 31, no. 4, pp. 1096–1104, 1992.

[10] T. Peterson, E. Hernandez, Y. Arkun, and F. Schork, "Nonlinear DMC algorithm and its application to a semibatch polymerization reactor," *Chem. Eng. Sci.*, vol. 47, no. 4, pp. 737–753, 1992.

[11] D. Shi, P. Mhaskar, N. El-Farra, and P. Christofides, "Predictive control of crystal size distribution in protein crystallization," *Nanotechnol.*, vol. 16, no. 7, pp. 562–574, 2005.

[12] D. Shi, N. H. El-Farra, M. Li, P. Mhaskar, and P. D. Christofides, "Predictive control of particle size distribution in particulate processes," *Chem. Eng. Sci.*, vol. 61, no. 1, pp. 268–281, 2006.

[13] V. Havlena and P. Barva, "Nonlinear MPC and inferential sensing for PVC production," *Proc. of the Conf. on Control Applications*, vol. 2, pp. 915–920, 1999.

[14] P. Geladi and B. Kowalski, "Partial least-squares regression: A tutorial," *Anal. Chim. Acta*, vol. 185, pp. 1–17, 1986.

[15] N. Fletcher, A. Morris, G. Montague, and E. Martin, "Local dynamic partial least squares approaches for the modelling of batch processes," *Can. J. of Chem. Eng.*, vol. 86, pp. 960–970, 2008.

[16] G. Baffi, E. B. Martin, and A. J. Morris, "Non-linear dynamic projection to latent structures modelling," *Chemom. & Intell. Lab. Syst.*, vol. 52, no. 1, pp. 5–22, 2000.

[17] G. Baffi, E. Martin, and A. Morris, "Non-linear projection to latent structures revisited: The quadratic PLS algorithm," *Comp. & Chem. Eng.*, vol. 23, no. 3, pp. 395–411, 1999.

[18] —, "Non-linear projection to latent structures revisited (the neural network PLS algorithm)," *Comp. & Chem. Eng.*, vol. 23, no. 9, pp. 1293–1307, 1999.

[19] R. Rosipal, "Kernel Partial Least Squares for Nonlinear Regression and Discrimination," *Neural Network World*, vol. 13, no. 3, pp. 291–300, 2003.

[20] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari, "A Clustering Technique for the Identification of Piecewise Affine Systems," *Automatica*, vol. 39, pp. 205–217, 2003.

[21] X. Jin and B. Huang, "Robust identification of piecewise/switching autoregressive exogenous process," *AIChE J.*, vol. 56, no. 7, pp. 1829–1844, 2010.

[22] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man Cybern.*, vol. 15, no. 1, pp. 116–132, 1985.

[23] N. Nandola and S. Bhartiya, "A multiple model approach for predictive control of nonlinear hybrid systems," *J. of Process Control*, vol. 18, no. 2, pp. 131–148, 2008.

[24] B. Aufderheide and B. W. Bequette, "Extension of dynamic matrix control to multiple models," *Comp. & Chem. Eng.*, vol. 27, no. 8-9, pp. 1079–1096, 2003.

[25] F. D. Palma and L. Magni, "A multi-model structure for model predictive control," *Annu. Rev. Control*, vol. 28, pp. 47–52, 2004.

[26] J. Flores-Cerrillo and J. F. MacGregor, "Latent variable MPC for trajectory tracking in batch processes.," *J. Process Control*, vol. 15, no. 6, pp. 651–663, 2005.

[27] W. L. Lin, S. J. Qin, and L. Ljung, "On consistency of closed-loop subspace identification with innovation estimation," in *Proc. of the IEEE Conf. on Decision and Control*, 2004, 2195–2200.

[28] B. Huang, X. S. Ding, and S. J. Qin, "Closed-loop subspace identification: an orthogonal projection approach," *J. Process Control*, vol. 15, no. 1, 53–66, 2005.

[29] J. Wang and S. J. Qin, "Closed-loop subspace identification using the parity space," *Automatica*, vol. 42, no. 2, 315–320, 2006.

[30] J. Wang, T. Chen, and B. Huang, "Closed-loop identification via output fast sampling," *J. of Process Control*, vol. 14, no. 5, pp. 555–570, 2004.

[31] A Höskuldsson, "PLS regression methods," *J. Chemom.*, vol. 2, pp. 211–228, 1988.

[32] G. A. F Seber, *Multivariate Observations*. New York, NY: John Wiley & Sons, 1984.

[33] E. H. Ruspini, "Numerical methods for fuzzy clustering," *Information Sciences*, vol. 2, no. 3, pp. 319–350, 1970.

[34] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA: Kluwer Academic Publishers, 1981.

[35] —, "A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 2, no. 1, 1–8, 1980.

[36] D. E. Gustafson and W. C. Kessel, "Fuzzy Clustering with a fuzzy covariance matrix," in *Proc. of the IEEE Conf. on Decision and Control*, 1979, pp. 761–766.

[37] J. Bezdek, C. Coray, R. Gunderson, and J. Watson, "Detection and Characterization of Cluster Substructure I. Linear Structure: Fuzzy *c*-Lines," *SIAM J. Appl. Math.*, vol. 40, no. 2, 339–357, 1981.

[38] R. Krishnapuram and C. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognit.*, vol. 25, no. 4, pp. 385–400, 1992.

[39] H. Frigui and R. Krishnapuram, "Clustering by competitive agglomeration," *Pattern Recognit.*, vol. 30, no. 7, pp. 1109–1119, 1997.

[40] I. Gath and A. Geva, "Unsupervised Optimal Fuzzy Clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pp. 773–780, 1989.

[41] X. L. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 841–847, 1991.

[42] P. Nelson, J. MacGregor, and P. Taylor, "Missing data methods in PCA and PLS: Score calculations with incomplete observations," *Chemom. & Intell. Lab. Syst.*, vol. 35, pp. 45–65, 1996.

[43] H. S. Fogler, *Elements of Chemical Reaction Engineering*, Fourth. New York, NY: Prentice Hall, 2006.

[44] S. A. Russell, D. G. Robertson, J. H. Lee, and B. A. Ogunnaike, "Control of product quality for batch nylon-6,6 autoclaves," *Chem. Eng. Sci.*, vol. 53, no. 21, pp. 3685–3702, 1998.

[45] M. Joly and J. Pinto, "Optimal control of product quality for batch nylon-6,6 autoclaves," *Chem. Eng. J.*, no. 97, pp. 87–101, 2004.

[46] Y. Yabuki and J. F. MacGregor, "Product quality control in semibatch reactors using midcourse correction policies," *Ind. & Eng. Chem. Res.*, vol. 36, no. 4, pp. 1268–1275, 1997.

CHAPTER 4

Adding Online Learning Ability to the Data-based Modelling Approach

The results in this chapter have been submitted to the following:

JOURNAL PAPERS:

[1] S. Aumi and P. Mhaskar, "Adaptive data-based model predictive control of batch systems," *Ind. & Eng. Chem. Res.*, 2011, (submitted).

REFEREED CONFERENCE PROCEEDINGS:

[2] S. Aumi and P. Mhaskar, "Adaptive data-based model predictive control of batch systems," in *Proc. of the American Control Conf.*, (accepted), Montréal, QC, 2012.

4.1 INTRODUCTION

In the previous chapter, we proposed a multi-model approach that unifies the concepts of auto-regressive exogenous (ARX) modelling, latent variable regression techniques, fuzzy c-means clustering, and multiple local linear models. While this modelling approach is capable of capturing nonlinear process dynamics, it does not explicitly account for *time-varying* dynamics. As a result, in this chapter we generalize the modelling approach to incorporate the ability to capture time-varying dynamics by using information available from new operating conditions immediately (instead of waiting until batch termination to update the model).

In general, adding an adaptive element to model-based control designs wherein the model's parameters are updated online at each sampling instant has been a popular method for improving closed-loop performance. The existing contributions in adaptive, model-based control for batch processes can be broadly divided according to the type of model, deterministic [3–8] or empirical [9–12], being adapted. Updating a deterministic model consists of estimating a subset of the uncertain (possibly time-varying) parameters in the state-space model whereas with empirical models, the entire set of model parameters is typically updated. In either case, the key to success of using an adaptive model is a well-designed real-time, recursive parameter estimation algorithm.

For nonlinear state-space models, a nonlinear estimator, such as an extended or unscented Kalman filter, is typically necessary. In most cases, since a state estimator is used in conjunction with the state-space model, the parameter estimation problem is embedded in the state estimation by augmenting the state vector with a vector of uncertain parameters. This calls for defining a dynamic model for each parameter; however, this is usually unknown (note that in some instances, a dynamic model can be hypothesized for some of the parameters using process knowledge [7]) and a random walk model (to approximate time-varying parameters) is therefore assumed. Using an adaptive input parameterization technique has been another popular approach for improving closed-loop performance, particularly for bio-processes [8]. In this approach, a nominal solution of the MPC problem is first obtained offline (based on the deterministic model) and then subsequently characterized in terms of the qualitative behaviour of the specific growth and production rates. Using available measurements, the growth and productions rates are updated online (using Kalman filters or Luenberger type observers) thereby resulting in an update of the pre-characterized optimal solution, which, in turn, makes the controller adaptive.

The recursive least squares (RLS) algorithm (a linear estimator) is most commonly used for updating empirical model parameters due to the model's assumed linear form (with respect to the parameters). One drawback of conventional RLS algorithms is an inherent assumption of static/stationary model parameters. The RLS framework can straightforwardly

accommodate a random walk model for handling time-varying parameters, but the more popular method to accomplish this has been to discount past data in an exponentially weighted manner with a **forgetting factor**. The use of a forgetting factor (or, in fact, any mechanism to account for time-varying dynamics) is particularly important for maintaining the validity of empirical models of batch processes. In particular, because of using a linear model (or an appropriate combination of linear models) to describe the inherently nonlinear process, new operating conditions can be encountered during a batch around which the model is highly inaccurate or completely invalid, specifically if such conditions were absent from the "training" data set. By updating the model more aggressively based on the current operating conditions, the newly encountered local dynamics can be modelled to some extent, helping to preserve the model's accuracy and validity.

Motivated by the above considerations, in this work, we add online learning ability to our previously developed modelling approach and use the resulting adaptive model in a trajectory tracking predictive controller. Two algorithms are used: (1) the standard RLS algorithm with a forgetting factor and (2) a probabilistic RLS (PRLS) algorithm (also with a forgetting factor) specifically developed for the modelling approach. The rest of this chapter is organized as follows: We begin by showing how the standard RLS algorithm can be applied in a straightforward manner to update the ARX model coefficients of the local linear models. This is followed by the development of a PRLS estimator for each local model that takes each model's probability of being representative of the current plant dynamics into account during the update. Simulation results of the nylon-6,6 batch polymerization process are then presented. Specifically, we take its models that were developed in Section 3.5.2 in Chapter 3 and make them adaptive using the RLS and PRLS algorithms. Subsequently, we demonstrate the improved closed-loop performance achieved from using the adaptive model (over a non-adaptive model) in the trajectory tracking predictive controller. Finally, we summarize our results.

4.2 ONLINE ESTIMATION OF THE DATA-BASED MODEL PARAMETERS

Recall that in the modelling approach in the previous chapter, the final model for output i, y_i, took the following form:

$$\hat{y}_i[k] = \sum_{\ell=1}^{L} \omega_\ell[k] \left(\sum_{j=1}^{n_y} \boldsymbol{\alpha}'_{\ell,j} \boldsymbol{y}[k-j] + \sum_{j=1}^{n_u} \boldsymbol{\beta}'_{\ell,j} \boldsymbol{u}[k-j] + \gamma_\ell \right) \tag{4.1}$$

$$= \sum_{\ell=1}^{L} \omega_\ell[k] \boldsymbol{\theta}'_\ell \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} \tag{4.2}$$

where $\bar{\boldsymbol{x}}[k]$ is defined as before (a vector of lagged concatenated outputs and inputs with n_y and n_u lags, respectively) and $\omega_\ell[k]$ is the (normalized) weight given to model ℓ:

$$\omega_\ell[k] = \frac{\|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{\sum_{\ell=1}^{L} \|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}$$

The vectors, $\boldsymbol{\alpha}_{\ell,j}$ and $\boldsymbol{\beta}_{\ell,j}$, and the scalar, $\boldsymbol{\gamma}_\ell$, define the ℓ-th local ARX model while the vector $\boldsymbol{\theta}_\ell$ stores ℓ-th model's coefficients. Using the following definitions,

$$\boldsymbol{\Theta} := \begin{bmatrix} \boldsymbol{\theta}_1' & \cdots & \boldsymbol{\theta}_\ell' & \cdots & \boldsymbol{\theta}_L' \end{bmatrix}' \tag{4.3}$$

$$\boldsymbol{\psi}[k] := \begin{bmatrix} \omega_1[k] \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} & \cdots & \omega_\ell[k] \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} & \cdots & \omega_L[k] \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} \end{bmatrix}' \tag{4.4}$$

Equation (4.1) can be rewritten in the common least squares vector form:

$$\hat{y}_i[k] = \boldsymbol{\psi}'[k]\boldsymbol{\Theta} \tag{4.5}$$

The parameters to be identified in the final model form are the:

1. Cluster centre points, $\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_\ell$, that define the weighting function
2. ARX model coefficients, $\boldsymbol{\Theta}$

One way to adapt this model is where both sets of parameters are updated at every sampling instance using the plant measurements. In this work, we look at a subset of this problem by assuming that the originally computed cluster centre points hold for the current batch, and we focus on updating only the ARX model coefficients during the batch. We deem this assumption reasonable because the initially identified cluster centre points span the range of operating conditions in previous batches, making it unlikely for a new batch to encounter operating conditions that would significantly change the centre points. Note that once the current batch is finished, its data can be added to the existing database, and the cluster centres along with the ARX model coefficients can be updated.

In this section, we address the problem of how to recursively update the ARX model coefficients during a batch as measurements become available. We start by demonstrating that the standard RLS algorithm is a natural solution for this problem and bring attention to a few issues regarding its implementation. Next, we develop a probabilistic RLS (PRLS) algorithm specific to the multi-model approach that adopts a localized, probabilistic approach to the model updates.

4.2.1 Recursive Least Squares Parameter Estimation

Consider the scenario where OLS regression or PCR has been used to estimate the ARX model coefficients and their co-variances, denoted henceforth by $\boldsymbol{\Theta}[0]$ and $\mathbf{P}[0]$ (respectively). The ARX model coefficient estimates can be updated based on the error between the predicted output and the plant measurement by rearranging the regression solution in a recursive form, yielding the standard recursive least squares (RLS) algorithm [13]. The updated estimates given by RLS are equivalent to those that would be obtained from OLS/PCR after appending the response and regressor matrices with new rows corresponding to the current measurement. In this way, the old training data is given just as much importance as the new data. With limited training data (as is the case for batch processes), operating conditions can be encountered during a new batch that are not originally modelled. Under these circumstances, it is desirable to update the models more aggressively based on the new plant data.

One extension of RLS to cover this situation is to replace the conventional least squares criterion with one having time-varying weighting of the data as shown below.

$$V(\boldsymbol{\Theta}, N) = \frac{1}{2\sigma_y^2} \sum_{j=1}^{N} \lambda^{N-i} \left\{ y_i[j] - \boldsymbol{\psi}'[j]\boldsymbol{\Theta} \right\}^2 \tag{4.6}$$

where N is the number of observations, σ_y^2 is the measurement noise variance, and the scalar, $0 < \lambda \leq 1$, is a **forgetting factor**. With the inclusion of λ, the most recent data is given unit weight while data that is n time units old is weighted down by λ^n. The effective memory length of the data (i.e., the number of observations used at any sampling instance for the update), N_t, is given by [13]:

$$N_t = \frac{1}{1-\lambda} \tag{4.7}$$

The set of recursive equations shown in Equations (4.8a) to (4.8d) for updating $\boldsymbol{\Theta}[k]$ and its co-variance, $\mathbf{P}[k]$, can be derived for the least squares criterion in Equation (4.6) [13]. Note that by starting from a time-varying criterion, the inherent assumption of static model parameters associated with RLS without a forgetting factor is removed. This becomes important if the process has time-varying uncertainties.

$$v[k] = y_i[k] - \boldsymbol{\psi}'[k]\boldsymbol{\Theta}[k-1] \tag{4.8a}$$

$$\boldsymbol{\Theta}[k] = \boldsymbol{\Theta}[k-1] + \boldsymbol{k}[k]v[k] \tag{4.8b}$$

$$\boldsymbol{k}[k] = \frac{\mathbf{P}[k-1]\boldsymbol{\psi}[k]}{\boldsymbol{\psi}'[k]\mathbf{P}[k-1]\boldsymbol{\psi}[k] + \lambda\sigma_y^2} \tag{4.8c}$$

$$\mathbf{P}[k] = \frac{1}{\lambda}\left\{\mathbf{I} - \boldsymbol{k}[k]\boldsymbol{\psi}'[k]\right\}\mathbf{P}[k-1] \tag{4.8d}$$

In this set of recursive equations, $v[k]$ is the prediction error or innovation, $y_i[k]$ is the plant measurement of output i, and $\boldsymbol{k}[k]$ is a gain vector. The initial estimate and co-variance can be initialized at the solution of the original regression problem, $\boldsymbol{\Theta}[0]$ and $\mathbf{P}[0]$. For $\lambda = 1$, Equations (4.8b) to (4.8d) reduce to the standard RLS algorithm, and the effective memory length of the data is ∞, signifying all the data is used with equal weight. Additionally, Equation (4.8d) indicates that the estimate co-variance always decreases as the batch progresses for $\lambda = 1$. This can be understood as the level of uncertainty in the estimates decreasing with time as more information (i.e., the inverse of uncertainty) is being used for the estimation.

When $\lambda < 1$, as new data becomes available, old data is continuously discounted. While this behaviour is necessary for adapting more aggressively based on new data, it becomes problematic when there is no information in the new data. This is referred to as periods of low excitation. During low excitation periods, data loss occurs since old data is discounted while the new data has no information. Consequently, the co-variance matrix elements can increase unboundedly and become ill-conditioned, leading to the estimates "blowing-up" or "bursting". In a closed-loop environment, periods of low excitation can occur when the controller begins to track constant or slowly varying set-points during which the inputs and outputs show little variation between successive sampling instances and no information is obtainable from the plant data while the old (useful) data continues to be discounted. A common practice to keep the plant persistently excited has been to add small dither signals on top of the inputs that do not significantly affect the closed-loop behaviour.

Low excitation periods are less common in batch processes, particularly when considering the trajectory tracking problem, compared to continuous processes. Recall that the set-point trajectories for batch processes are typically computed offline or regenerated periodically during the batch by solving a dynamic optimization problem. These set-point trajectories tend to vary throughout the batch duration while covering a wide range of operating conditions. The inputs are continually adjusted in response, keeping the process more or less persistently excited. The flexibility to use complex, time-varying set-point trajectories is, in fact, one of the main reasons for the popularity of batch processing. Nevertheless, there may be periods when the set-point trajectories plateau specifically during the initial and/or finishing stages of the batch. There have been many techniques proposed to handle the effects of low excitation (see [13, Chapter 11] for a review). Two approaches that require limited modifications to the RLS framework include using a variable forgetting factor [14] such that the information content in the filter remains constant or the constant trace algorithm [13, Chapter 11] wherein the co-variance matrix is explicitly bounded by scaling it at each iteration such that its trace remains constant.

4.2.2 Probabilistic Recursive Least Squares Parameter Estimation

The standard RLS algorithm with a forgetting factor in Equations (4.8a) to (4.8d) simultaneously updates all of the local ARX model coefficients based on the prediction error. Suppose that at a given sampling instance, model ℓ is a substantially better representation of the process dynamics compared to the other models. In this situation, model ℓ should be updated to a greater extent than the other models once the measurement becomes available. However, with the standard RLS algorithm, models known to be a poor representation of the current dynamics can be unnecessarily updated. This motivates the idea of adopting a more local update approach. To this end, we develop the probabilistic recursive least squares (PRLS) algorithm for each model that takes the probability of the plant measurement originating from the different models into account.

Denote the original ARX model coefficient estimates and their co-variance for the ℓ-th model as $\boldsymbol{\theta}_\ell[0]$ and $\mathbf{P}_\ell[0]$ (respectively). In a local update approach, L estimators are operated independently of each other (i.e., each estimator only considers its corresponding model). As a result, the following L events are mutually exclusive in each estimator:

$$\mathcal{E}_\ell : y_i[k] \text{ originated from plant dynamics representable by model } \ell, \quad \text{for } \ell \in [1, L] \tag{4.9}$$

For the development of the PRLS equations, we first introduce the following assumptions.

Assumption 4.1: There is negligible prediction error from the originally identified model such that:

$$\mathbf{E}\{y_i[k]\} = \hat{y}_i[k] \tag{4.10}$$

where $\mathbf{E}\{\cdot\}$ denotes the expected value and $\hat{y}_i[k]$ is the model prediction of the plant measurement, $y_i[k]$, from Equation (4.1).

Assumption 4.2: At a given sampling instance, the plant dynamics are representable by a linear combination of the L local linear models in Equation (4.1). Thus, the events in Equation (4.9) are exhaustive (in addition to being mutually exclusive in each estimator):

$$\sum_{\ell=1}^{L} \Pr\{\mathcal{E}_\ell\} = 1$$

where $\Pr\{\cdot\}$ denotes the probability of an event.

As an initial step, we compute the posterior probability of $y_i[k]$ originating from model ℓ using a Bayesian approach. This is formalized below in Theorem 4.1.

Theorem 4.1: *Consider L estimators (corresponding to the L models) operating independently and subject to Assumptions 4.1 to 4.2. The posterior probability, $\zeta_\ell[k]$, that the plant measurement, $y_i[k]$, originated from plant dynamics representable by model ℓ is given by:*

$$\zeta_\ell[k] = \frac{\mathcal{N}\left(y_i[k];\, \hat{y}_{i,\ell}[k], \sigma^2_\ell[k]\right) \|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{\sum_{j=1}^{L} \mathcal{N}\left(y_i[k];\, \hat{y}_{i,j}[k], \sigma^2_j[k]\right) \|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_j\|^{-2}} \tag{4.11}$$

where $\mathcal{N}\left(x;\, \mu, \sigma^2\right)$ represents the value of x on a normal distribution with mean μ and variance σ^2, and $\hat{y}_{i,\ell}[k]$ and $\sigma^2_\ell[k]$ are given by:

$$\hat{y}_{i,\ell}[k] = \bar{\boldsymbol{x}}'[k]\boldsymbol{\theta}_\ell[k-1] \tag{4.12}$$

$$\sigma^2_\ell[k] = \bar{\boldsymbol{x}}'[k]\mathbf{P}_\ell[k-1]\bar{\boldsymbol{x}}[k] + \sigma^2_y \tag{4.13}$$

Proof: Using the total probability theorem with respect to the events in Equation (4.9), the expected value of the plant measurement at sampling instance k, given measurements up to $k-1$, can be written as:

$$\mathbf{E}\left\{y_i[k] \mid Y_i^{[k-1]}\right\} = \sum_{\ell=1}^{L} \mathbf{E}\left\{y_i[k] \mid \mathcal{E}_\ell, Y_i^{[k-1]}\right\} \Pr\left\{\mathcal{E}_\ell \mid Y_i^{[k-1]}\right\} \tag{4.14}$$

where $Y_i^{[k-1]}$ denotes the measurement sequence up to $k-1$. We have used the fact that the events in Equation (4.9) are mutually exclusive and exhaustive. Note that this follows from the total probability theorem, which, in general, can be stated as:

$$\mathbf{E}\{x\} = \sum_{i=1}^{L} \mathbf{E}\{x \mid \mathcal{E}_i\} \Pr\{\mathcal{E}_i\}$$

where x is a continuous random variable and $\{\mathcal{E}_1, \ldots, \mathcal{E}_L\}$ are mutually exclusive and exhaustive events.

With no prediction error (Assumption 4.1), the expected value of the plant measurement, conditioned on the event that the ℓ-th model represents the plant dynamics, follows from Equation (4.1) with $\omega_\ell[k] = 1$ and $\omega_j[k] = 0$ for $j \neq \ell$ where $j \in [1, L]$.

$$\mathbf{E}\left\{y_i[k] \mid \mathcal{E}_\ell, Y_i^{[k-1]}\right\} = \bar{\boldsymbol{x}}'[k]\boldsymbol{\theta}_\ell[k-1]$$

Using this result and Equation (4.10), which implies $\mathbf{E}\left\{y_i[k] \mid Y_i^{[k-1]}\right\} = \hat{y}_i[k]$, Equation (4.14) can be expressed as:

$$\hat{y}_i[k] = \sum_{\ell=1}^{L} \bar{\boldsymbol{x}}'[k]\boldsymbol{\theta}_\ell[k-1] \Pr\left\{\mathcal{E}_\ell \mid Y_i^{[k-1]}\right\} \tag{4.15}$$

Comparing Equation (4.1) and Equation (4.15), we have:

$$\Pr\left\{\mathcal{E}_\ell \mid Y^{[k-1]}\right\} = \omega_\ell[k] = \frac{\|\tilde{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{\sum_{j=1}^{L} \|\tilde{\boldsymbol{x}}[k] - \boldsymbol{c}_j\|^{-2}} \tag{4.16}$$

Thus, the *prior* probability of $\mathcal{E}_\ell$ (based on information up to $k-1$) can be estimated with model ℓ's weight (i.e., its membership function value). The prior probability can be corrected once the plant measurement becomes available using Bayes' rule. Denoting this posterior probability as $\zeta_\ell[k]$, the application of Bayes' rule yields:

$$\begin{aligned}\zeta_\ell[k] := \Pr\left\{\mathcal{E}_\ell \mid y_i[k], Y_i^{[k-1]}\right\} &= \frac{p\left(y_i[k] \mid Y_i^{[k-1]}, \mathcal{E}_\ell\right) \Pr\left\{\mathcal{E}_\ell \mid Y_i^{[k-1]}\right\}}{p\left(y_i[k] \mid Y_i^{[k-1]}\right)} \\ &= \frac{p\left(y_i[k] \mid Y_i^{[k-1]}, \mathcal{E}_\ell\right) \|\tilde{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{p\left(y_i[k] \mid Y_i^{[k-1]}\right) \sum_{j=1}^{L} \|\tilde{\boldsymbol{x}}[k] - \boldsymbol{c}_j\|^{-2}}\end{aligned} \tag{4.17}$$

where $Y_i^{[k]}$ has been partitioned as $\left\{y_i[k], Y_i^{[k-1]}\right\}$ and $p\left(y_i[k] \mid Y_i^{[k-1]}, \mathcal{E}_\ell\right)$ is the prior likelihood of the measurement conditioned on $\mathcal{E}_\ell$. This conditional likelihood is simply the measurement's height on the probability density function (pdf) of model ℓ's prediction. With zero-mean Gaussian measurement noise, the pdf of the prediction is a normal distribution with mean $\hat{y}_{i,\ell}[k]$ and variance $\sigma_\ell^2[k]$. These pdf parameters are given by Equation (4.12) and Equation (4.13), respectively. The mean is simply the prediction from only using model ℓ (i.e., the pdf is centred at the model's prediction). The variance equation follows from the assumed linear model form and its derivation procedure is identical to the one for the innovation co-variance in the Kalman filter. Thus, $p\left(y_i[k] \mid Y_i^{[k-1]}, \mathcal{E}_\ell\right)$ is given by:

$$p\left(y_i[k] \mid Y_i^{[k-1]}, \mathcal{E}_\ell\right) = \mathcal{N}\left(y_i[k]; \hat{y}_{i,\ell}[k], \sigma_\ell^2[k]\right) \tag{4.18}$$

The denominator in Equation (4.17) can be rewritten by invoking the total probability theorem with respect to the events in Equation (4.9) and using the results in Equation (4.16) and Equation (4.18):

$$\begin{aligned}p\left(y_i[k] \mid Y_i^{[k-1]}\right) &= \sum_{j=1}^{L} p\left(y_i[k] \mid Y_i^{[k-1]}, \mathcal{E}_j\right) \Pr\left\{\mathcal{E}_j \mid Y_i^{[k-1]}\right\} \\ &= \frac{\sum_{j=1}^{L} \mathcal{N}\left(y_i[k]; \hat{y}_{i,j}[k], \sigma_{i,j}^2[k]\right) \|\tilde{\boldsymbol{x}}[k] - \boldsymbol{c}_j\|^{-2}}{\sum_{s=1}^{L} \|\tilde{\boldsymbol{x}}[k] - \boldsymbol{c}_s\|^{-2}}\end{aligned} \tag{4.19}$$

Substituting Equation (4.18) and Equation (4.19) in Equation (4.17) yields the final expression for the posterior probability in the theorem, completing the proof.

A standard assumption of zero-mean, normally distributed measurement/sensor noise is made for the result in Theorem 4.1, but its key idea holds even when this is not the case. If the noise distribution's parameters are known, the likelihood expressions derived in Equation (4.18) and Equation (4.19) can be modified accordingly. In the absence of any information about the noise characteristics, the assumption of zero-mean, normally distributed noise is a meaningful way to determine the likelihoods that are required for the result in the theorem.

In this theorem, we use the model weights (obtained from the membership function) as the prior probabilities when applying Bayes' rule. To rigorously show the equivalence between the prior probabilities and model weights, an assumption of negligible prediction error is made. In practice, this assumption will of course not hold because of unavoidable plant-model mismatch. Hence, the weights essentially represent an *estimate* of the prior probabilities with the quality of the estimate depending on the model quality. Note also that the prior probabilities are corrected to some extent once the measurement becomes available from Bayes' rule, and the level of plant-model mismatch can also be reduced as the model is continually updated online. As presented in the sequel, if the initial model built using the multi-model approach is of good quality, the result of this theorem provides a mechanism to update the individual models within the RLS framework in a probabilistic sense. This is formalized below in Theorem 4.2.

Theorem 4.2: *Consider the problem of probabilistically updating each model's coefficients (using independent estimators) at sampling instance k when we have y_i and $\boldsymbol{u}$ measurements up to k and $k-1$, respectively. The coefficients for model ℓ that minimize the time-varying least squares criterion in Equation (4.6) in a probabilistic sense at k are given by:*

$$v_\ell[k] = y_i[k] - \bar{\boldsymbol{x}}'[k]\boldsymbol{\theta}_\ell[k-1] \tag{4.20a}$$

$$\boldsymbol{\theta}_\ell[k] = \boldsymbol{\theta}_\ell[k-1] + \zeta_\ell[k]\boldsymbol{k}_\ell[k]v_\ell[k] \tag{4.20b}$$

$$\boldsymbol{k}_\ell[k] = \frac{\mathbf{P}_\ell[k-1]\bar{\boldsymbol{x}}[k]}{\bar{\boldsymbol{x}}'[k]\mathbf{P}_\ell[k-1]\bar{\boldsymbol{x}}[k] + \lambda\sigma_y^2} \tag{4.20c}$$

$$\mathbf{P}_\ell[k] = \mathbf{P}_\ell[k-1]\left\{1-\zeta_\ell[k]\right\} + \frac{\zeta_\ell[k]}{\lambda}\left\{\mathbf{I} - \boldsymbol{k}_\ell[k]\bar{\boldsymbol{x}}'[k]\right\}\mathbf{P}_\ell[k-1] + \tilde{\mathbf{P}}_\ell[k] \tag{4.20d}$$

where $\tilde{\mathbf{P}}_\ell[k] = \boldsymbol{k}_\ell[k]\left\{\zeta_\ell[k]v_\ell^2[k] - \zeta_\ell[k]^2 v_\ell^2[k]\right\}\boldsymbol{k}_\ell[k]'$.

Proof: Note that this proof is a modification of the probabilistic data association filter derivation in [15]. Given the plant measurement, $y_i[k]$, the following complementary, mutually exclusive, and exhaustive events are possible in the ℓ-th estimator:

$$\begin{aligned}&\mathcal{E}_\ell : y_i[k] \text{ originated from plant dynamics representable by model } \ell\\ &\mathcal{E}_0 : y_i[k] \text{ did not originate from plant dynamics representable by model } \ell\end{aligned} \tag{4.21}$$

Next, we derive the recursive equations for the coefficient estimate and co-variance updates that take the posterior probabilities for each model into account. For the events in Equation (4.21), the estimate of model ℓ's coefficients can be expressed using the total probability theorem as follows:

$$\begin{aligned}\boldsymbol{\theta}_\ell[k] &= \mathbf{E}\left\{\boldsymbol{\theta}_\ell[k] \mid Y_i^{[k]}\right\} \\ &= \mathbf{E}\left\{\boldsymbol{\theta}_\ell[k] \mid \mathcal{E}_0, Y_i^{[k]}\right\} \Pr\left\{\mathcal{E}_0 \mid Y_i^{[k]}\right\} + \mathbf{E}\left\{\boldsymbol{\theta}_\ell[k] \mid \mathcal{E}_\ell, Y_i^{[k]}\right\} \Pr\left\{\mathcal{E}_\ell \mid Y_i^{[k]}\right\} \\ &= \boldsymbol{\theta}_\ell^{(0)}[k]\left\{1-\zeta_\ell[k]\right\} + \boldsymbol{\theta}_\ell^{(\ell)}[k]\zeta_\ell[k] \end{aligned} \tag{4.22}$$

where $\zeta_\ell[k]$ is given by Equation (4.11) and $\boldsymbol{\theta}_\ell^{(i)}[k]$ denotes the coefficient estimates at k conditioned on $\mathcal{E}_i$. For $\mathcal{E}_0$, the model coefficient estimates and their co-variance should remain unchanged from their previous estimates since the plant dynamics represented by model ℓ did not play a role in generating the measurement. Accordingly, we have:

$$\begin{aligned}\boldsymbol{\theta}_\ell^{(0)}[k] &= \boldsymbol{\theta}_\ell[k-1] \\ \mathbf{P}_\ell^{(0)}[k] &= \mathbf{P}_\ell[k-1]\end{aligned}$$

where $\mathbf{P}_\ell^{(i)}[k]$, denotes the co-variance of $\boldsymbol{\theta}_\ell^{(i)}[k]$ (i.e., the co-variance conditioned on $\mathcal{E}_i$). For $\mathcal{E}_\ell$, the standard RLS equations with a forgetting factor, written specifically for model ℓ, can be used to update model ℓ's coefficients and their co-variance:

$$\begin{aligned}\nu_\ell[k] &= y_i[k] - \bar{\boldsymbol{x}}'[k]\boldsymbol{\theta}_\ell[k-1] \\ \boldsymbol{\theta}_\ell^{(\ell)}[k] &= \boldsymbol{\theta}_\ell[k-1] + \boldsymbol{k}_\ell[k]\nu_\ell[k] \\ \boldsymbol{k}_\ell[k] &= \frac{\mathbf{P}_\ell[k-1]\bar{\boldsymbol{x}}[k]}{\bar{\boldsymbol{x}}'[k]\mathbf{P}_\ell[k-1]\bar{\boldsymbol{x}}[k] + \lambda\sigma_y^2} \\ \mathbf{P}_\ell^{(\ell)}[k] &= \frac{1}{\lambda}\left\{\mathbf{I} - \boldsymbol{k}_\ell[k]\bar{\boldsymbol{x}}'[k]\right\}\mathbf{P}_\ell[k-1]\end{aligned}$$

Combining the expressions for $\boldsymbol{\theta}_\ell^{(0)}[k]$ and $\boldsymbol{\theta}_\ell^{(\ell)}[k]$ into Equation (4.22) yields the update equation in the theorem (Equation (4.20b)). Equation (4.22) shows that the final estimate is a weighted sum of 2 conditional estimates. In this case, the final estimate's co-variance is a weighted sum of the two conditional co-variances, $\mathbf{P}_\ell^{(0)}[k-1]$ and $\mathbf{P}_\ell^{(\ell)}[k-1]$, and an additional co-variance term, $\tilde{\mathbf{P}}_\ell[k]$, arising from the measurement origin uncertainty. This term, given by $\tilde{\mathbf{P}}_\ell[k] = \boldsymbol{k}_\ell[k]\left\{\zeta_\ell[k]\nu_\ell^2[k] - \zeta_\ell[k]^2\nu_\ell^2[k]\right\}\boldsymbol{k}_\ell[k]'$, is similar to the "spread of the means" term in the co-variance of a random variable that is the sum of two random variables. Its lengthy derivation is available in existing literature [15] and omitted here for brevity. The final co-variance update equation is given by:

$$\mathbf{P}_\ell[k] = \mathbf{P}_\ell[k-1]\left\{1-\zeta_\ell[k]\right\} + \frac{\zeta_\ell[k]}{\lambda}\left\{\mathbf{I} - \boldsymbol{k}_\ell[k]\bar{\boldsymbol{x}}[k]\right\}\mathbf{P}_\ell[k-1] + \tilde{\mathbf{P}}_\ell[k]$$

In summary, by weighting the updated model coefficients (and their co-variance) by the probability that they should have been updated and by using the standard RLS equations to compute the updated coefficients, the final coefficient estimates minimize the least squares criterion in Equation (4.6) in the probabilistic sense.

Compared to the standard RLS algorithm, using the PRLS algorithm for updating each model's coefficients provides more flexibility for controlling the adaptation. First, the forgetting factor for each estimator can be tuned independently. More importantly, the local adaptations can be made more aggressive while maintaining more precise coefficient estimates. This is because the co-variance increase from discounting old data is weighted by the (posterior) probability, $\zeta_\ell[k]$. In this way, co-variance increases only occur during the appropriate periods when the local model adaptation is active. With the standard RLS algorithm, the co-variance associated with all the models can increase at every sampling instance whether or not the model should have been updated.

Remark 4.1: In general, no conclusions can be drawn as to which RLS update strategy, conventional or probabilistic, will perform better in practice. The 2 strategies adopt different approaches in updating the models. In the PRLS algorithm, the models are updated independently/locally according to their (posterior) probabilities of being consistent with the current plant dynamics. This approach is more suitable for specific situations than a global update of all model coefficients. For instance, when a batch is operating near a cluster's centre point, it is more meaningful to update the cluster's corresponding model rather than all the models during this period. Note also that due to the probabilistic nature of the update, the PRLS algorithm is still capable of meaningful adaptations when the operating conditions are somewhere between different cluster centre points.

Remark 4.2: One advantage of the proposed PRLS algorithm is that it has lower computational requirements than the standard RLS algorithm (for $L > 1$). Since the standard RLS algorithm for a given output is interpret-able as a Kalman filter for the following system [13]:

$$\begin{aligned}\boldsymbol{\Theta}[k] &= \boldsymbol{\Theta}[k-1] \\ y_i[k] &= \boldsymbol{\psi}'[k]\boldsymbol{\Theta}[k] + v[k]\end{aligned}$$

its computational requirements are approximately proportional to $(L \times n_{\boldsymbol{\theta}_\ell})^3$ where $n_{\boldsymbol{\theta}_\ell}$ is the number of coefficients in each local model. This follows from the fact that the computational requirements of a Kalman filter are proportional to n^3 where $n = \max\{n, p\}$ with n and p denoting the number of states and outputs (respectively) for the system under consideration [15]. By comparison, since each model is updated individually in the PRLS algorithm, its computational requirements are proportional to $\sum_{\ell=1}^{L} n_{\boldsymbol{\theta}_\ell}^3 = L \times n_{\boldsymbol{\theta}_\ell}^3$. Thus, the computational requirements are lower by a factor of L^2, which may be significant for a large L.

4.3 Simulation Example: Nylon-6,6 Batch Polymerization

In this section, we revisit the nylon-6,6 batch polymerization example in Section 3.5.2 in Chapter 3 and investigate the advantages of adding model adaptation. For more details on this process, see Section 3.5.2.

As before, the control objective we considered was to track reference trajectories for the reaction mixture temperature, T (K), and reactor pressure, P (psi), denoted by T_{ref} and P_{ref} (respectively), by manipulating the steam jacket pressure, P_j (psi), and vent rate, v (kg/h). Thus, we had: $\boldsymbol{y} = \begin{bmatrix} T & P \end{bmatrix}'$ and $\boldsymbol{u} = \begin{bmatrix} P_j & v \end{bmatrix}'$. The inputs were constrained between $\boldsymbol{u}_{\min} = \begin{bmatrix} 700 & 0 \end{bmatrix}'$ and $\boldsymbol{u}_{\max} = \begin{bmatrix} 1800 & 2000 \end{bmatrix}'$. The duration of the batch was $t_{\text{end}} = 3$ hours, and it was sampled every 1 minute. The reference trajectories (see Figure 3.6) were assumed to be identified appropriately in some fashion.

4.3.1 Data-based Model Development

The data-based models for T and P developed in the previous chapter were designated to be the non-adaptive models (see Section 3.5.2 for the details on how the artificial database was generated and the model fitting procedure) in this chapter. Table 4.1 shows the key model parameters, namely the lags and number of clusters/models and the final RMSE values. A discussion of these results is available in Section 3.5.2.

Table 4.1: Final lag structures, number of clusters, L, and RMSE values of the data-based models for the nylon-6.6 batch polymerization process

	Lags					
Output	T	P	P_j	v	L	RMSE
T	1	0	1	1	5	1.65
P	0	1	0	1	1	0.18

The outputs from the data-based models are compared with those from a batch in the validation data set in Figure 4.1. Validation batches were chosen (one for each output) for which the data-based models displayed the poorest predictions (highest RMSE values)[1]. The predicted temperature, particularly during the finishing stages, demonstrated the need/room for improvement of the temperature model. This is addressed in the next section by incorporating online learning ability into the model using the RLS and PRLS algorithms. The pressure model, on the other hand, did not require any further improvements.

[1] In Figure 3.7, a random validation batch was chosen for both outputs, explaining the difference between the results in Figure 3.7 and Figure 4.1

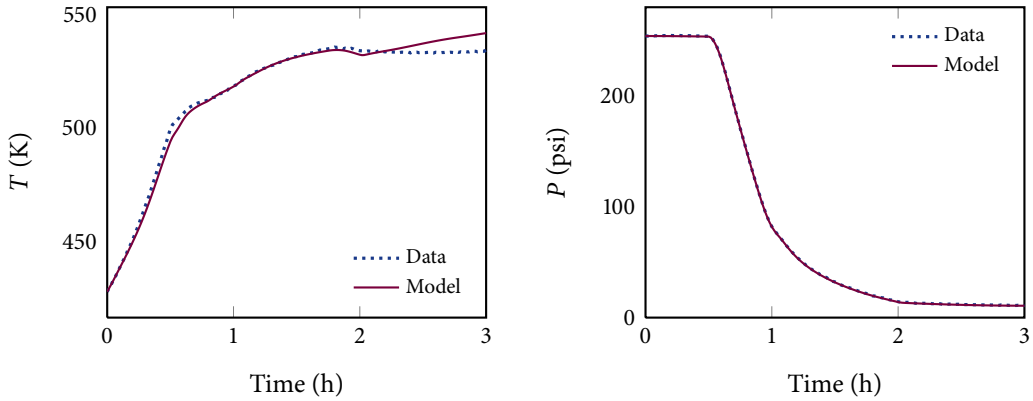


Figure 4.1: Comparison of the data-based models' outputs with the corresponding trajectories in the validation data for the nylon-6,6 batch polymerization process. The database trajectories selected correspond to the highest RMSE values for each output.

4.3.2 Online Update of the Temperature Model

In this section, we use an adaptive temperature model to predict the temperature of the validation batches. The adaptation performance was evaluated according to prediction accuracy and precision of the model coefficient estimates. The prediction accuracy was quantified using the RMSE metric while the precision was quantified using the co-variance matrix as follows. At sampling instance k for batch b, the sum of the variances of the coefficient estimates is trace $\left\{\mathbf{P}^{(b)}[k]\right\}$. The average of this trace over the batch duration and number of batches was termed the mean sum of the variances (MSV) and used as the metric to asses the estimation precision.

$$\text{MSV} = \frac{1}{B}\frac{1}{K}\sum_{b=1}^{B}\sum_{k=1}^{K}\text{trace}\left\{\mathbf{P}^{(b)}[k]\right\}$$

The results from applying the RLS and PRLS algorithms on the temperature model with various forgetting factors are summarized in Table 4.2. The forgetting factor in each PRLS estimator was kept the same for simplicity.

Table 4.2: Performance of the RLS and PRLS algorithms with an adaptive data-based T model of the nylon-6,6 batch polymerization process

	RLS		*PRLS*	
λ	**RMSE**	**MSV**	**RMSE**	**MSV**
1	0.9074	7.861	0.8219	5.657
0.995	0.7927	12.49	0.8073	5.918
0.95	0.4980	3.349×10^3	0.6721	11.57
0.90	0.4399	1.465×10^6	0.5631	50.83

The improved prediction accuracy obtained by adapting the model is evident by comparing the RMSE value in Table 4.1 with those in Table 4.2. Adapting the model with $\lambda = 1$ improved the prediction accuracy by 45% (RLS) and 50% (PRLS). As the forgetting factor was decreased to make the adaptation more aggressive, there was a trade-off between the prediction accuracy and parameter precision. However, the loss in precision was significantly less sensitive for the PRLS algorithm, permitting more aggressive adaptation (lower values of λ) with acceptable parameter variances. Thus, the PRLS algorithm was concluded to offer a better management of the trade-off between the prediction accuracy and parameter precision. The temperature prediction error magnitude, $|\hat{T} - T|$, for the batch in Figure 4.1 is shown in Figure 4.2 before and after incorporating model adaptation. Forgetting factors of 0.995 and 0.90 for the RLS and PRLS estimators (respectively) were selected for a fair comparison since they resulted in comparable estimation precision. These forgetting factors were also used for adapting the model in the closed-loop simulations.

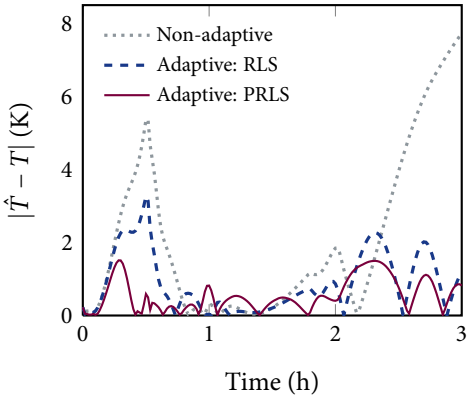


Figure 4.2: Prediction error magnitudes (for the batch in Figure 4.1) for the data-based T model of the nylon-6,6 batch polymerization process

4.3.3 Closed-loop Results

Closed-loop simulation results of a trajectory tracking predictive controller that uses the *adaptive* data-based temperature model as its underlying model are presented in this section.

The temperature model is updated at each sampling instance prior to the controller computations, making the controller adaptive. The inputs at each sampling instance are computed by solving the optimization problem presented earlier in Equations (3.26a) to (3.26b). The controller was tuned for the non-adaptive model and left unchanged for the adaptive cases to avoid confounding the results with tuning. The tuning parameters were set as follows: $\boldsymbol{\Xi} = \text{diag}\{20, 50\}$, $\boldsymbol{\Pi} = \text{diag}\{0.05, 0.05\}$, and $P = 10$.

The performance of the adaptive predictive controller was compared against the non-adaptive version when encountering disturbances in the initial conditions. Note that the superior tracking ability of the non-adaptive version of the design over existing trajectory tracking approaches, namely PI control and latent variable MPC [16], have already been established in Chapter 3. Thus, in this section, we are mainly concerned with highlighting the potential gains in closed-loop performance from including model adaptations.

Two initial conditions, 1 within and 1 outside the range of initial conditions in the training data, were considered for the simulations. The metric used to assess tracking performance was the ITAE. The closed-loop performance is summarized in Table 4.3 and Table 4.4 for both initial conditions.

Table 4.3: Tracking performance with the proposed MPC design for initial conditions within the training data range for the nylon-6,6 batch polymerization process

	Adaptation algorithm		
	None	**RLS** ($\lambda = 0.995$)	**PRLS** ($\lambda = 0.95$)
Temperature ITAE:	2.825	1.108	1.034
Pressure ITAE:	4.146	2.805	1.357

Table 4.4: Tracking performance with the proposed MPC design for initial conditions outside the training data range for the nylon-6,6 batch polymerization process

	Adaptation algorithm		
	None	**RLS** ($\lambda = 0.995$)	**PRLS** ($\lambda = 0.95$)
Temperature ITAE:	30.70	1.676	1.211
Pressure ITAE:	29.20	2.881	1.474

Focusing first on the case when the initial conditions were within the training data range, incorporating standard RLS with $\lambda = 0.995$ offered improvements of 61% and 32% in temperature and pressure tracking, respectively. The temperature and pressure tracking were further improved by by 7% and 50% (respectively) when the PRLS algorithm was used. These results illustrate that considerably better tracking can be achieved by using an adaptive model in the controller.

From the results in Table 4.4, model adaptation was crucial for achieving acceptable closed-loop performance for large disturbances in the initial conditions (i.e., when the initial conditions were outside the training data range). The tracking errors and input profiles for these initial conditions are shown in Figure 4.3.

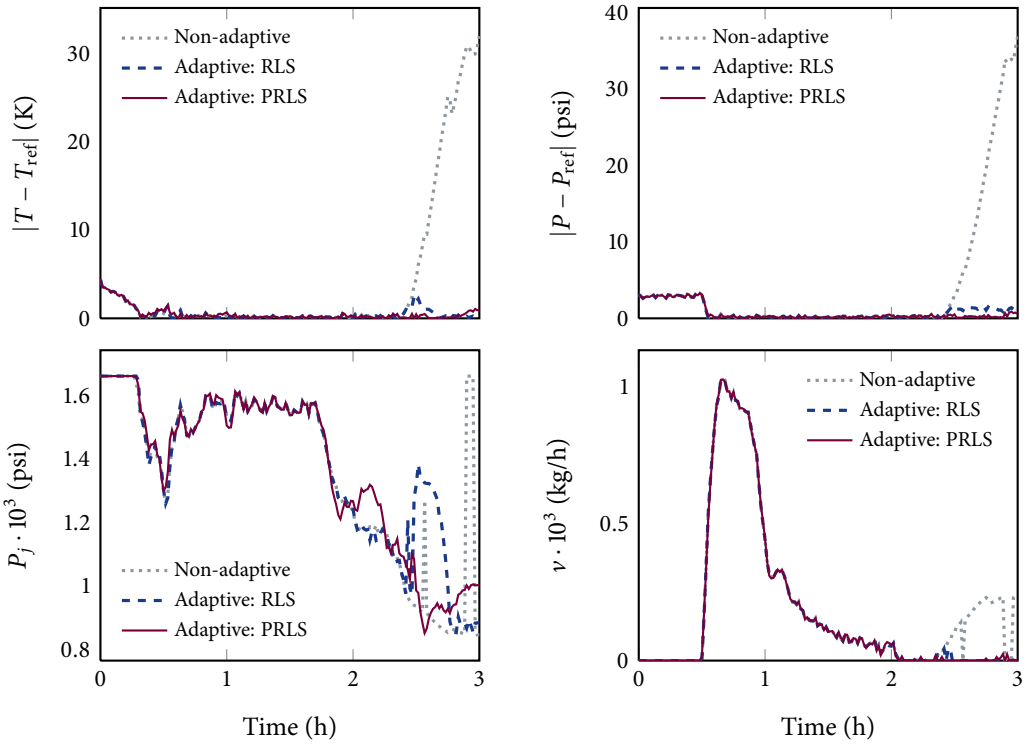


Figure 4.3: Tracking error and input profiles with the proposed MPC design (for initial conditions outside the training data range) for the nylon-6,6 batch polymerization process

There was substantial deviation from the reference trajectories during the finishing stages of the batch with the non-adaptive controller. With these initial conditions, foreign operating conditions (and therefore dynamics) were likely encountered that were not originally modelled by the temperature model. This led to poor temperature predictions in the controller calculations, and the coupled nature of the control problem led to poor tracking for both outputs. The adaptive designs, on the other hand, were able to learn the new dynamics using the plant measurements, leading to more accurate predictions and significantly improved tracking performance. While there is no general guarantee that the PRLS algorithm will always outperform standard RLS, among the two adaptation approaches, using the PRLS algorithm produced lower ITAE values for both outputs consistently for this simulation example.

4.4 CONCLUSIONS

In this chapter, online learning ability was integrated within a previously developed data-based modelling methodology for batch processes. First, it was demonstrated how the standard RLS algorithm with a forgetting factor can be applied in a straightforward manner to provide online updates of the model parameters. Next, a probabilistic RLS (PRLS) estimator (also with a forgetting factor) was developed that updated each model individually according to its probability of being representative of the local dynamics. The advantage of the PRLS algorithm was tuning flexibility. Specifically, the forgetting factors for the individual estimators can be tuned independently and also more aggressively than in standard RLS while maintaining good precision (i.e., low parameter variances). The benefits from incorporating the 2 RLS algorithms in the modelling approach were demonstrated via simulations of the nylon-6,6 batch polymerization process considered in Chapter 3. Open-loop simulations verified that the precision of the PRLS algorithm is less sensitive to the adaptation aggressiveness compared to conventional RLS. Closed-loop simulations indicated that both RLS algorithms can help improve the performance of a trajectory tracking predictive controller, particularly for large disturbances in the initial conditions.

REFERENCES

[1] S. Aumi and P. Mhaskar, "Adaptive data-based model predictive control of batch systems," *Ind. & Eng. Chem. Res.*, 2011, (submitted).

[2] —, "Adaptive data-based model predictive control of batch systems," in *Proc. of the American Control Conf.*, (accepted), Montréal, QC, 2012.

[3] B. J. Cott and S. Macchietto, "Temperature control of exothermic batch reactors using generic model control," *Ind. & Eng. Chem. Res.*, vol. 28, no. 8, pp. 1177–1184, 1989.

[4] Z. L. Wang, F. Pla, and J. P. Corriou, "Nonlinear adaptive control of batch styrene polymerization," *Chem. Eng. Sci.*, vol. 50, no. 13, pp. 2081–2091, 1995.

[5] P. Jarupintusophon, M. V. L. Lann, M. Cabassud, and G. Casamatta, "Realistic model-based predictive and adaptive control of batch reactor," *Comp. & Chem. Eng.*, vol. 18, pp. 445–449, 1994.

[6] J. D. Boskovic and K. S. Narendra, "Comparison of linear, nonlinear and neural-network-based adaptive controllers for a class of fed-batch fermentation processes," *Automatica*, vol. 31, no. 6, pp. 817–840, 1995.

[7] T. Clarke-Pringle and J. F. MacGregor, "Nonlinear adaptive temperature control of multi-product, semi-batch polymerization reactors," *Comp. & Chem. Eng.*, vol. 21, no. 12, pp. 1395–1409, 1997.

[8] I. Y. Smets, J. E. Claes, E. J. November, G. P. Bastin, and J. F. V. Impe, "Optimal adaptive control of (bio)chemical reactors: Past, present and future," *J. of Process Control*, vol. 14, no. 7, pp. 795–805, 2004.

[9] A. M. F. Fileti, S. L. Cruz, and J. A. F. R. Pereira, "Control strategies analysis for a batch distillation column with experimental testing," *Chem. Eng. Process.*, vol. 39, no. 2, pp. 121–128, 2000.

[10] J. Flores-Cerrillo and J. F. MacGregor, "Within-batch and batch-to-batch inferential-adaptive control of semibatch reactors: A partial least squares approach," *Ind. & Eng. Chem. Res.*, vol. 42, no. 14, pp. 3334–3345, 2003.

[11] C. Kiparissides and S. Shah, "Self-tuning and stable adaptive control of a batch polymerization reactor," *Automatica*, vol. 19, no. 3, pp. 225–235, 1983.

[12] P. Li and G. Wozny, "Tracking the predefined optimal policies for multiple-fraction batch distillation by using adaptive control," *Comp. & Chem. Eng.*, vol. 25, pp. 97–107, 2001.

[13] K. Åström and B. Wittenmark, *Adaptive Control*, Second. Boston, MA: Addison-Wesley Longman Publishing Co., 1994.

[14] T. Fortescue, L. Kershenbaum, and B. Ydstie, "Implementation of self-tuning regulators with variable forgetting factors," *Automatica*, vol. 17, no. 6, pp. 831–835, 1981.

[15] Y. Bar-Shalom and X. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, Third. Storrs, CT: University of Connecticut, 1995.

[16] J. Flores-Cerrillo and J. F. MacGregor, "Latent variable MPC for trajectory tracking in batch processes.," *J. Process Control*, vol. 15, no. 6, pp. 651–663, 2005.

CHAPTER 5

Model Predictive Quality Control of Batch Processes

The results in this chapter have been submitted to the following:

JOURNAL PAPERS:

[1] S. Aumi, B. Corbett, T. Clarke-Pringle, and P. Mhaskar, "Model predictive quality control of batch processes," *J. of Process Control*, 2011, (submitted).

REFEREED CONFERENCE PROCEEDINGS:

[2] S. Aumi, B. Corbett, and P. Mhaskar, "Model predictive quality control of batch processes," in *Proc. of the American Control Conf.*, (accepted), Montréal, QC, 2012.

5.1 INTRODUCTION

The control objective in batch processes is to achieve a specified product quality by batch termination. The economic benefits from batch processing are realized from the consistent production of on-spec product. However, as mentioned in Chapter 1, direct control to a specified quality is often impractical because quality measurements are unavailable online and only made offline after batch completion. Over time, motivated by the increased demands of consistent production of high quality products, numerous batch-to-batch (offline) and within-batch (online) control strategies have been adopted to improve batch process reproducibility.

The idea behind batch-to-batch control is to refine the batch recipe and operating trajectories for the upcoming batch using past data in an attempt to bring the new batch's quality closer to the specified value (e.g., see [3]). Batch-to-batch control strategies range from updating model parameters and then recomputing the batch input trajectories (and/or batch recipe) to directly updating the process variable trajectories using an optimization-based algorithm (e.g. [4]) or the iterative learning control (ILC) framework (e.g., see [5]). The former drives the process towards a specified optimum batch-wise while the latter exploits the repetitive nature of batch systems by using the error in the quality from the last batch to update the process variable trajectories and/or initial conditions.

Batch-to-batch control strategies represent an entirely offline strategy and lack any real-time feedback mechanism for rejecting disturbances encountered *during* batch evolution. This motivates the use of real-time, within-batch control approaches. In many cases, particularly in an industrial setting, a reliable first-principles-based process model that is also computationally amenable for control applications is unavailable, and there is also an absence of quality-related measurements such that the product quality is not observable. For these cases, within-batch control approaches rely on data-based models and can be be broadly divided into trajectory tracking and inferential quality control.

Trajectory tracking is common once the batch recipe has been fixed and reference output trajectories (such as for the reactor temperature) have been optimized to meet the specified quality. As discussed previously, the trajectory tracking control problem is the one of tracking these reference trajectories batch after batch using local controllers. For good tracking performance, advanced control designs that are capable of compensating for the effects of nonlinearities and tracking over a wide operating range are required. While trajectory tracking controllers can reject some disturbances, even with perfect tracking, there is no guarantee that the desired quality will be met. This is because disturbances encountered during a new batch can significantly alter the complex relationships between the quality and output variables. Thus, implementing the same reference trajectories batch-to-batch is not guaranteed to consistently produce on-spec product. To partially counter this problem, the

reference profiles can be “re-optimized" periodically during the batch through a within-batch control approach that employs an inferential quality model.

Inferential quality control is most commonly achieved through multivariate statistical process control (SPC) approaches, particularly those utilizing latent variable tools, such as principal component analysis (PCA) or partial least squares (PLS) regression [6]. For batch processes, the model development for the majority of SPC applications begins with the so-called “batch-wise" unfolding of multiway batch data [7, 8]. The unfolded data is regressed (commonly via PLS regression) onto a matrix of final quality measurements to obtain an inferential PLS quality model (e.g., see [9]) that is usable for predicting the final quality prior to batch completion. For batches with multiple phases or stages with distinct dynamics, multiple phase specific (and transition) models can also be constructed (e.g., see [10, 11]). By processing only successful batches through the resulting model, time-varying control charts for systematically monitoring the quality can be constructed. During the batch evolution, the final quality can be predicted (at every sampling instance or predetermined decision points), and if the prediction exceeds the control limits, appropriate remedial action can be taken to correct the batch. The nature of the corrective action may be heuristics or knowledge-based or more systematic wherein the quality model is inverted (one way or another) to directly compute the future input trajectories that recover the batch. The latter approach has been classified as a mid-course correction (MCC) control strategy (e.g., see [12, 13]). Since it requires model inversion, the effectiveness of a MCC approach is particularly dependent on the quality of the underlying quality model and in general, demands richer training data that spans a wider operating range and exhibits more input variation compared to modelling for conventional SPC (e.g., see [14]).

An important issue that arises in SPC and MCC approaches is that future online measurements that are required to predict the quality are incomplete. More specifically, the data arrangement in the model building process calls for the entire batch trajectory to predict the quality of the batch. However, during a batch, measurements are only available up to the current time, and the future data is required to be completed in some fashion. The choice of the data completion technique plays a key role in the overall performance of the control design. Prediction error in the future data is propagated to the quality prediction error, and both of these errors add uncertainty to any control action computed using the model. This problem is particularly prevalent during the early stages of the batch when most of the information is unknown. In fact, with poor prediction of the future batch behaviour, inputs determined from using the model can drive the batch to a point from where good quality product cannot be produced. This characteristic is typical of methods that lack a causal relationship between the inputs and outputs, and in turn, the quality, which leads to the treatment of the future trajectories as a “missing data" problem (e.g., see [12]).

A variety of ad-hoc approaches exist to handle this “missing data" problem. Many methods utilize missing data algorithms available for latent variable methods. These missing data algorithms work on the assumption that the correlation structure between the collected measurements and future measurements for the new batch is the same as in the training data. Another approach has been to build a finite set of quality models at predetermined decision points (possibly at every sampling instance), and in building each model, rather than using the entire batch trajectory, data only up to the decision point is used [14, 15]. This idea of an evolving model has also been modified for improving the quality prediction in multi-stage batches through consideration of critical-to-quality time periods at specific phases of the batch [16, 17]. One issue with these multi-model based approaches, however, is that quality models developed at early time points can be highly inaccurate since they will not capture the effects of large periods of the batch duration towards the batch quality. Additionally, since the quality prediction is not based on the entire batch trajectory, the time cumulative effects of the process variables on the quality are ignored. While these missing data approaches are useful for predicting the quality and monitoring applications, when the inferential model is used in a control design, the need to consider the nonlinear casual relationship between the future input-output behaviour is obvious. The quality control problem, therefore, stands to gain from the use of a causal, nonlinear model that does not treat the future trajectory as a missing data problem and instead recognizes it as the problem of choosing the remaining input trajectory to obtain a desired quality.

Motivated by the above considerations, in this chapter we develop a within-batch quality control strategy for batch processes that unites a single PLS inferential quality model with the previously developed nonlinear, data-based modelling approach in Chapter 3. By properly representing the future behaviour using a causal model, inputs can be chosen that result in improved quality control. The rest of this chapter is organized as follows. First, we discuss the class of processes considered and outline how to build an inferential quality model from an existing database. Next, we present the details of a predictive controller that is designed to drive a batch process to a desired specified product quality by batch termination. The efficacy of the control design is then demonstrated via simulations of the nylon-6,6 batch polymerization process. Finally, we summarize our results.

5.2 PRELIMINARIES

In this section, we describe the class of batch processes considered. Next, we discuss how an inferential quality model can be identified from a batch database through multiway analysis.

5.2.1 *Process Description*

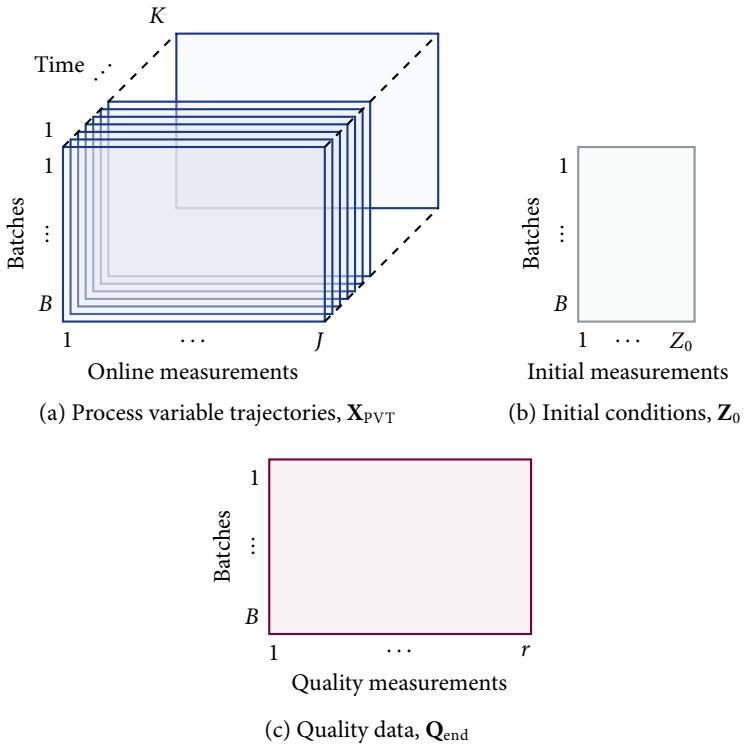
We consider batch processes described by the model in Equation (1.1) but also with measurements of end-use quality variables, $\boldsymbol{q} \in \mathbb{R}^r$, that are available only at batch termination. The general model form is shown below.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \\ \boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) + \boldsymbol{v} \\ \boldsymbol{q}(t_{\text{end}}) &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \\ t &\in [t_0, t_{\text{end}}] \end{aligned} \tag{5.1}$$

The vector function, $\boldsymbol{h}(\cdot) : \mathbb{R}^n \times \mathcal{U} \times \mathcal{W} \to \mathbb{R}^r$, is the quality variables' measurement function.

5.2.2 *Inferential Quality Model*

To understand how to build a model for predicting the quality at batch termination during the batch, we first describe the nature of data available in a batch database. Consider a batch run in which $J = m + p$ input and output variables are measured at K sampling instances. For B batches, these measurements can be organized into a three-dimensional array, $\mathbf{X}_{\text{PVT}} \in \mathbb{R}^{B \times J \times K}$, as shown in Figure 5.1a. Each vertical slice in $\mathbf{X}_{\text{PVT}}$ represents the values of all the measurable variables for all batches at a common sampling instance.



(a) Process variable trajectories, $\mathbf{X}_{\text{PVT}}$

(b) Initial conditions, $\mathbf{Z}_0$

(c) Quality data, $\mathbf{Q}_{\text{end}}$

Figure 5.1: Nature of data in a typical batch database

In addition to $\mathbf{X}_{\text{PVT}}$, measurements of the r quality variables taken post-batch can be summarized in a $B \times r$ quality matrix, $\mathbf{Q}_{\text{end}}$, as shown in Figure 5.1c. Finally, information about the initial conditions for each batch is also typically available (i.e., feed-stock properties, measured raw material properties and compositions, charges of each ingredient, etc.), and this can be summarized in a $B \times Z_0$ matrix, $\mathbf{Z}_0$, (see Figure 5.1b) where Z_0 is the number of known variables related to the initial conditions.

To identify a model that can be used to predict the batch quality, the three-dimensional array, $\mathbf{X}_{\text{PVT}}$, is first transformed into a two-dimensional $B \times JK$ matrix by unfolding it "batch-wise" such that each of its vertical slices is arranged side-by-side [7, 8]. Next, the initial conditions matrix, $\mathbf{Z}_0$, is concatenated to the unfolded matrix, forming a regressor matrix, $\begin{bmatrix}\mathbf{Z}_0 & \mathbf{X}_{\text{PVT}}\end{bmatrix}$, as shown in Figure 5.2.

Figure 5.2: Rearrangement of the data in Figure 5.1a and Figure 5.1b to form the regressor matrix for identifying the quality model in Equation (5.2)

The matrix in Figure 5.2 can be regressed onto the quality data matrix (Figure 5.1c) using linear regression, yielding a model that relates the initial and process conditions over the batch duration to the quality characteristics as shown below.

$$\hat{\mathbf{Q}}_{\text{end}} = \begin{bmatrix}\mathbf{Z}_0 & \mathbf{X}_{\text{PVT}}\end{bmatrix} \mathbf{\Lambda} \tag{5.2}$$

where $\hat{\mathbf{Q}}_{\text{end}}$ is the predicted quality and $\mathbf{\Lambda} \in \mathbb{R}^{(Z_0+JK)\times r}$ defines the quality model. Due to the high dimensionality/multivariate nature of the regressor matrix and the likely presence of correlations among its variables, a latent variable regression technique, such as partial least squares (PLS) regression or principal component regression (PCR), is necessary for appropriately computing $\mathbf{\Lambda}$ (see Section 3.2.3). In this work, we exclusively use PLS regression.

In the PLS model, the process variable trajectories (and initial conditions) and the final qualities are projected onto corresponding latent variable subspaces. The values of the latent variables (i.e., scores) of the process variables are related to those of the final qualities through the linear, inner relationship. The projection essentially represents an estimation of the batch states at the end of the batch (given the process conditions until batch termination) while the inner relationship is a "measurement" function that relates these states to the final quality.

For a *new* batch, at sampling instance k, process variable trajectories are only available only up to k[1]. As a result, the vector required to make the state estimation at batch termination is incomplete. There are ways to eliminate this problem in monitoring applications (e.g., by using multiple models [14], look up tables [18], or a different unfolding scheme [7]); however, when using the model for *predictive control*, the prediction of the future behaviour for a given input trajectory is a necessity. Rather than eliminating the need for future data, we recognize the causal nature of the inputs in determining the future trajectory and in turn the quality.

Remark 5.1: In conventional PLS modelling, a common pre-processing step is to normalize the regressor and response matrices to zero-mean and unit variance. Scaling to unit variance gives each variable equal importance during model identification; however, in many batch processes, there are specific time periods that play a more critical role in determining the final quality than others. A simple way to account for these quality critical periods within the PLS regression framework is to multiply the appropriate columns in the regressor matrix by weighting factors that make them more influential during the computation of the model parameters (and therefore during quality prediction). More formalized approaches for considering time-specific effects are also available in [15, 16].

Remark 5.2: An assumption made during the batch-wise unfolding scheme is that all batches are of equal length and the trajectories are synchronized. In practice, this assumption may not hold for raw batch data. Consequently, several methods have been proposed for addressing unequal batch lengths and to synchronize trajectories. The most common method involves choosing a monotonic indicator variable common to all batches, such as conversion, and re-sampling the measurements with respect to this variable instead of time (e.g., see [19] for an illustrative example for a batch polymerization process). Methods for aligning batch trajectories include dynamic time-warping [20] and curve registration [21].

5.3 MODEL PREDICTIVE QUALITY CONTROL

In this section, the multi-model, data-based modelling approach proposed in Chapter 3 is used in conjunction with the inferential quality model in Section 5.2.2 in a MPC design. The quality model captures the (time cumulative) effects of the entire batch trajectory on the final quality while the multiple linear models for the output variables take the causality and nonlinear relationship between the inputs and outputs into account. The benefit from this approach is the ability to account for the direct connection between the control action and the quality, something that is both expected and desired.

[1] More specifically, the outputs are available up to k and the inputs are available up to $k-1$.

For the process output models, we assume the models for all the outputs are identified simultaneously via PLS regression and take the form shown below.

$$\hat{\boldsymbol{y}}[k] = \sum_{\ell=1}^{L} \omega_\ell[k]\boldsymbol{\theta}_\ell \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix} \tag{5.3}$$

In this model, $\hat{\boldsymbol{y}}$ denotes the predicted outputs and $\bar{\boldsymbol{x}}$ is a vector of lagged outputs and inputs with n_y and n_u lags for each output and input variable (respectively):

$$\bar{\boldsymbol{x}}[k] := \begin{bmatrix} \boldsymbol{y}'[k-1] & \cdots & \boldsymbol{y}'[k-n_y] & \boldsymbol{u}'[k-1] & \cdots & \boldsymbol{u}'[k-n_u] \end{bmatrix}'$$

The *matrix*, $\boldsymbol{\theta}_\ell$, defines ℓ-th model, and the model weights, $\omega_\ell[k]$, are given by:

$$\omega_\ell[k] = \frac{\|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{\sum_{j=1}^{L} \|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_j\|^{-2}}$$

where $\boldsymbol{c}_\ell$ is model ℓ's centre point. The details of the identification procedure are in Section 3.3.

Once a quality model of the form in Equation (5.2) and a model for the outputs of the form in Equation (5.3) have been identified, the MPC optimization problem shown below can be solved at sampling instance k for controlling the quality to the given target $\boldsymbol{q}_{\text{des}}$.

$$\min_{\boldsymbol{u}[k]\in\mathcal{U}} J_Q = \|\hat{\boldsymbol{q}} - \boldsymbol{q}_{\text{des}}\|_{\boldsymbol{\Xi}}^2 + \sum_{i=k}^{K} \|\boldsymbol{u}[i] - \boldsymbol{u}[i-1]\|_{\boldsymbol{\Pi}}^2 \tag{5.4a}$$

subject to: $\hat{\boldsymbol{y}}[k] = \boldsymbol{y}(t)$ (5.4b)

$$\hat{\boldsymbol{y}}[k] = \sum_{\ell=1}^{L} \omega_\ell[k]\boldsymbol{\theta}_\ell \begin{bmatrix} \bar{\boldsymbol{x}}[k] \\ 1 \end{bmatrix}, \quad \text{for } k \in [k+1, K] \tag{5.4c}$$

$$\omega_\ell[k] = \frac{\|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_\ell\|^{-2}}{\sum_{j=1}^{L} \|\bar{\boldsymbol{x}}[k] - \boldsymbol{c}_j\|^{-2}} \tag{5.4d}$$

$$\bar{\boldsymbol{x}}[k] = \begin{bmatrix} \boldsymbol{y}'[k-1] & \cdots & \boldsymbol{y}'[k-n_y] & \boldsymbol{u}'[k-1] & \cdots & \boldsymbol{u}'[k-n_u] \end{bmatrix}' \tag{5.4e}$$

$$\boldsymbol{x}'_{\text{future}} = \begin{bmatrix} \boldsymbol{u}'[k] & \hat{\boldsymbol{y}}'[k+1] & \boldsymbol{u}'[k+1] & \cdots & \hat{\boldsymbol{y}}'[K] \end{bmatrix}' \tag{5.4f}$$

$$\hat{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{x}_{\text{past}} & \boldsymbol{x}_{\text{future}} \end{bmatrix} \boldsymbol{\Lambda} \tag{5.4g}$$

In this optimization problem, the objective function consists of a term for minimizing the discrepancy between the target product quality and the predicted end-point quality, $\hat{\boldsymbol{q}}$, and a move suppression factor. Each term's relative importance is set by the positive-definite weighting matrices, $\boldsymbol{\Xi}$ and $\boldsymbol{\Pi}$. Equation (5.4b) represents the MPC initialization at the current plant conditions, and Equation (5.4c) represents the prediction of the future outputs using the data-based model (given the current trajectory of inputs in the optimizer). The predicted process outputs and optimizer inputs are stored appropriately in the row vector,

$\boldsymbol{x}_{\text{future}}$, through Equation (5.4f). This vector is concatenated with a vector of previous process outputs and implemented inputs, $\boldsymbol{x}_{\text{past}}$, which is known prior to solving the MPC optimization problem:

$$\boldsymbol{x}'_{\text{past}} = \begin{bmatrix} \boldsymbol{z}'[0] & \boldsymbol{y}'[0] & \boldsymbol{u}'[0] & \boldsymbol{y}'[1] & \boldsymbol{u}'[1] & \cdots & \boldsymbol{y}'[k] \end{bmatrix}'$$

The vector, $\boldsymbol{z}'(0)$, denotes all the information known prior to starting the batch (i.e., the initial conditions). The concatenated vector, $\begin{bmatrix} \boldsymbol{x}_{\text{past}} & \boldsymbol{x}_{\text{future}} \end{bmatrix}$, is used to predict the quality through Equation (5.4g).

A distinguishing feature of this MPC design is the use of a causal (and nonlinear) model for filling in the future outputs prior to making the quality prediction. In contrast, the majority of the control designs that have utilized an inferential end-point quality model exploit missing data algorithms that exist for latent variable models. Based on the data collected up to the current sampling instant, these algorithms essentially invert the *linear* PLS model such that the future behaviour maintains the same correlation structure as previous batches. This leads to an inherent mismatch in the sense that the predicted future behaviour is based on past data (that typically uses existing PI controllers), which, in turn, is used to compute the current control action via a different control algorithm than in the data set.

By comparison, the proposed approach recognizes that the problem is not that of missing data because the future trajectories depend on output and input trajectories up to the current point as well as future input moves. The only "missing" part therefore is the part that needs to be computed by the controller - the set of future control moves. By utilizing an appropriate model (which captures the process nonlinearities) that links the future inputs to the future outputs and in turn to the quality, the controller then computes the set of input moves that would yield the desired quality.

Remark 5.3: Many control designs have tried to eliminate the missing data problem entirely through evolving quality models (at each sampling instance or a selected number of predetermined decision points), which utilize measurements only up to a given time. These models are designed to forecast the final quality without the future batch trajectories and inherently rely on the assumption that the same control action is implemented for the rest of the batch. Therefore, while such methods may be good to predict the quality under an existing controller, they are not well suited for use in a control design aimed at computing the control action to yield the desired quality.

Remark 5.4: One requirement of existing quality control approaches is that of equal batch times. In the existing approaches where the batch times are different, some kind of a synchronization or alignment technique along some variable other than time (e.g., conversion) is required. Such situations can be readily handled by the proposed quality prediction approach by virtue of using a dedicated (nonlinear) model for predicting the future batch behaviour. In

particular, all the batch data can be used to build the model for the outputs while the quality model can be built using a common time from the end of the batch for all batches (i.e., using the batch time of the shortest batch).

Remark 5.5: The prediction horizon for the MPC optimization problem must extend to the end of the batch; thus, the prediction horizon, $P = K - k$, shrinks at every sampling instance. During the early stages of the batch when k is low, the MPC optimization problem may be too computationally demanding for real-time implementation. Under such circumstances, the optimization problem can be used to update the reference trajectories for local controllers rather than directly computing the inputs. Specifically, while the optimization problem is being solved, trajectory tracking controllers can be used to track the nominal reference trajectories, and upon completion of the optimization problem, the trajectories of the measurable output variables from the solution can be specified as the updated nominal reference trajectories in the local trajectory tracking controllers.

5.4 Simulation Example: Nylon-6,6 Batch Polymerization

In this section, we demonstrate the efficacy of the MPC design in Section 5.3 through closed-loop simulations of the nylon-6,6 batch polymerization process. We develop data-based models for the quality and output variables using the results in this chapter and Chapter 3. Using these models, we implement the MPC design in Section 5.3 and compare its performance against trajectory tracking control.

A process overview was previously given in Section 3.5.2. As before, we utilized the mathematical model of this process in [22] that takes the general form shown in Equation 5.1. The state vector is comprised of the molar amounts of each functional group and evaporated HMD, the reaction medium mass, temperature, and volume, and reactor pressure. The states were assumed to be measured once at the initial time, but note that many were trivially zero due to the absence of any reaction. The manipulated inputs were taken to be the steam jacket pressure, P_j (psi), and vent rate, v (kg/h): $\boldsymbol{u} = \begin{bmatrix} P_j & v \end{bmatrix}'$, and the constraints were defined as follows: $\boldsymbol{u}_{\min} = \begin{bmatrix} 700 & 0 \end{bmatrix}'$ and $\boldsymbol{u}_{\max} = \begin{bmatrix} 1800 & 2000 \end{bmatrix}'$. All batches were 3 hours, eliminating the requirement for any trajectory synchronization, with a sampling period of 1 minute.

The outputs were taken to be the reaction mixture temperature, T (K), volume, V (L), and the viscosity, η (cP): $\boldsymbol{y} = \begin{bmatrix} T & V & \eta \end{bmatrix}'$. Note that in practice, while the viscosity may not be directly measurable at the sampling period of 1 minute, stirrer torque measurements are typically available in real time at every sampling instance. The stirrer torque is strongly correlated with the solution viscosity with a more viscous polymer resulting in higher torque

(for a fixed RPM stirrer motor). Thus, the viscosity measurements could be readily replaced with torque measurements without significant changes in the results.

The product quality of nylon-6,6 polymer is defined by the number average molecular weight, MW, and the residual amide concentration, R_{NH_2} (mol/g); thus, we had: $\boldsymbol{q} = \begin{bmatrix} \text{MW} & R_{NH_2} \end{bmatrix}'$. The control objective considered in this work was to achieve end-point qualities of $\boldsymbol{q}_{\text{des}} = \begin{bmatrix} 5569 & 136 \end{bmatrix}'$. Both qualities are related to the state variables through highly nonlinear relationships (see [22] for details).

5.4.1 Data-based Model Development

To develop data-based models for the quality and output variables, an artificial batch database of the form in Figure 5.1 was first generated. To this end, the state-space model was simulated 80 times from different initial conditions (30 batches were reserved as the validation data set). In generating the database, a set of reference T and V profiles, denoted by T_{ref} and V_{ref}, presented in Figure 5.3 were tracked using P_j and v (respectively) via 2 tightly tuned PI controllers. In addition to these closed-loop trajectories, the database was supplemented with 4 open-loop identification batches. For these batches, low amplitude, pseudo-random binary sequence (PRBS) signals were added on top of the nominal input trajectories. In Figure 5.4, the input trajectories for one of these identification batches are shown together with the nominal trajectories (that correspond to T_{ref} and V_{ref} in Figure 5.3).

The final database consisted of measurements of the states at the initial time, T, V, and η at every sampling instance, and the qualities at batch termination. Prior to developing the models, the η measurements were replaced by $\ln(\eta)$ for a linearizing effect after observing an exponential rise in the viscosity towards the latter stages of all the batches.

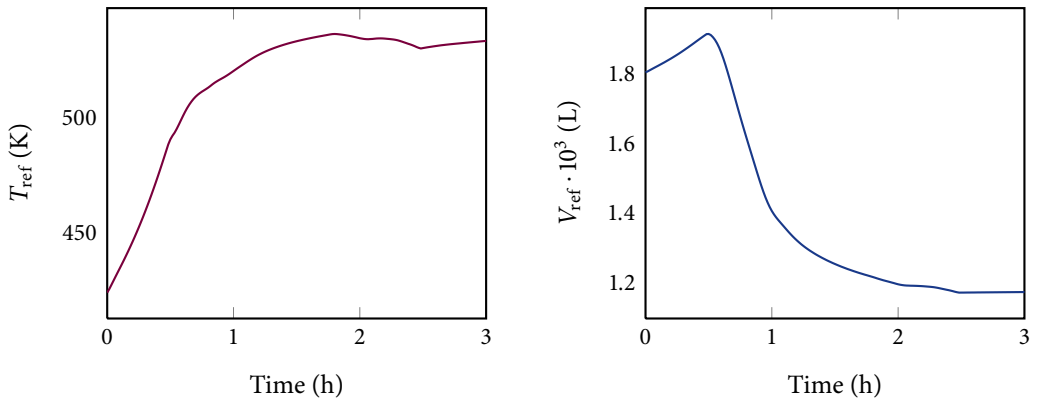


Figure 5.3: Reference T and V profiles for the nylon-6,6 batch polymerization process. These trajectories were tracked using 2 PI controllers in generating the database.

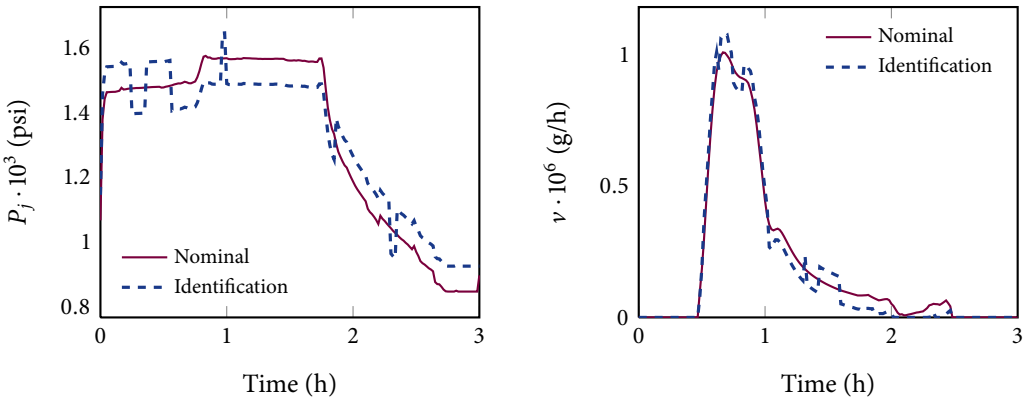


Figure 5.4: Representative input trajectories for the identification batches and nominal input trajectories in generating the database for the nylon-6,6 batch polymerization process

Inferential Quality Model

Using the resulting database, the procedure outlined in Section 5.2.2 was carried out to build an inferential quality model using PLS regression. As discussed in Remark 5.1, quality critical periods during the batch can be given more weight prior to computing the model parameters. For the nylon-6,6 process, the initial conditions and process behaviour during the boiling and finishing phases are more influential to the final quality compared to the heating phase; consequently, columns corresponding to these conditions in the unfolded regressor matrix (see Figure 5.2) were given 6, 2, and 4 times more weight than the heating phase (respectively). The motivation behind placing the lowest weight on the heating phase was because it corresponded to a limited extent of the polymerization compared to the other phases. The weights for the other phases and initial conditions were found iteratively and chosen to minimize the root mean squared error (RMSE) in the predicted qualities of the validation batches. Thus, these weights were essentially tuning parameters in the model. The high weight was placed on the initial conditions to compensate for the fact that they constituted a very small portion of the regressor matrix compared to the other phases. In Figure 5.5, the qualities predicted by the PLS model for the 30 validation batches are displayed along with the database qualities. The number of latent variables/principal components in the PLS model, 24, was selected to minimize the RMSE in the predicted qualities of the validation batches. From the discussion in Section 5.2.2, this meant 24 latent variables were required to estimate the states at batch termination. Note that the total number of columns in the regressor matrix was over 900; thus, 24 latent variables still represent a significant reduction in the dimensionality of the process. Additionally, the training data contained 4 identification batches that expanded the range of typical operating conditions, calling for additional latent variables. Overall, the inferential quality model predicted the final qualities with relatively good accuracy.

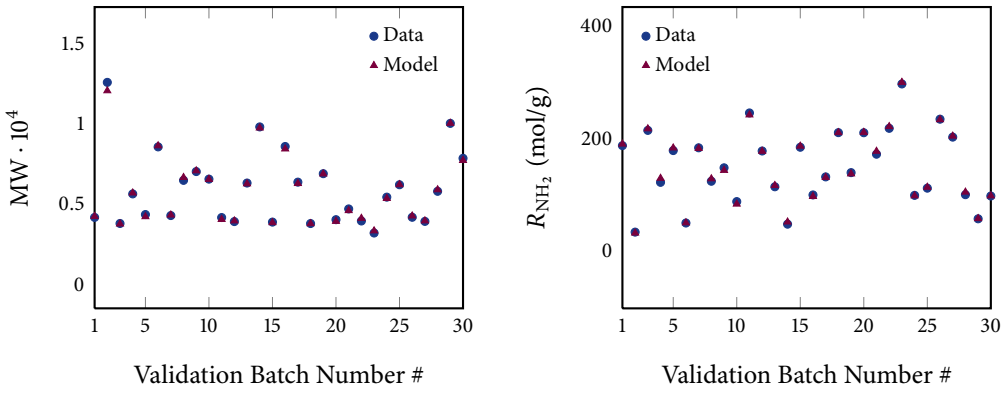


Figure 5.5: Comparison of the qualities predicted by the inferential quality model with the qualities in the validation data set for the nylon-6,6 batch polymerization process

Process Outputs Model

Next, multiple local linear models were fit for T, V, and $\ln(\eta)$ since these variables must be predicted (for a given trajectory of the inputs) as part of the quality prediction. The steps in Algorithm 3.2 were carried out with PLS regression. The number of input and output lags were found to be 1 (i.e., $n_y = n_u = 1$) and the number of clusters, L, was 7. One explanation for requiring only one lag is the assumption of the same lag structure for all the local models (note that this assumption can be readily relaxed if needed). With this assumption, using all first order models minimized the possibility of over-fitting, and in this case, yielded the lowest RMSE values. In Figure 5.6, we compare the data-based model's outputs with the corresponding trajectories in the validation data set. Overall, the multi-model approach captured the major nonlinearities and provided relatively reliable output predictions.

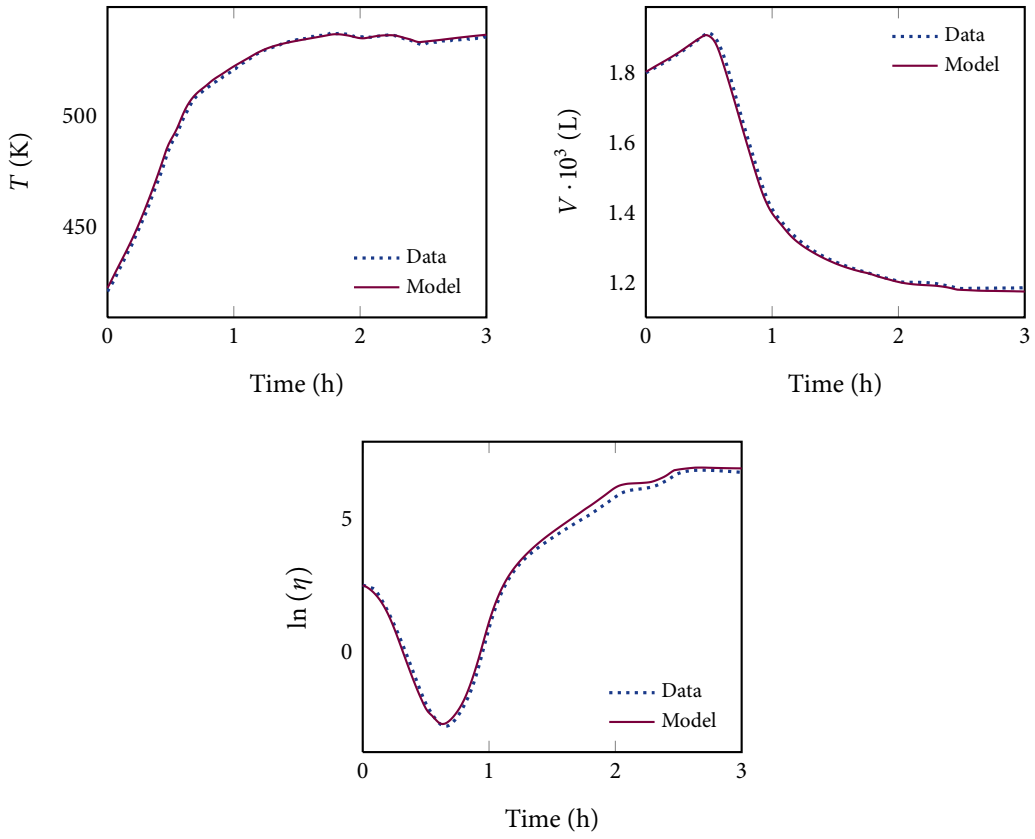


Figure 5.6: Comparison of the data-based model's outputs with the corresponding trajectories in the validation data set for the nylon-6,6 batch polymerization process

5.4.2 Closed-loop Results

Using the models developed in the previous subsection, in this section, we present the results from implementing the predictive controller in Section 5.3 and compare its control performance against trajectory tracking via PI controllers. For these simulations, we considered 21 new initial conditions that were not in the training or validation data sets. The reference trajectories for the PI-based trajectory tracking simulations were those presented earlier in Figure 5.3, and the loop-pairings and tunings were kept consistent with the database generation procedure. In solving the MPC optimization problem, the initial guess for the input trajectories was set to the nominal trajectories at $t = 0$ and the tail of the solution at the previous sampling instance for all subsequent sampling instances. The computation time (as reported by the Matlab functions, `tic` and `toc`) for the predictive controller at $t = 0$ was 1.5 seconds on an Intel Quad Core machine using GAMS with IPOPT as the optimization software. The computation times for all successive sampling times were lower due in part to the shrinking horizon nature of the optimization problem, indicating that the MPC design was amenable to real-time application.

In Figure 5.7, the final qualities yielded from trajectory tracking are compared with those from the proposed MPC design. On average, there was a significant improvement in meeting the specified quality. With trajectory tracking, the standard deviations from the target quality were 3392 and 69.69 mol/g for MW and R_{NH_2} respectively. These values were reduced to 1240 and 40.71 mol/g by the predictive controller.

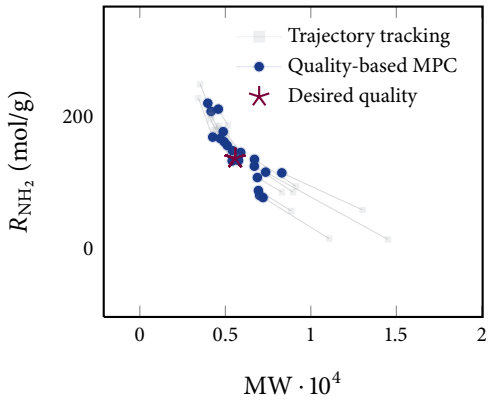


Figure 5.7: Comparison of the final qualities from trajectory tracking and the proposed quality-based MPC design for 21 new initial conditions for the nylon-6,6 batch polymerization process

Input trajectories (from both controllers) for 1 of the batches are shown in Figure 5.8. For these inputs, trajectory tracking yielded qualities of 6434 and 110 mol/g for MW and R_{NH_2} (respectively) while MPC yielded qualities of 5559 and 131 mol/g. Recall that the desired values for the final MW and R_{NH_2} were 5569 and 136 mol/g. Comparing the input trajectories in Figure 5.8, we observe that the MPC prescribed inputs followed the same general trends as with trajectory tracking but with sufficient refinements to significantly improve upon the quality.

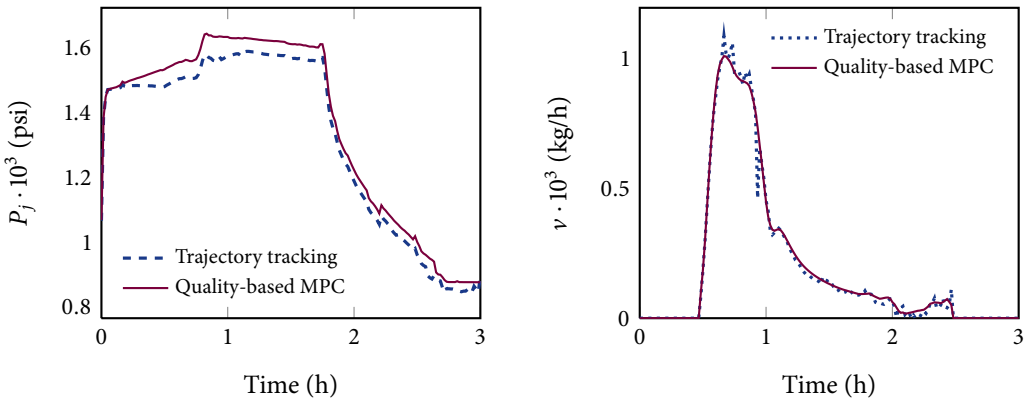


Figure 5.8: Representative inputs profiles prescribed by the quality-based MPC design and trajectory tracking PI controllers for the nylon-6,6 batch polymerization process

5.5 CONCLUSIONS

In this work, we proposed a predictive control design for batch processes designed to drive the batch to a specified quality by batch termination. The MPC design utilized two types of models: a PLS model which related the process conditions over the entire batch duration to the final product quality and weighted local linear models that were used to predict the future (unknown) process conditions up to batch termination (based on a candidate input trajectory). Accounting for the causality and nonlinear relationships in the future through the multiple local linear models led to more effective control action. The proposed control design was applied on the highly nonlinear nylon-6,6 batch polymerization process, and it significantly reduced the variances in the quality variables that were obtained using PI-based trajectory tracking control.

REFERENCES

[1] S. Aumi, B. Corbett, T. Clarke-Pringle, and P. Mhaskar, "Model predictive quality control of batch processes," *J. of Process Control*, 2011, (submitted).

[2] S. Aumi, B. Corbett, and P. Mhaskar, "Model predictive quality control of batch processes," in *Proc. of the American Control Conf.*, (accepted), Montréal, QC, 2012.

[3] T. L. Clarke-Pringle and J. F. MacGregor, "Optimization of molecular-weight distribution using batch-to-batch adjustments," *Ind. & Eng. Chem. Res.*, vol. 37, no. 9, pp. 3660–3669, 1998.

[4] D. Dong, T. J. McAvoy, and E. Zafiriou, "Batch-to-batch optimization using neural network models," *Ind. & Eng. Chem. Res.*, vol. 35, no. 7, pp. 2269–2276, 1996.

[5] J. Flores-Cerrillo and J. F. MacGregor, "Iterative learning control for final batch product quality using partial least squares models," *Ind. & Eng. Chem. Res.*, vol. 44, no. 24, pp. 9146–9155, 2005.

[6] P. Geladi and B. Kowalski, "Partial least-squares regression: A tutorial," *Anal. Chim. Acta*, vol. 185, pp. 1–17, 1986.

[7] S. Wold, P. Geladi, K. Esbensen, and J. Ahman, "Multi-way principal components-and PLS-analysis," *J. of Chemometrics*, vol. 1, no. 1, pp. 41–56, 1987.

[8] P. Nomikos and J. F. MacGregor, "Monitoring batch processes using multiway principal component analysis," *AIChE J.*, vol. 40, no. 8, pp. 1361–1375, 1994.

[9] J. Flores-Cerrillo and J. F. MacGregor, "Within-batch and batch-to-batch inferential-adaptive control of semibatch reactors: A partial least squares approach," *Ind. & Eng. Chem. Res.*, vol. 42, no. 14, pp. 3334–3345, 2003.

[10] C. Undey and A. Cinar, "Statistical monitoring of multistage, multiphase batch processes," *IEEE Control Syst. Mag.*, vol. 22, no. 5, pp. 1361–1375, 2002.

[11] Y. S. Ng and R. Srinivasan, "An adjoined multi-model approach for monitoring batch and transient operations," *Comp. & Chem. Eng.*, vol. 33, no. 4, pp. 887–902, 2009.

[12] J. Flores-Cerrillo and J. F. MacGregor, "Control of particle size distributions in emulsion semibatch polymerization using mid-course correction policies," *Ind. & Eng. Chem. Res.*, vol. 41, no. 7, pp. 1805–1814, 2002.

[13] Y. Yabuki and J. F. MacGregor, "Product quality control in semibatch reactors using midcourse correction policies," *Ind. & Eng. Chem. Res.*, vol. 36, no. 4, pp. 1268–1275, 1997.

[14] D. Wang and R. Srinivasan, "Multi-model based real-time final product quality control strategy for batch processes," *Comp. & Chem. Eng.*, vol. 33, no. 5, pp. 992–1003, 2009.

[15] J. C. Gunther, J. S. Conner, and D. E. Seborg, "Process monitoring and quality variable prediction utilizing PLS in industrial fed-batch cell culture," *J. Process Control*, vol. 19, no. 5, pp. 914–921, 2009.

[16] N. Lu and F. Gao, "Stage-based process analysis and quality prediction for batch processes," *Ind. & Eng. Chem. Res.*, vol. 44, no. 10, pp. 3547–3555, 2005.

[17] C. Zhao, F. Wang, Z. Mao, N. Lu, and M. Jia, "Quality prediction based on phase-specific average trajectory for batch processes," *AIChE J.*, vol. 54, no. 3, pp. 693–705, 2008.

[18] S. Patel, R. Yelchuru, S. Ryaliand, and R. D. Gudi, "Discriminatory learning based performance monitoring of batch processes," in *Proc. of the American Control Conf.*, 2011, pp. 2552–2557.

[19] T. Kourti, J. Lee, and J. F. Macgregor, "Experiences with industrial applications of projection methods for multivariate statistical process control," *Comp. & Chem. Eng.*, vol. 20, pp. 745–750, 1996.

[20] A. Kassidas, J. F. MacGregor, and P. A. Taylor, "Synchronization of batch trajectories using dynamic time warping," *AIChE J.*, vol. 44, no. 4, pp. 864–875, 1998.

[21] C. Undey, B. A. Williams, and A. Cinar, "Monitoring of batch pharmaceutical fermentations: Data synchronization, landmark alignment, and real-time monitoring," in *Proc. of the IFAC World Congress on Automatic Control*, vol. 15, Barcelona, ES, 2002.

[22] S. A. Russell, D. G. Robertson, J. H. Lee, and B. A. Ogunnaike, "Control of product quality for batch nylon-6,6 autoclaves," *Chem. Eng. Sci.*, vol. 53, no. 21, pp. 3685–3702, 1998.

CHAPTER 6

Conclusions and Future Work

In this chapter, the main contributions of this work are summarized and future research directions are discussed.

6.1 CONCLUSIONS

In this work, we addressed the problems of modelling and control of batch processes. For the initial phase of this research, covered in Chapter 2, a computationally efficient, nonlinear, and robust predictive controller was formulated. A key feature of this controller was that it required the online computation of only the immediate control action while guaranteeing reachability to a desired end-point neighbourhood. By adopting this one-step ahead type approach, the MPC optimization problem size was amenable for real-time implementation even when embedded with the full nonlinear process model. To ensure each control move prescribed by the controller steered the process towards the desired final conditions, the novel concept of RTRRs was used. An optimization-based algorithm was developed to generate and mathematically characterize these sets (as ellipsoids) at each sampling instance *a priori*, and they were subsequently incorporated into the MPC design. The a priori nature of all RTRR related computations implied the real-time performance of the controller is independent of the RTRR generation computational requirements. In addition to reducing the online computational demands, the proposed MPC design also proved to be a useful tool in the safe-steering framework for handling faults in batch processes. In this framework, the RTRR-based MPC design is utilized in steering state trajectories (using the functioning inputs) during fault rectification such that they are maintained in a region from where the batch can be recovered following fault repair. The advantages of the proposed RTRR-based MPC law over conventional end-point based MPC designs were illustrated via simulations of a fed-batch reactor process.

An underlying assumption in Chapter 2 and the majority of the existing results on model-based batch process control is the availability of the reliable first-principles-based process model. In many cases, however, such a model may be unavailable due to overly complex underlying process phenomena that make the modelling procedure too difficult/expensive. Moreover, even when available, the resulting model may be inaccurate because many of the key model parameters are not known precisely and/or the simplifying assumptions taken during model development are violated in practice. Furthermore, the model may be ill-suited for online applications due to significant nonlinearities, discontinuities, and/or high dimensionality. These factors motivated the next phase of the research in Chapter 3 wherein we addressed the problem of data-based modelling and control of batch processes.

In contrast to conventional data-based modelling (which have been mainly developed with continuous processes in mind for which a single linear model is often sufficient for control purposes), in Chapter 3, we proposed a multi-model approach designed to capture batch process nonlinearities. The proposed modelling approach exploited historical batch data, the simplicity of local linear models, the data extraction capabilities of latent variable methods, and the use of appropriate clustering and weighting techniques to capture the nonlinear nature of batch process dynamics. The resultant model was used in 2 ways. First, the RTRR framework developed in Chapter 2 was appropriately modified to use the data-based model, leading to a data-based or empirical RTRR-based predictive controller. As shown through simulations of a fed-batch reactor process, this new controller shared the same fault-tolerant characteristics as the originally proposed RTRR-based MPC design. Next, we applied the data-based modelling methodology on a complex, industrially relevant nylon-,6, batch polymerization process to develop models for the key process outputs, namely the reactor temperature and pressure. The resulting models were used to design a predictive controller for tracking reference/set-point profiles of these key outputs. Closed-loop simulation results clearly demonstrated the advantages of using the proposed control design over PI control and a simple implementation of LV-MPC.

Next, in Chapter 4, we generalized the data-based modelling approach in Chapter 3 to handle time-varying uncertainties by adding online learning ability to the model. First, it was demonstrated how the standard RLS algorithm with a forgetting factor can be applied in a straightforward manner to provide online updates of the model parameters. Next, a probabilistic RLS (PRLS) estimator (also with a forgetting factor) was developed that updated each model individually according to its probability of being representative of the local dynamics. The advantage of the PRLS algorithm was tuning flexibility. Specifically, the forgetting factors for the individual estimators could be tuned independently and also more aggressively than in standard RLS while maintaining good precision. The benefits from incorporating the two RLS algorithms in the modelling approach were demonstrated via simulations of the nylon-6,6 batch polymerization reactor.

Finally, in Chapter 5, we proposed a predictive control design designed to directly control the process to a desired end-point in situations where the control objective is typically pursued indirectly via trajectory tracking approaches. The MPC design utilized 2 types of models: an inferential quality model that related the process conditions over the entire batch duration to the final product quality and weighted local linear models to predict the future (unknown) process conditions up to batch termination, based on a candidate input trajectory. Accounting for the causality and nonlinear relationships in the future through the multiple local linear models led to more effective control action, and the computation times of the controller were sufficiently low to permit its real-time implementation. The proposed control design was demonstrated via simulations of the nylon-6,6 batch polymerization process where it was shown to significantly reduce the variances in the quality variables that were observed with trajectory tracking control.

6.2 FUTURE WORK

In this section, a few topics of future research are suggested.

- The data-based modelling approach developed in Chapter 3 has no limitations on the type of process, batch or continuous, for which it is applicable. In this work, the focus was specifically on batch processes; thus, an important goal of the future research is to demonstrate the effectiveness of the modelling approach on continuous processes. For the simulation examples considered in this research, the models were identified from a batch database consisting of closed-loop trajectories that were a result of trajectory tracking control. In these databases, the set-point/reference output trajectories varied throughout the entire batch duration over a fairly large range of operating conditions, causing the inputs to be adjusted in response. This, in turn, kept the process more or less persistently excited and resulted in “good" training data for fitting the models. When the modelling approach is applied to continuous processes, the occurrence of time-varying set-points over a wide range of operating conditions will be rare and questions regarding the training data satisfying important identifiability conditions have to be answered.

- A predictive controller designed using a model developed from the data-based modelling approach in Chapter 3 requires solving a NLP due to the nonlinear weighting function in the final model form. In some cases, this can prevent the real-time applicability of the controller. Note however that the nonlinearities in the final model form are entirely a result of the weighting function. As a result, if the model weights are computed prior to the MPC optimization, the predictive model becomes linear in the decision variables, and consequently, the MPC optimization problem can be formulated as an efficiently solvable convex program. Based on this argument, one

computationally flexible way to solve the MPC optimization problem is to adopt an iterative approach as follows.

1. At the current sampling instance, compute the model weights over the prediction horizon using nominal reference trajectories (for both the output and input variables).
2. Fix the model weights in the MPC formulation to those computed in Step 1 and solve the resulting convex program for the input trajectories.
3. From the solution trajectories, recompute the model weights.
4. Using the updated weights, resolve the convex program to obtain a new input trajectory.
5. If the solution time has expired or the difference between the input trajectories in Steps 2 and 4 is less than a predefined tolerance, accept the solution from Step 4 as the final solution. Otherwise, return to Step 1 with the input trajectories from Step 4.

The main issue that has to be investigated with this approach is the convergence of the inputs to an optimum of the original NLP.