Robust Nonlinear Model Predictive Control using Polynomial Chaos Expansions

by

Divya Kumar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Chemical Engineering

Waterloo, Ontario, Canada, 2015

AUTHOR'S DECLARATION

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

STATEMENT OF CONTRIBUTION

Chapter 3 is based on published work by Kumar *et al.* entitled, “Robust Nonlinear MPC based on Volterra series and Polynomial Chaos Expansions”, in *Journal of Process Control*, (2014) 24, 304-317 . The entire work in this publication has been done by thesis author under direct supervision of PhD supervisor Dr. H. Budman.

Chapter 4 is partly based on refereed conference processdings by Kumar *et al*. entitled, “Robust nonlinear predictive control for a bioreactor based on a Dynamic Metabolic Flux Balance model”, in *IFAC* 2015, Vancouver, Canada. The entire work in this publication has been done by thesis author under direct supervision of PhD supervisor Dr. H. Budman.

Chapter 5 is based on refereed conference proceedings by Kumar *et al.* entitled, “Robust-distributed MPC with robust observer to handle communication loss”, in *IFAC* 2012, Singapore. The entire work in this publication has been done by thesis author under direct supervision of PhD supervisor Dr. H. Budman and co-author Dr. Walid Al-Gherwi.

Abstract

The performance of model predictive controllers (MPCs) is largely dependent on the accuracy of the model predictions as compared to the actual plant outputs. Irrespective of the model used, first-principles (FP) or empirical, plant-model mismatch is unavoidable. Consequently, model based controllers must be robust to mismatch between the model predictions and the actual process behavior. Controllers that are not robust may result in poor closed loop response and even instability. Model uncertainty can generally be formulated into two broader forms, parametric uncertainty and unstructured uncertainty. Most of the current robust nonlinear MPC have been based on FP-model where only robustness to bounded disturbances rather than parametric uncertainty has been addressed. Systematically accounting for parametric uncertainty in the robust design has been difficult in FP-models due to varying forms in which uncertain parameters occur in the models. To address parametric uncertainty robustness tests based on Structured Singular Value (SSV) and Linear Matrix Inequalities (LMI) have been proposed previously, however these algorithms tend to be conservative because they consider worst-case scenarios and they are also computationally expensive. For instance the SSV calculation is NP-hard and as a result it is not suitable for fast computations. This provides motivation to work on robust control algorithms addressing both parametric and unstructured uncertainty with fast computation times. To facilitate the design of robust controllers which can be computed fast, empirical models are used in which parametric uncertainty is propagated using Polynomial Chaos Expansion (PCE) of parameters. PCE assists in speeding up the computations by providing an analytical expression for the $\mathcal{L}^2$-norm of model predictions while also eliminating the need to design for the worst-case scenario which results in conservatism. Another way of speeding up computations in MPC algorithms is by grouping subsets of available the inputs and outputs into subsystems and by controlling each of the subsystems by MPC controllers of lower dimensions. This latter approach, referred in the literature as Distributed MPC, has been tackled by different strategies involving different degrees of coordination between subsystems but it has not been studied in terms of robustness to model error.

Based on the above considerations the current work investigates different robustness aspects of predictive control algorithms for nonlinear processes with special emphasis on the following three situations, i) a nonlinear predictive control based on a Volterra series model where the uncertain parameters are formulated as PCE's, ii) The application of a PCE-based approach to control and optimization of bioreactors where the model is based on dynamic flux metabolic models, and iii) A Robust Distributed MPC with a robust estimator that is needed to account for the interactions between sub-systems in distributed control.

Acknowledgements

First, I would like to thank Prof. Hector Budman for his support, inspiration and guidance throughout my PhD. Thank you Hector for the patience, trust and support during the numerous skype sessions in the final years I spent flying coast to coast.

Next, I would like to thank the committee members: Dr. Prashant Mhaskar, Dr. Kirsten Morris, Dr. Luis Sandoval and Dr. Eric Croiset for their comments and creating a space for healthy discussion during the exam.

Thanks to Liz Bevan, Judy Caron, Rose Guderian and Ingrid Sherrer; who helped me get through all the paperwork easily. Dennis Herman and Ravindra Singh have been of immense help for timely support with computational resources.

I am also grateful to the NSERC funding provided for this work.

I would like to thank my colleagues at University (Kaveh, Jasdeep, Hengameh and Yuncheng) for insightful discussions, server reboots and making my time at UW fun.

I would like to thank Ali Esmaili, my manager at Air Products and Chemicals, for providing me flexible hours.

Last but not the least, the love and support from my family has made this day possible and I feel blessed having them in my life. I cannot forget Ma/Pa's constant reminders of my schedule, Amma/Appa's encouragement and Mahesh's support. Samiksha, Arjun and Mahesh: you are AWESOME!

Dedication

To Ma, Pa, Mahesh, Arjun and Samiksha

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

The performance of a model-based controller is largely dependent on the accuracy of model predictions as compared to actual plant outputs. Plant dynamics can be either represented by first-principles or empirical models. First-principles' models, which refer to models based on material and energy balances are in general considered superior for predicting behavior outside of the window of data used for calibration because they include underlying physics of the process as opposed to empirical models which are trying to map a relationship between available plant input and output data. However, developing first-principles model is not always possible given the complex nature of chemical plants. On the other hand empirical models are easier to develop, but can have structural errors due to missing physical insight of the process and therefore are less accurate for extrapolating behavior beyond the window of operation used for model calibration. Model Predictive Control (MPC) is one of the most prominently used model-based control methodology in the process industries. MPC calculates a set of future moves by minimizing the error between predefined set-points or reference trajectories and future plant outputs predicted on the basis of a model over a prediction horizon. Since the control actions are obtained by solving an optimization problem, input and output constraints can be included as part of this problem. Most of the applications of MPC in industry are based on linear models which are empirical in nature and are identified from input-output experimental data. This trend is attributed to the fact that MPC based on first-principles (FP) models are both expensive to develop and to implement. Instead, several predictive control algorithms based on nonlinear empirical models such as Volterra series, Weiner series and Hammerstein series have been proposed since they are easier to develop and implement (Doyle *et al.*, 1995, Fruzzetti *et al.*, 1997, Maner and Doyle, 1997, Norquay *et al.*, 1999).

Irrespective of the type of model used, i.e. empirical or first-principles, plant-model mismatch is unavoidable. Consequently, model-based controllers must be robust to mismatch between the model predictions and the actual process behavior. Controllers that are not

robust may result in poor closed loop response and even instability. Model uncertainty can generally be formulated into two broader forms, parametric uncertainty and unstructured uncertainty. Parametric uncertainty is referred as structured since it involves errors in model parameters within a model of a given structure. On the other hand, unstructured uncertainty is used to represent model error that cannot be related to specific model parameters.

One of the biggest drawback when designing an FP-model based control strategy for robustness is the fact that the uncertainty in parameters often appears in functional forms, e.g. in exponential Arrhenius terms, that are not amenable for traditional robustness tests such as infinity norms based tests (McFarlane and Glover, 1992, Kwakernaak, 1993). On the other hand since in empirical models the outputs are often linear with respect to parameters, parametric uncertainty can be more easily propagated into the model predictions and traditional robustness tests can be used. These considerations are motivating the development of robust control schemes based on empirical models as in the current work. Assuming that model parametric uncertainty defines a family of models to be referred as an uncertainty set, a robust controller design involves satisfying stability and a level of performance for the entire set of uncertainty. Then, the resulting robustness tests are referred to as robust stability and robust performance tests. To this end robust control tools such as Structured Singular Value (SSV, μ) and Linear Matrix Inequalities (LMI) can be used to formulate such tests.

Various nonlinear robust control design approaches previously reported in the literature, such as designing robust "tubes" around a nominal optimal trajectory (Mayne *et al.*, 2011, Mayne *et al.*, 2005, Mayne *et al.*, 2006) or Newton-type robust algorithms (Diehl *et al.*, 2008a, Zavala and Biegler, 2009), are primarily based on mechanistic (first principles) models where the uncertainty is related to unmeasured disturbances. Thus, in these previously reported approaches parametric uncertainty has not been explicitly considered mostly because, as mentioned above, the models are nonlinear with respect to parameters thus ruling out the application of available robustness tests. To address the problem of parametric uncertainty in nonlinear predictive control Diaz-Mendoza and Budman, 2010b, proposed an algorithm based on a nonlinear empirical Volterra series model where the output is linear with respect to parameters thus allowing for the effect of parametric uncertainty to be accounted for by

calculating a worst prediction error using an SSV norm. Then, optimal control actions were obtained from the minimization of this norm with constraints. However, this earlier approach was found to have disadvantages in terms of conservatism and computational expense. The conservatism of the controller was mostly related to the use of the worst possible error over the horizon as the cost function to be minimized. The high computational cost was due to the time needed to calculate the SSV norm for uncertain model predictions over the entire prediction horizon. The above drawbacks in the work of Diaz-Mendoza and Budman, 2010b has motivated the first part of this research, in which a novel predictive controller was developed by using an empirical Volterra model, as in Diaz-Mendoza and Budman, but where the uncertain coefficients were represented by Polynomial Chaos Expansion (PCE) which significantly facilitates the propagation of parametric uncertainty onto the output. By using PCE, an uncertain output can be represented by a spectral expansion of orthogonal polynomials in terms of of the manipulated variables. Due to the availability of analytical formulae for propagating the effect of uncertainty onto the output, this method has been proposed in the current work as an approach to formulate a robust controller. Furthermore, being PCE's able to quickly produce estimates of the probability density function of the predicted outputs, they open the novel possibility of formulating algorithms that are based on probabilities' distributions which can be potentially less conservative and more realistic as compared to algorithms based on worst bounds. Based on these considerations the current work has investigated different applications of PCE's in nonlinear predictive control algorithms with special emphasis on the following three situations,

1. A nonlinear predictive control based on a Volterra series model where the uncertain parameters are formulated as PCE's.
2. Applications of PCE's in optimization and control of bioreactors where the process model is based on dynamic flux metabolic models which is a mechanistic (first principle model), that is becoming increasingly popular in the pharmaceutical industry,

3. In order to speed up computations and to ease implementation and/or maintenance most of the industrial applications of MPCs rely on implementing distributed control strategies in contrast to one global control strategy. However distributed control systems also may be sensitive to model errors. Distributed controllers are applied to subsets of inputs and outputs which interact with each other. Hence, the overall performance of the system heavily relies not only on the local plant model but also on the interactions between each sub-system. Hence the third objective of this thesis is to design distributed MPC algorithms that are robust to model errors.

Overall the thesis is organized into six chapters. Introduction and research objectives are discussed in Chapter 1. Chapter 2 discusses the background and literature review pertinent for the research objectives, like robust control using LMI's and SSV, Polynomial chaos expansion, control and optimization of bioreactors, Metabolic flux model and distributed model predictive control. Chapters 3-5 are presented in manuscript format and present the findings of three research objectives earlier discussed. Chapter 6 reviews the key contributions of this thesis and how it can be extended in future research.

Chapter 3 discusses the development of a robust nonlinear model predictive controller (NMPC) based on a Volterra series. PCE is used to represent the uncertainty in Volterra series coefficients and this uncertainty is then propagated onto the output predictions, which is used to formulate a $\mathcal{L}^2$-based norm as the cost function. Input constraints and stability of the controller is guaranteed using a SSV-based test. Finally the controller performance and computational time is compared to a SSV-based robust controller and a nominal controller. The results of this chapter have been published in the Journal of Process Control, (Kumar and Budman, 2014).

In chapter 4, the PCE approach is used to develop an offline-robust optimization and an online-robust control methodology for batch biochemical processes, based on dynamic metabolic flux models. The robust controller uses an economic objective function to determine the control actions. The performance of the robust controller is compared to that of nominal controller for several operating conditions and it was found to be superior in terms

of disturbance rejection capabilities. The proposed robust optimization study involves with the maximization of the probability of an end-point property of the batch, based on a PCE-surrogate model of the process. The results of the robust controller case study presented in this chapter have been accepted in ADCHEM 2015 (Refereed Conference Proceedings), and the entire chapter combining the robust control and the robust optimization approaches has been prepared for submission as a journal paper.

Chapter 5 discusses the development of a robust distributed MPC strategy that treats both uncertainty in model error as well as uncertainty related to the information exchanged between the sub-controllers arising due to a loss of communication. Distributed control was pursued as a means to speed up calculations of robust MPC algorithms. The intention was initially to apply the distributed controller with both linear and nonlinear models. However, since the PCE based approach proved to be highly computationally efficient, the distributed controller was developed for linear models only. Accordingly, a robust estimator was developed and combined with a linear distributed MPC that is both robust to model error as well as to errors in the information exchanged between subsystems during periods of communication loss. The performance of the controller that uses a robust estimator is compared to that of nominal estimator for two different processes: i) A classical 2x2 distillation column and ii) A Reactor-separator process consisting of 3 sub-systems, 9 states and 4 manipulated variables. The results and formulation of the distributed controller and the robust estimator have been published in ADCHEM 2012, Kumar *et al.*, 2012. After publication of that work the performance of the robust estimator was further improved by using a bilinear program solver and also by extending the application to the larger reactor-separator system to show online-feasibility of the controller

Refereed Conference Proceedings:

- Kumar, D., Al-Gherwi, W. & Budman, H. 2012. Robust-distributed MPC with robust observer to handle communication loss. *IFAC 2012.* Singapore.
- Kumar, D. & Budman, H. 2015. Robust nonlinear predictive control for a bioreactor based on a Dynamic Metabolic Flux Balance model. *IFAC 2015.* Vancouver, Canada.

Non-refereed Conference Presentations:

- Kumar, D., Al-Gherwi, W. & Budman, H. 2012. Robust-distributed MPC with robust observer to handle communication loss. 62nd Canadian Chemical Engineering Conference, Vancouver, October 14-17.
- Kumar, D. & Budman, H. 2014. Robust nonlinear MPC based on volterra series and polynomial chaos expansions. 61st Canadian Chemical Engineering Conference, London, October 23-26.

Chapter 2
Background and Literature Review

2.1 Model Predictive Control

Model Predictive Control (MPC) is a widely used control design technique in the process industries. MPC performs predictions that can be based on either a linear or nonlinear model. When a nonlinear model is used for prediction the resulting control strategy is referred to as Nonlinear Model Predictive Control (NMPC). As explained in Chapter 1, , since plant-model mismatch is unavoidable, it becomes important to design a control strategy which is robust to model uncertainty. While there exist a variety of techniques to analyze robustness of controllers that are based on linear models, the design of robust nonlinear controllers such as NMPC is challenging and it is currently an active field of research (Allgower *et al.*, 2004, Magni and Scattolini, 2010). Most of the robust control techniques developed for linear system, like SSV (Structured Singular Value or μ) and LMI's (Linear Matrix Inequalities) based tests, are not directly applicable to nonlinear systems. On the other hand, the use of a nonlinear model for prediction within a predictive control algorithm is desirable since it is expected to result in better closed loop performance when controlling chemical processes which are generally highly nonlinear (Allgower *et al.*, 2004). Accounting for robustness to model errors has been identified as one of the key challenges in the research of NMPC controllers (Allgower *et al.*, 2004).

In general model predictive control refers to a class of control algorithms in which a performance criterion is optimised along a prediction horizon while satisfying a set of input/output constraints. A nominal dynamic process model, linear or nonlinear, is used to predict the outputs along the horizon and a norm of these outputs is used to quantify the closed loop performance. While the decision variables for optimization in MPC may consist of a certain number of future control moves, where this number is referred to as the control horizon, only the first control action is actually implemented into the plant and the optimization is then solved all over again in the following time interval (Findeisen *et al.*, 2003, Henson, 1998). By repeating the calculation at every time step the controller is able to

correct for unmeasured disturbances that may enter the process at each new time interval and to compensate for the effect of model error that leads to the incorrect predictions obtained with the nominal model. In a Linear MPC algorithm, the nominal model used for prediction is linear with respect to the manipulated variables and correspondingly, the input and output constraints are also formulated as linear equalities or inequalities. The cost function in linear MPC is generally quadratic with respect to the inputs thus the corresponding optimization problem is a QP (quadratic programming) for which a global optimum can be found in a finite number of iterations. A comprehensive review of Linear MPC, its development and applications are provided in Qin and Badgwell, 2003. On the other hand, for NMPC the nominal model used for prediction as well as the constraints may be linear or nonlinear with respect to the manipulated variables. In general, MPC scheme which are based on either a nonlinear process model or nonlinear input/output constraints or with a non-quadratic cost function are considered as NMPC strategies. The main steps of an NMPC algorithm are as follows:

1. Using a nonlinear model of the form,

$$\dot{x} = f\big(x(t), u(t)\big)$$
$$\hat{y} = g(x, u) \qquad 2.1$$

   where x is plant states, and y is plant output, f and g are nonlinear vector functions of plant inputs and outputs and starting from a current output measurement y_k to correct for unmeasured disturbances, future predictions of the outputs are generated over a prediction horizon, p, using as a function of a sequence of inputs defined over a control horizon m, where $m \leq p$.

2. The cost function to be minimized with respect to the decision variables, i.e. the optimal control actions, is in general assumed as a weighted sum of the errors in future predictions with respect to a reference trajectory (or set-point profile) and plant inputs, where the latter are included in the cost to avoid excessive control actions. Accordingly, the typical cost used in predictive control algorithms is as follows:

$$J = \min_{\boldsymbol{u(k+1|k)},\boldsymbol{u(k+2|k)},\ldots,\boldsymbol{u(k+m|k)}} \sum_{i=1}^{i=p} [(\hat{\boldsymbol{y}}(k+i|k) - \boldsymbol{y}^{sp})^T \boldsymbol{Q} (\hat{\boldsymbol{y}}(k+i|k) - \boldsymbol{y}^{\boldsymbol{sp}})] + \sum_{i=1}^{m} \boldsymbol{u}(k+i|k)^T \boldsymbol{R} \boldsymbol{u}(k+i|k) \quad 2.2$$

Where p, m, Q, R are tuning parameters.

3. The solution of the optimisation problem shown in (2.2) is m control actions but as explained before only the first control action, i.e. $u(k+1|k)$ is implemented in the plant.

4. After implementation of the control action, new plant measurements are obtained and step 1 to 4 are repeated for the next sampling interval.

The algorithm outlined above does not guarantee stability unless additional stability constraints are enforced. One way proposed in the literature to ensure stability is to require that the error between the predictions and setpoints are eliminated at the end of an infinite control horizon. This approach, referred to as a terminal constraint condition, is computationally demanding (Findeisen *et al.*, 2003, Chen and Allgower, 1998) and it may be infeasible in the presence of input constraints. Two possible ways to enforce stability while maintaining a finite control horizon have been suggested:

- Inclusion of a terminal equality constraint within the optimization problem as per equation (2.3). However, the solution of the optimisation problem (2.2) in the presence of this constraint becomes computationally expensive and in some cases may be infeasible and hence is not desirable.

$$\hat{\boldsymbol{y}}(k+p|k) - \boldsymbol{y}^{\boldsymbol{sp}}(k+i) = 0 \quad 2.3$$

- A second approach is to include a terminal inequality constraint and terminal cost E, into problem (2.2). The terminal inequality constraint forces the terminal output prediction to be within certain prespecified error, δ, of the reference trajectory rather than to be equal to the reference as in the previous approach. Then, the error between

the terminal output prediction and the reference is penalized within the cost function as follows (2.4).

$$J = \min_{\substack{\boldsymbol{u(k+1|k)},\\ \boldsymbol{u(k+2|k)},\ldots,\\ \boldsymbol{u(k+m|k)}}} \sum_{i=1}^{i=p} [(\hat{\boldsymbol{y}}(k+i|k) - \boldsymbol{y}^{sp})^{\boldsymbol{T}} \boldsymbol{Q} (\hat{\boldsymbol{y}}(k+i|k) - \boldsymbol{y}^{sp})]$$
$$+ \sum_{i=1}^{m} [\boldsymbol{u}(k+i|k)^T \boldsymbol{R} \boldsymbol{u}(k+i|k)] \qquad 2.4$$
$$+ (\hat{\boldsymbol{y}}(k+p|k) - \boldsymbol{y}^{sp})^T \boldsymbol{E} (\hat{\boldsymbol{y}}(k+p|k) - \boldsymbol{y}^{sp})$$

$$\mathcal{E}: \{y | \|y - y^{sp}\|_2 \leq \delta\}$$

Several studies have focused on the design of the terminal region $\mathcal{E}$, and terminal penalty, $\boldsymbol{E}$ (Chen and Allgower, 1998, Michalska and Mayne, 1993). The terminal penalty weight $\boldsymbol{E}$, forces the output to reach the region $\mathcal{E}$, and if $\mathcal{E}$ is selected properly then the plant output will eventually converge to the reference trajectory.

Robustness remains a key challenge for the design of NMPC algorithms (Findeisen *et al.*, 2003, Magni and Scattolini, 2010). Model/plant mismatch arises due to inaccurate knowledge of parameters, uncertainty resulting from simplifications regarding model structure or model reduction, disturbances in the plant or lack of knowledge about certain physical mechanisms of the process like interactions among systems. Hence, robustness needs to be addressed explicitly in the design of NMPC schemes.

2.2 Robust Nonlinear Model Predictive Control (MPC)

The conservatism of a robust MPC controller is directly related to the level of model error considered in the design. If the uncertainty description is overly conservative the resulting robust controller will be also conservative. Accordingly it is desired to select a nominal model for prediction that is accurate enough so the associated uncertainty is small. Two desired properties of closed-loop system are robust stability and robust performance. Thus, along with the properties of nominal stability and nominal performance to be satisfied when the uncertainty is ignored, robust stability and robust performance must be also satisfied in

the presence of model error. Different mathematical tools are available to test for robust stability and performance that are reviewed in the following sections.

2.2.1 LMI's for Robust Control

Linear Matrix Inequalities (LMI's) provides a means of formulating various inequalities related to stability and performance conditions as convex constraints. LMI's are an attractive choice for solving complex problems because they can be solved using convex optimization algorithms (VanAntwerp and Braatz, 2000, Boyd, 1994). In control theory, three main types of problems are solved using LMI, i) Feasibility problem ii) Generalised Eigenvalue problem, and iii) Linear programming problem. Since robust performance and stability tests together with input and output constraints can be formulated as LMI's (Kothare *et al.*, 1996), this mathematical tool has gained significant interest in the control community. In general a linear matrix inequality can be expressed as follows:

$$F(x) = F_0 + \sum_{i=0}^{m} x_i F_i > 0 \qquad 2.5$$

where $F_i \in \mathbb{R}^{nxn}, i = 0, 1, \dots, m$ are symmetric and real matrices and defined by the problem, $x \in \mathbb{R}^m$ is a variable, and the inequality means that $F(x)$is a positive definite matrix. When the problem involves multiple LMI's, it can be converted into a single LMI of higher dimension as follows: $Given: F^1(x), F^2(x), \dots, F^p(x) > 0,$ becomes $diag(F^1(x), F^2(x), \dots, F^p(x)) > 0$. The Schur's complement Lemma can be used to convert nonlinear constraints occurring in control problems to an LMI as follows:

$$\text{Given: } R(x) > 0, Q(x) - S(x)R(x)^{-1}S(x)^T > 0 \qquad 2.6$$

where, $Q(x)$ and $R(x)$ are symmetric and $S(x)$ depends on x affinely. Then the Schur's complement lemma can be used to convert equation (2.6) to an equivalent LMI (2.7). Proof for this lemma can be found in VanAntwerp and Braatz, 2000.

$$\begin{bmatrix} Q(x) & S(x) \\ S(x)^T & R(x) \end{bmatrix} > 0 \qquad 2.7$$

Kothare *et al.*, 1996, proposed a formal theoretical approach for synthesis of robust MPC by using an infinite horizon and for different forms of model uncertainty (polytope and structured uncertainty). This approach was extended in the same work to include input/output constraints using a norm approach as proposed in (Boyd, 1994). Al-Gherwi *et al.*, 2011 extended this work to the design of Robust Distributed MPC for polytopic uncertainty in both time-varying and time-invariant models, where the term *distributed* refers to the application of several MPC's to different subsets of inputs and outputs while communication is exchanged among these controllers. In the current work, the topic of loss of communication has been addressed in the algorithm of Al-Gherwi *et al.*, 2011 with a robust estimator based on LMI's. Distributed MPC has been studied in this work as a possible way to reduce complexity of computation in the algorithms to be investigated in the current work. LMI's are widely used for robust control of linear systems. They have also been considered to investigate robustness of nonlinear processes where the nonlinearity is approximated by uncertainty polytopes with respect to a nominal linear model. However, identifying polytopic uncertainty to describe the actual nonlinear process is challenging and may result in overly conservative uncertainty descriptions (Doyle *et al.*, 1989). For instance one possibility to bound the nonlinearity is to calculate bounds on the terms of the Jacobian matrix of the nonlinear model describing the process but this generally results in conservative uncertainty descriptions (VanAntwerp and Braatz, 2000).

2.2.2 SSV for Robust Control

The Structured Singular Value (SSV) norm also referred as μ is an additional mathematical tool developed for assessing stability and performance of controllers based on uncertain models with either structured and unstructured uncertainties. When using μ norms based analysis, the idea is to split the uncertain part of the model from the nominal part of the model and then develop a Linear Fractional Transformation (LFT) relation between the inputs and outputs that can be schematically described by an interconnection matrix given in Figure 2.1. Then for a given structure of the resulting interconnection matrix (M) and the uncertainty description Δ, the μ norm provides a measure of the smallest perturbation within the given uncertainty set, that can destabilize the plant. It is also possible to use this norm to

test robust performance by calculating a norm based bound on the outputs for bounded closed loop inputs, i.e. disturbances and set-points. This feature will be used in this work to calculate worst deviations of the output with respect to the set-point along the prediction horizon in the predictive control strategy. The definition of the SSV norm is as follows.

$$\mu_{\Delta}(\boldsymbol{M}) = \frac{1}{\min\limits_{\Delta \in \boldsymbol{\Delta}} \{\bar{\sigma}(\boldsymbol{\Delta}) | \det(\boldsymbol{I} - \boldsymbol{M\Delta}) = 0\}} \tag{2.8}$$

In the case there is no $\Delta \in \boldsymbol{\Delta}$, for which $\det(\boldsymbol{I} - \boldsymbol{M\Delta})$ becomes singular, then $\mu_{\boldsymbol{\Delta}}(\boldsymbol{M}) = 0$. A key advantage of μ norms as compared to general singular value norms is that it explicitly accounts for the structure in the uncertainty thus producing less conservative bounds. As a result, less conservative controllers can be obtained using μ (Bates and Postlethwaite, 2002) as compared to designs that are based on norms that do not take the structure of the uncertainty into account. Since Δ contains information about both structured and unstructured uncertainty, the Δ block used for calculation of the SSV includes both scalar as well as complex uncertainty elements as follows:

$$\boldsymbol{\Delta} = \{diag[\delta_1 \boldsymbol{I_{r1}}, \dots, \delta_S \boldsymbol{I_{rS}}, \boldsymbol{\Delta_{S+1}}, \dots, \boldsymbol{\Delta_{S+F}}] : \delta_i \in \mathbb{C}, \boldsymbol{\Delta_{S+j}} \in \mathbb{C}^{m_j x m_j}, 1 \leq i \leq S, 1 \leq j \leq F \tag{2.9}$$

where S and F are repeated scalar blocks and full complex blocks respectively.

As mentioned above, robust stability and performance tests can be formulated in terms of μ norm. For the example in Figure 2.1, $d$and r are the exogenous inputs and outputs respectively. In closed loop the exogeneous inputs are set-points and disturbances whereas the outputs are the controlled variables or other variables that should be kept within limits.

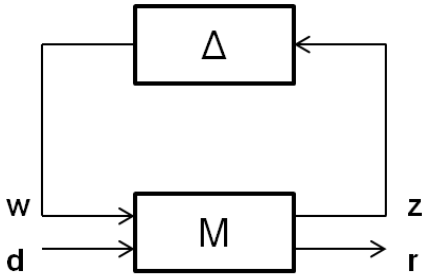


Figure 2.1 LFT between exogenous input and output

After partitioning M with dimensions compatible with Δ, the equations describing the system are

$$\begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M_{11}} & \boldsymbol{M_{12}} \\ \boldsymbol{M_{21}} & \boldsymbol{M_{22}} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{d} \end{bmatrix}$$

$$w = \Delta z \qquad 2.10$$

$$d = F_u(\boldsymbol{M}, \boldsymbol{\Delta})\boldsymbol{r} = (\boldsymbol{M_{22}} + \boldsymbol{M_{21}}\boldsymbol{\Delta}(\boldsymbol{I} - \boldsymbol{M_{11}}\boldsymbol{\Delta})^{-1}\boldsymbol{M_{12}})\boldsymbol{r}$$

The input-output relationship derived using the LFT can be chosen to investigate robust stability or robust performance. The worst case bound related to the input-output relationship defined in 2.10 can be calculated with $\mu_{\Delta}(\boldsymbol{M})$.

2.2.3 Literature Review on Robust NMPC

Design of robust nonlinear predictive controllers has been generally based on the solution of a minmax problem, where the maximisation is done with respect to model uncertainties or disturbances and the minimisation of the cost is done to determine the control actions (Magni and Scattolini, 2010, Findeisen *et al.*, 2003). A recently proposed approach, referred to as the "tube" approach, has been proposed to determine nominal inputs (control actions) using an auxiliary controller based on a nominal model and then input deviations from the nominal inputs are calculated to guarantee that actual outputs' trajectories are bounded within tubes around the nominal output trajectories (Mayne *et al.*, 2011, Magni *et al.*, 2006). As such the algorithm can be viewed as a combination of a feedforward controller corresponding to the calculations performed with the nominal model and a feedback controller corresponding to the calculations related to the deviations form the nominal trajectories. The tubes around the

nominal trajectories are determined at every time step by including certain state constraints necessary to ensure Lyapunov stability in the presence of bounded disturbances. Variations of this scheme have been proposed involving various modifications of the cost function, e.g. by including the terminal equality constraint only in the auxilliary controller rather than in the entire cost, (Mayne, 2011) by optimising for the initial condition used in the cost function (Mayne *et al.*, 2005), or by including a robustly stable observer (Mayne *et al.*, 2006). An additional modification of the tubes' approach (Cannon *et al.*, 2011) involved successive linearization of the nonlinear model for every prediction in the horizon. Then, the effects of model errors on cost function and constraints were bounded using robust tubes and the sizes of the tubes were included as decision variables in the optimisation problem. In general, algorithms based on the "tube" approach have used mechanistic models and they have only considered uncertainty in disturbances thus making it difficult to generalize this analysis to any type of nonlinearity and to parametric uncertainty. For these reasons the tubes' approach has not been adopted in the current project where a key objective is to design a robust NMPC in the presence of parametric uncertainty.

An alternative approach to the tubes' method that has been gaining significant attention for tackling the problem of robust NMPC involves simplification of the optimisation problem to obtain a faster online solution. For example, Diehl *et al.*, 2008b represented parametric uncertainty by a bounded set, replaced the inner maximisation problem with necessary first order optimality conditions and then assumed that the worst case solution occurs on the bounds of the uncertainty set. However, this approach assumes that a perfect model is available and it requires a mechanistic model along with the derivatives of the objective function, constraints and uncertainty set. Zavala and Biegler, 2009 compute a preliminary estimate of the control actions at a given time step by using the nominal model. In this work, computation was sped-up by determining the nominal solution in between sampling times. Finally a quick correction to the nominal solution was provided by NLP (Nonlinear programming) sensitivity concepts. Yet again parametric uncertainty was not considered in this approach and the model for prediction was a mechanistic one.

On the other hand, minmax formulations dealing with parametric uncertainty as proposed in the current work have been few. Regarding optimization algorithms in the presence of uncertainty based on the minmax formulation two different approaches have been proposed in the literature, (i). A simulation based approach involving the minimization of (Kawohl *et al.*, 2007) the weighted contribution of the first two statistical moments of an objective function using Monte Carlo simulations, where the goal was robust optimization under uncertainty rather than control per se; (ii)- an analytical approach in which parametric uncertainty is propagated using Taylor series, and the worst case deviation as determined by using Structured Singular Values (SSV) (Ma and Braatz, 2001, Ma *et al.*, 1999, Nagy and Braatz, 2003) is minimized with respect to the decision variables. This latter methodology was found useful when first order estimates are not sufficient to quantify uncertainty and second order or higher order estimates are required for determining worst case scenario in the presence of model uncertainty. Using this method Diaz-Mendoza and Budman, 2010b presented an RNMPC strategy based on SSV norms for continuous processes, in which the cost function is formulated as a function of an SSV norm where the latter provides a bound on the worst possible output deviation with respect to the setpoint in the presence of model errors. The minimization of this cost with respect to the future control actions, ensured also satisfaction of inputs and terminal constraints. This algorithm was based on empirical Volterra models with uncertain coefficients. Empirical models were selected in that previous work as well as in the current study because, as mentioned above, it is difficult to formulate a general robust approach for mechanistic models due to the various forms in which nonlinearities appear in these models. Since computation of the SSV-norm is a NP-hard type of problem, the computational time of the algorithm previously proposed by Diaz-Mendoza and Budman, 2010b increases exponentially as the dimensions of the process in terms of inputs and outputs increase. Also since the controller was based on worst case outputs' deviations with respect to the set-point it resulted in conservative performance.

To address these limitations, in the current work an alternative approach is proposed whereby parametric uncertainty is propagated onto the outputs using Polynomial Chaos Expansions (PCE). Few studies on the use of PCE for control design have been reported. For example in

Hover and Triantafyllou, 2006 a PCE was used to perform stability analysis of a particular nonlinear system with random initial conditions or random parameters and to propagate the uncertainty onto the output, thereby reducing computational time as compared to an alternative Monte Carlo simulations' based approach. In Smith *et al.*, 2009 an LQG controller was designed based on PCE approximations of the parametric uncertainty, but that study was limited to linear systems.

2.3 Polynomial Chaos Expansion

A Polynomial Chaos Expansion (PCE) describes a random process as a spectral expansion of random variables(θ_i), using orthogonal basis functions, Φ_i (Ghanem and Spanos, 1990, Ghanem and Spanos, 1997). For example, any second-order (finite variance) random process y^d, can be described using a PCE as follows:

$$y^d = a_0^d\Phi_0 + \sum_{i_1=1}^{\infty} a_{i1}^d\Phi_1(\theta_{i_1}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2}^d\Phi_2(\theta_{i_1},\theta_{i_2}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3}^d\Phi_3(\theta_{i_1},\theta_{i_2},\theta_{i_3}) \quad 2.11$$

where $a_{i1}^d$are deterministic coefficients for each term in the expansion. Since the basis functions Φ_i, are orthogonal, the first term in (2.11), a_0^d is the nominal value of $y^d$and its variance can be obtained from $\sum_{i=1}^{\infty}(a_{i1}^d)^2 + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}(a_{i_1 i_2}^d)^2 + \ldots$. Based on the Cameron-Martin theorem these expansions are convergent in the $\mathcal{L}^2$-norm (Xiu and Karniadakis, 2002, Xiu, 2010, Najm, 2009) and for practical application they can be truncated to a finite number of terms. When a random process is described via a truncated PCE, then the dimensionality n_0 and the maximum polynomial order for the basis function, q, need to be defined. The number of independent sources of random variables ($\theta_{i_1}, \theta_{i_2}, \theta_{i_3}$), generally defines the dimensionality, n_0. The number of terms in expansion P_{PCE}, is then given by $(n_0+q)!/(n_0!\,q!) - 1$. Using these notations a truncated PCE expansion can be represented as follows:

$$y^d \approx \sum_{i=1}^{P_{PCE}} a_i^d \Phi_i(\theta) \qquad 2.12$$

Due to the orthogonality of the basis functions, Φ_i, inner product of polynomial functions in the space spanned by the basis functions $\{\Phi_i\}_{i=0}^{P_{PCE}}$, is non-zero only with respect to the same Φ_i.

$$\langle \Phi_i \Phi_j \rangle = \int \Phi_i(\theta)\, \Phi_j(\theta) \rho_\theta(\theta) d\theta = \delta_{ij} \langle \Phi_i^2 \rangle \qquad 2.13$$

This orthogonality property is the basis of the calculation of the coefficients when propagating uncertainty from the input random variables$(\theta_{i_1}, \theta_{i_2}, \theta_{i_3})$, to the output random variables (y^d). The choice of the basis functions Φ_i depends on the type of stochastic distribution to be represented, i.e. normal or uniform. In Xiu and Karniadakis, 2002, Xiu, 2010, Xiu and Tartakovsky, 2004 the Askey scheme is proposed which determines the optimum polynomial functions to be used for each type of stochastic distribution to be modeled. For example if the random variables (θ_i), are of Gaussian distribution then Hermite polynomials will describe the probability distribution with the least number of terms, because the weighting function of Hermite polynomials are the same as for the Gaussian probability density function. However, if Hermite polynomials are used to describe non-Gaussian behavior then the expansion would require second-order polynomial terms as well.

The coefficients of the PCE's which are used to approximate particular data are calculated as follows. Given a process model with uncertain output, $y = f(x; \lambda)$, where x is the uncertain input and λ is the uncertain parameter, the aim is to quantify uncertainty in $y(\theta)$ from $x(\theta), \lambda(\theta)$ using the process model. Then the first step is to construct PCE's of $x(\theta)$, and $\lambda(\theta)$, by determining their PCE coefficients x_i and λ_i.

$$\begin{aligned} x(\theta) &= \sum_{i=1}^{P_{PCE}} x_i \Phi_i(\theta) & \boldsymbol{x_i} &= \frac{\int \boldsymbol{x \phi_i(\theta) g(\theta) d\theta}}{\langle \boldsymbol{\Phi_i^2} \rangle} \\ \lambda(\theta) &= \sum_{i=1}^{P_{PCE}} \lambda_i \Phi_i(\theta) & \boldsymbol{\lambda_i} &= \frac{\int \boldsymbol{\lambda \phi_i(\theta) g(\theta) d\theta}}{\langle \boldsymbol{\Phi_i^2} \rangle} \end{aligned} \qquad 2.14$$

where $g(\theta)$ is probability distribution function (pdf) of θ. Since x and λ are not directly

related to θ, so a map has to be built for evaluation of x_i and λ_i. This is done by transforming both the uncertainty θ and uncertain parameter x_i to another uniformly distributed space, $U(0,1)$, such that it represents the cumulative distribution function (CDF) of both x and θ Xiu and Karniadakis, 2002.

$$u = \int_{-\infty}^{x} f(x)dx = \int_{-\infty}^{\theta} g(\theta)d\theta \qquad 2.15$$

where $f(x)$ is the pdf of x and u is the uniformly distributed variable. After determining x (and λ) and θ for the corresponding u, x_i(and λ_i) can be determined using (2.14). The next step is to develop PCE for $y(\theta)$ from $x(\theta)$ and $\lambda(\theta)$, which can be done by evaluating the inner product of $y(\theta)$with each basis functions Φ_i to determine the i^{th}- PCE coefficient y_i.

$$y_i = \frac{\langle y\Phi_{\mathrm{i}}\rangle}{\langle \Phi_{\mathrm{i}}^2\rangle} = \frac{\langle f(x;\lambda)\Phi_{\mathrm{i}}\rangle}{\langle \Phi_{\mathrm{i}}^2\rangle} \qquad 2.16$$

Evaluating the inner product $\langle y\Phi_{\mathrm{i}}\rangle$, requires computation of multi-dimensional integrals which can be performed by one of two approaches referred to as non-intrusive and intrusive. Non-intrusive, as the name suggests, estimates the integral based on N samples of the whole space of basis functions and evaluates $\langle y\Phi_{\mathrm{i}}\rangle$ at those predetermined points θ_j according to Eq. (2.17). Non-intrusive techniques require 3 steps to determine (Najm, 2009), y_i as per Eq. 2.17, i) a methodology to generate the N samples for θ_j ii) evaluating the function $y^j = f(x;\lambda)$ at θ_j , and iii) evaluating the integral represented in Eq. 2.17, using numerical techniques. The sampling techniques can be random based sampling like Monte Carlo (MC), or probability based methods or quadrature based methods like Smolyak or different collocation methods have been suggested to this end. A thorough review of existing non-intrusive techniques can be found elsewhere (Najm, 2009). In the current work, Gaussian quadratures have been used to develop surrogate models based on non-intrusive methods. The computational demand of developing non-intrusive based PCE's is determined by $y^j = f(x;\lambda)$ at θ_j. Thus the choice of sampling strategy determines the computation load for non-intrusive techniques.

$$y_i = \frac{1}{\langle \Phi_i^2 \rangle} \frac{1}{N} \sum_{j=1}^{N} y^j \Phi_i(\theta_j), \qquad i = 1, 2, \dots, P_{PCE} \tag{2.17}$$

In the intrusive method the integral is evaluated by using the Galerkin projection approach and by employing the property of orthogonality of the basis functions. For example if $y = \lambda x$, then after substituting the PCE's for x and λ in (2.16), the PCE coefficients y_i for the output, is as follows:

$$y_i = \sum_{j=1}^{P_{PCE}} \sum_{k=1}^{P_{PCE}} \lambda_j x_k \frac{\langle \Phi_i \Phi_j \Phi_k \rangle}{\langle \Phi_i^2 \rangle}, \qquad i = 1, 2, \dots, P_{PCE} \tag{2.18}$$

The tensor

$$C_{ijk} = \frac{\langle \Phi_i \Phi_j \Phi_k \rangle}{\langle \Phi_i^2 \rangle} \tag{2.19}$$

is a known property of the basis functions, which can be computed once and stored for offline computations. This approach however needs modification depending on the different forms of nonlinearity occurring in the process model.

2.4 Bioreactor control and optimization

A main application to be investigated in this work is control of bioreactors. Modeling of biological systems is generally very challenging due to the inherent nonlinear behavior of these systems and to the inaccuracy in data used for model calibration. Hence, the uncertainty associated to these models is generally very large. Accordingly, robust control design is of key importance for this type of application. Bioreactors have traditionally been operated in three different modes, batch, fed-batch and continuous. The most popular mode of operation in the pharmaceutical industry has been the fed-batch mode (eg. production of baker's yeast, food additives and penicillin production), in which substrate is slowly fed to the reactor and the product is only drawn at the end of the batch. This mode of operation helps mitigating the inhibitory effects of high substrate concentration or by-products such as ammonia or lactate. For example, high ethanol concentration inhibits yeast growth, so by operating the reactor in fed-batch mode ethanol production can be enhanced since the substrate is supplied gradually

rather than in high concentration at the start of the batch. Also since the product is drawn only at the end, it becomes easier to maintain sterilized conditions, (Rani and Rao, 1999). Therefore, since pharmaceutical applications will be considered for this study the focus of this part of the review is on fed-batch reactor operation and control.

Traditionally fed-batch bioreactors have been controlled via simple PID controllers, to maintain operating temperature, pH or dissolved oxygen (DO) concentration at their set-points (Lee *et al.*, 1999) by manipulating the substrate feed-rate. This limited control strategies has been the norm in the industry mostly because of two factors: i) Poorly known nonlinear dynamics of bioprocesses and the estimated parameters ii) Measurements of metabolites are rarely available on-line and the available measurements, e.g. pH and oxygen, are insufficient for estimation of the states. Product/cell concentration measurements are usually done offline, because of unreliable online analyzers (Lubbert and Jorgensen, 2001, Rani and Rao, 1999, Smets *et al.*, 2004). However recent advances in online measurements combined with stringent FDA guidelines regarding bioprocess operations may facilitate future implementations of model-based controllers for biochemical processes (Henson, 2010).

A common fed-batch optimization problem has been that of determining substrate feeding policy by optimizing the final product concentration or a cost function. This objective can be achieved offline via dynamic optimization and is applied online to a fed-batch reactor in an open-loop fashion, i.e. without accounting for feedback errors. To this end numerous studies have been proposed (Frahm *et al.*, 2002, Hjersted and Henson, 2006, Banga *et al.*, 1997) using different optimization problems. Banga *et al.*, 1997 recognized the drawbacks of not including feedback and suggested to perform periodic online recalculation of feed-profiles especially in the presence of large disturbances.

Very few studies have included the feedback error within the optimal control problem (Smets *et al.*, 2004 and Chen *et al.*, 1995). These studies are based on the idea that Fed-batch reactor controllers do not have to globally stabilize the system, but instead keeps the unstable system under control for the duration of the process which is finite, (Smets *et al.*, 2004 and

Chen *et al.*, 1995). Chen *et al.*, 1995 have presented a nonlinear adaptive control strategy based on the feedback linearization idea that was used to simplify the model relating product concentration to substrate feed rate. Smets *et al.*, 2004 developed an optimal adaptive control strategy, in which a suboptimal solution to the cost function is used to formulate a nonlinear linearizing controller. Also biomass concentration measurements are used to estimate the specific growth rate using an observer. The importance of robustness for fed-batch bioreactor control has been stressed in (Kuhlmann *et al.*, 1998), due to i) parametric uncertainty occurring because parameters are generally identified as time invariant though in reality they should be considered time varying, e.g. due to biological adaptation mechanisms of the cells ii) unmodeled dynamics, iii) large disturbances occurring in the process.

To address robustness, Renard *et al.*, 2006 proposed a robust controller based on a Youla parametrization in which two stable transfer functions are designed one for rejecting disturbances during the exponential cell growth and one for robustness against unstructured uncertainties. However this technique did not include parametric uncertainty and also the controller was applied to an operating condition where the linear process model was considered sufficiently accurate to describe the process dynamics.

However, bioreactors' operations in the pharmaceutical industries generally lack sophisticated online-measurement techniques required for implementation of online control and are often limited by certification procedures in terms of the types of control and monitoring systems that they can rely upon. In these cases off-line optimization would be preferred over on-line feedback control strategies. Then, a single offline-robust optimization calculation can be performed to obtain an optimal feeding recipe. Hence it becomes important to consider the effect of plant-model mismatch and disturbances (Srinivasan *et al.*, 2003) on recipes resulting from robust optimization calculations. For most of the off-line robust optimization techniques uncertainty propagation methods are required and then a probability distribution of the objective function is used to define the cost. Studies have been proposed where the objective function consists of *i)* the expected or extremum value of a terminal property Dewasme *et al.*, 2011, *ii)* a worst-case scenario of the cost, Ma *et al.*, 1999, *iii)* a weighted function of the expected value and variance of the terminal property, Nagy

and Braatz, 2003, Nagy and Braatz, 2004, *iv)* a probabilistic objective function to meet a certain quality criteria, Terwiesch *et al.*, 1998, or *v)* a linearization of the objective function around the nominal conditions combined with bounds of the model uncertainties Logist *et al.*, 2011. Most of these mentioned studies either rely on first-principles model for uncertainty quantification and propagation or use Monte Carlo sampling methods (which is computationally heavy). All constraints in these formulations are transformed to corresponding robust counterparts. On the other hand if measurements are available along the batch, they can be used to counter the effect of uncertainties by adapting the model to be used for subsequent batches (Mandur and Budman, 2015, Srinivasan and Bonvin, 2007, Srinivasan *et al.*, 2003). In the current work, PCE's is used for uncertainty quantification and propagation, which are known to facilitate quick computation of statistical measures and uncertainty propagation.

2.5 Metabolic Flux Model

Model based control of bioreactors requires an appropriate dynamic model of the process. Dynamic models for bioreactors can be generally classified as unstructured and structured based on the amount of detail included in the model regarding to cellular metabolism. The unstructured models are generally based on simplistic substrate and biomass balances but they do not account for the detailed interaction existent between the different nutrients such as amino-acids, e.g glucose, glutamine, aspargine, and by-products such as ammonia, lactate and carbon dioxide. A general dynamic unstructured model for a substrate inhibited enzyme kinetic model is shown in 2.20. On the other hand structured models are referred as such since they are based on the metabolic reactions corresponding to the organism under study. Hence, structured models are always more complex than unstructured ones but they correctly describe the relations between the different metabolites participating in the process.

$$\frac{dV}{dt} = F\,, \qquad \frac{d[X]}{dt} = \mu[X], \qquad \frac{d[S]}{dt} = F[S]_{in} - \frac{\mu X}{Y_{X|S}}\,, \qquad \frac{d[P]}{dt} = \frac{\mu[X]Y_{P|S}}{Y_{X|S}}\,, \tag{2.20}$$

F Feed Rate, V Batch Volume
$[X]$ Concentration of Biomass
μ Rate of cell growth,
$[S]$ Substrate concentration
$[P]$ Product concentration
μ_{max} Maximum growth rate

$$\mu = \frac{\mu_{max}[S]}{K_m + [S] + [S]^2/K_I}$$

K_m Substrate Saturation constant
K_I Substrate inhibition constant
$Y_{X|S}, Y_{P|S}$ Yield coefficients

Metabolic flux analysis (MFA) modeling is a method to develop structured models based on flux balance of metabolites at quasi-steady state (Varma and Palsson, 1994). It is assumed that intracellular metabolite dynamics are much faster than the cell growth and the dynamics of extracellular metabolites thus justifying a quasi steady-state assumption for the intracellular species. MFA consists of formulating a stoichiometric matrix ($\boldsymbol{A_{mXn}}$) for the corresponding vector of reaction fluxes ($\boldsymbol{\nu_{nx1}}$) and formulating mass balances of extracellular metabolites as per the following equation:

$$\boldsymbol{A\nu = b} \qquad 2.21$$

where b_{mx1}represents a vector of consumption or production rate of extracellular metabolites, i.e. nutrients and by-products. Generally the number of metabolites is less than the number of fluxes. Thus, equation 2.22 describes an under-determined system of equations to solve for the fluxes. To convert the problem into a determined one, assumptions or additional constraints have to be made. Varma and Palsson, 1994 suggested that it is reasonable to assume that the organism as a result of natural evolution is continuously trying to maximize growth and allocate resources in order to accomplish this task. This assumption is especially reasonable for bacterial cells whereas in mammalian cells a large amount of nutrients are consumed to provide for maintenance energy of the cells. Following this assumption, and assuming that the growth μ is to be maximized, the problem can be approached as a Linear Programming (LP) problem, with the flux balances' equation (2.22) imposed as constraints:

$$\mu = \max_{\nu_i} \sum_{i=1}^{n} w_i \nu_i \qquad 2.22$$

$$st.\ \boldsymbol{A\nu = b}$$

Then, assuming that at every time step the growth rate is maximized by the organism the consumption or production of species can be calculated with time as follows:

$$\max_{x,\boldsymbol{v}} \mu$$

$$s.t.\ \frac{d\boldsymbol{z}}{dt} = \boldsymbol{A}\boldsymbol{v}x, \qquad \frac{dx}{dt} = \mu x, \qquad \mu = \boldsymbol{w}^T\boldsymbol{v} \tag{2.23}$$

Where, x and $\boldsymbol{z}$ are the current biomass and metabolites' concentrations respectively. This dynamic modeling approach of the cell metabolism, referred to as Dynamic Flux Balance Modeling (DFBM) has been applied successfully by Mahadevan *et al.*, 2002, to explain the microbial growth of *Escherichia coli* in a batch reactor. Hjersted and Henson, 2006, have used DFBM to determine the glucose (substrate) feeding policy for a fed-batch bioreactor producing ethanol with the yeast *Saccharomyces cerevisiae*. In both Mahadevan and Hjersted studies, additional kinetic rate constraints are imposed in order to achieve realistic flux distribution, v and metabolite concentration,z. It should be noticed that dynamic flux models have not been used as yet for predictive control as proposed in the current work. Since these models are gaining increasing acceptability by the pharmaceutical research community it is very timely to investigate their application for control and optimization.

2.6 Robust Distributed MPC with loss of communication

One of the key challenges for the industrial implementation of robust MPC algorithms is the high computational costs related to the online calculations required for testing robustness. Previous work by our research group has demonstrated that the real time implementation of robust predictive controllers may become prohibitive when dealing with systems of large dimensions, many inputs and outputs, either with SSV based tests (Diaz-Mendoza and Budman, 2010b) or LMI based approaches (Al-Gherwi *et al.*, 2011). The use of PCEs for fast on-line calculations of output variance proposed in Chapter 3 is one of the approaches pursued in this thesis to reduce the computational burden of MPC. However, there is a good motivation to search for additional methods to speed up the calculations involved in robust predictive control strategies. One possible way to speed up the algorithms is by approaching the problem in a distributed fashion. In general, the use of one central controller to control highly interconnected process units in chemical plants is generally computationally challenging and difficult to implement and hence, a more practical approach is to partition

the process into smaller subsystems and to design lower dimensional controllers for each subsystem (Scattolini, 2009). This distributed control approach referred to as Distributed Model Predictive Control (DMPC) has gained significant attention from the research community with various algorithms being proposed which can be broadly classified with type of, i) cost function (local vs. global), ii) solution procedure being used (non-iterative, single iteration) and iii) degree of exchange of information Scattolini, 2009. Venkat *et al.*, 2005 proposed a method of cooperative distributed control that permits to recover the performance of a centralised controller when convergence of the distributed algorithm occurs. Other DMPC algorithms that have been proposed are Zhang and Li, 2007, Liu *et al.*, 2009, Scheu and Marquardt, 2011.

Within the framework of DMPC, it is also important to consider robustness to plant-model mismatch. To provide for robustness Al-Gherwi *et al.*, 2011 assumed the plant model of each subsystem to be included within a polytopic model and the control action was based on the minimisation of a robust performance bound where this latter minimisation step is conducted iteratively for every subsystem in a cooperative manner. Robust DMPC using the “tubes” concept (Trodden and Richards, 2006) consists of developing invariant regions (tubes) at each time instant for linear time invariant models with interactions between subsystems treated as bounded disturbance, though plant-model mismatch has not been explicitly included as yet in that approach.

A key component of most previously proposed DMPC algorithms is the exchange of state information at the beginning of all iterations. This exchange is required at every time step, thus a situation where communication is lost because of dropped packets or poor signal needs to be explicitly addressed (Rawlings and Stewart, 2008). It is also a very common event in the chemical industry that sub-controllers controlling a particular section of a process are momentarily stopped for maintenance or to address a particular alarm. Using a nominal model based estimator, de la Pena and Christofides, 2008 designed for communication loss within the cost function of Lyapunov based MPC which guaranteed stability and included constraints for the length of the data loss period. Maestre *et al.*, 2009 designed DMPC where each agent developed and communicated various options for future control action and they

cooperated towards the central optimisation problem. However, during communication loss the system acted like a decentralised controller. Heidarinejad *et al.*, 2011 developed a scheme to ensure stability with communication loss by including a feasibility problem and assuming zero control action for other subsystems in case of communication loss. Sun and El-Farra, 2008 proposed a model-based control method based on a decentralised approach and low communication requirements. In this scheme subsystems would interact with each other at predetermined time instants, and for remaining time instants a nominal model is used as a state estimator, essentially acting as an open loop network. Distributed MPC in the presence of communication loss and model error is addressed in the current study by the use of a robust estimator of the states.

Chapter 3

Robust Nonlinear MPC based on Volterra series and Polynomial Chaos Expansions[1]

(Published in Journal of Process Control)

3.1 Introduction

Model Predictive Control (MPC) is a widely used control design technique in the process industries. MPC may be designed based on either a linear or nonlinear models where the latter design is referred to as Nonlinear Model Predictive Control (NMPC). Model based controllers must be robust to mismatch between the model predictions and the actual process behavior. While there exist a variety of techniques to analyze robustness of controllers that are based on linear models, design of robust nonlinear controllers such as NMPC is challenging and it is currently an active field of research (Allgower *et al.*, 2004, Magni and Scattolini, 2010).

Some robust-NMPC (RNMPC) algorithms have been reported and they can be classified into two main groups: i) - algorithms that are based on minmax formulation Diaz-Mendoza and Budman, 2010b, and ii) - algorithms for which nominal inputs are computed using an auxiliary controller and then input deviations from the nominal inputs are calculated to guarantee that actual outputs' trajectories are bounded within a tube around the nominal path (Mayne *et al.*, 2011, Magni *et al.*, 2006). Algorithms based on the "tube" approach have used mechanistic models thus making it difficult to generalize the analysis to general forms of nonlinearity. Consequently, the tube approach have only considered uncertain disturbances rather than parametric uncertainty. On the other hand, minmax formulations involving minimization of worst plant performance with respect to model uncertainties have considered both unmeasured disturbances and parametric uncertainty. For minmax formulation of robust optimization problems two approaches have been proposed : i. a simulation based approach with the goal of minimizing (Kawohl *et al.*, 2007) the weighted contribution of the first two

[1] Adapted from Kumar, D. & Budman, H. 2014.

statistical moments of an objective function using Monte Carlo simulations. This approach is computationally expensive due to the use of Monte Carlo and such studies have concentrated on robust optimization rather than robust control or ii- an analytical approach (Nagy and Braatz, 2003) based on bounds calculated by using Structured Singular Values (SSV or μ). In Diaz-Mendoza and Budman, 2010b presented an RNMPC strategy based on Structured Singular Value norms (SSV) for continuous processes, in which the cost function is formulated as an SSV norm calculation to produce worst closed-loop predictions in the presence of uncertainty which are used to satisfy input and terminal constraints. This algorithm was based on empirical Volterra models with uncertain coefficients. Empirical models were selected in that previous work because, as mentioned above, it is difficult to formulate a general robust approach for mechanistic models due to the various forms in which nonlinearities appear in these models. Also, since computation of the SSV-norm is a NP-hard type of problem, the computational time of the algorithm previously proposed by Mendoza and Budman increases exponentially as the dimensions of the process in terms of inputs and outputs increase. Moreover, since the controller based on SSV calculations minimizes worst case maximal deviations it results in conservative performance.

In the current work an alternative approach is proposed whereby parametric uncertainty is propagated onto the outputs using Polynomial Chaos Expansions (PCE). Few studies on the use of PCE for control design have been reported. For example in Hover and Triantafyllou, 2006 a PCE was used to perform stability analysis of a particular nonlinear system with random initial conditions or random parameters where the output was determined with similar accuracy to Monte Carlo simulations but with much reduced computational burden. In Smith *et al.*, 2009 an LQG controller was designed where parametric uncertainty and bounded disturbances were represented by PCE, but the study was limited to linear systems.

In the current study a minmax type online robust controller is proposed based on an empirical Volterra series' model, in which parametric uncertainty is represented by PCE's and it is propagated onto the variance. The key idea for using PCE's is that the variance of the predicted outputs can be rapidly calculated by an analytical expression thus critically reducing computational times as compared to the approach previously used by Diaz-

Mendoza and Budman, 2010b. In the current approach an SSV calculation has to be still performed to enforce a terminal condition and input constraints; but it does not have to be applied to the entire prediction horizon, as in the previous work, thus significantly reducing the computational load as compared to the previous study of Diaz-Mendoza and Budman, 2010b. Furthermore, since in the current approach a variance of the predicted outputs is minimized instead of a worst error considered in the previous work this approach will be shown to be less conservative. To illustrate the approach a 2x2 pH neutralization system is simulated and the closed loop performance of the algorithm is then compared with the performance of three NMPC controllers previously proposed i) non-robust NMPC based on first principles' model ii) non-robust NMPC based on nominal Volterra series model iii) robust NMPC controller also based on a nominal Volterra model for which the worst error along the prediction horizon is calculated via a Structured Singular Value (SSV or μ) test. These three controllers will be referred heretofore as FP-NMPC, non-robust NMPC and μ-based RNMPC respectively. In contrast to these three controllers, the robust algorithm presented in the current study will be referred to as PCE-based RNMPC since the nominal model used for prediction is still a Volterra series but robustness to model uncertainty is addressed via PCE expansions. This chapter is organized as follows; Section 3.2 discusses the prediction model development based on Volterra series, relevant background on PCE expansions and how to propagate model uncertainty onto the outputs' predictions. The determination of the variance in model predictions in the presence of model uncertainty using PCE and the μ calculations related to the formulation of the terminal condition and input constraints, along with the cost function to be minimized at every time step is presented in Section 3.3. Finally in section 3.4 the Robust NMPC is applied to the pH-neutralisation case study and is compared with the three controllers mentioned above.

3.2 Definitions and Methodology

3.2.1 Closed-loop Prediction Model using Volterra series

An input-output empirical model based on an auto-regressive Volterra (ARX Volterra) series, known for its ability to describe nonlinear behavior Parker *et al.*, 2001, is used to predict the

outputs of the process to be controlled. The possibility to separate these series into nominal and uncertain parts and the linearity of the output with respect to the parameters' uncertainty bounds facilitates the calculation of robust norms. The use of an autoregressive term, results in a model with a lower number of parameters as compared to Volterra series without such term thus reducing model sensitivity to noise in the data. A feedback correction term is included in the closed-loop prediction model to account for this mismatch. As commonly done for both linear and nonlinear predictive control algorithms, the feedback correction in the current study is calculated as the difference of the measured plant output (y_{k-1}^{real}) and a nominal output prediction ($\hat{y}(k-1)$). The model accounting for the feedback correction and for multiple input variables is given as follows:

$$\begin{aligned}\hat{y}_{\chi}(k+1) = & \sum_{q=1}^{n_{ARX}} h_{q_{\chi}} \hat{y}_{\chi}(k+1-q) + \sum_{n=0}^{M-1} h_{n(\chi,1)} u_1(k+1-n) \\ & + \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} h_{i,j(\chi,1)} u_1(k+1-i) u_1(k+1-j) + \cdots \\ & + \sum_{n=0}^{M-1} h_{n(\chi,n_u)} u_{n_u}(k+1-n) \\ & + \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} h_{i,j(\chi,n_u)} u_{n_u}(k+1-i) u_{n_u}(k+1-j) + w_k\end{aligned} \qquad 3.1$$

where, $\hat{y}_{\chi}(k+1)$ is the output prediction at time instant $k+1$, $n_{ARX}$is the number of auto-regressive terms, n_{χ}, n_u are the number of outputs and inputs in the system respectively, $u_{1,2,\ldots,n_u}$ are the inputs to the system, M is the Volterra series memory, h_i's are Volterra series coefficients and w_k is the feedback term. To account for parametric uncertainty, Eq. 3.1 can be written by expressing the coefficients, h_i's, with their corresponding nominal and uncertainty bounds as follows:

$$\hat{y}_\chi(k+1) = \sum_{q=1}^{n_{ARX}} (h_{q_\chi} \pm \delta h_{q_\chi}) y_\chi(k+1-q) + \sum_{n=0}^{M-1} \left(h_{n(\chi,1)} \pm \delta h_{n(\chi,1)}\right) u_1(k+1-n) + \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} \left(h_{i,j(\chi,1)} \pm \delta h_{i,j(\chi,1)}\right) u_1(k+1-i) u_1(k+1-j) + \cdots + \sum_{n=0}^{M-1} \left(h_{n(\chi,n_u)} \pm \delta h_{n(\chi,n_u)}\right) u_{n_u}(k+1-n) + \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} \left(h_{i,j(\chi,n_u)} \pm \delta h_{i,j(\chi,n_u)}\right) u_{n_u}(k+1-i) u_{n_u}(k+1-j) + w_k \quad 3.2$$

To identify the Volterra series coefficients, the procedure presented in Diaz-Mendoza and Budman, 2010a was used whereby a persistently exciting input signal and a nonlinear optimization is used to solve for the model's coefficients. For a N^{th} order Volterra series, a N+1 level Pseudo-Random Multilevel Signal (PRMS) has been shown to provide sufficient excitation Nowak and Vanveen, 1994 for identifying the Volterra series parameters. To determine both the nominal and uncertain parts of Volterra series parameters, PRMS sequences are applied in the inputs around different operating points corresponding to different values of actual process parameters such as flow rates, inlet compositions etc. Then, the series' coefficients identified for each of these PRMS inputs are averaged to determine nominal values of the parameters of Volterra series and their variances are used as uncertainty bounds. These uncertainty bounds, associated to each of the individual parameters of the Volterra series, can then be used to compute variances for the outputs' predictions along the prediction horizon as explained in the following section.

3.2.2 Prediction of $\mathcal{L}^2$-norm in presence of model uncertainty using PCE

In the current work each Volterra series' parameter is considered uncertain and it is assumed

to be described using a PCE as follows:

$$h_{ij} = \sum_{l=0}^{P_{PCE}-1} h_{ij,l}\,\Phi_l \qquad 3.3$$

where h_{ij} is the uncertain parameter and $h_{ij,l}$ is the deterministic coefficient multiplying the corresponding basis function Φ_l. Although in reality the uncertainty coefficients are not purely random and the assumption of randomness may result in conservatism, this assumption is used since it greatly facilitates the quantification of output variance. Accordingly, the nominal part of the uncertain coefficient is given by the first term in the expansion, $h_{ij,0}$, while the variance is equated to the sum of squares of remaining PCE coefficients i.e. $\sum_{l=1}^{P_{PCE}-1}(h_{ij,l})^2$. In the current study, the basis functions Φ_l, are assumed to be dependent on only one uncertain variable ξ, thus each Volterra series parameter is represented for simplicity as a one-dimensional PCE. Although there can be multiple sources of uncertainty, ($\xi_i : i \in 1,2, \ldots \infty$) such as changes in operating conditions, hardware related uncertainties, one single random variable was selected here for simplicity. Another key assumption is related to the distribution of the random variable ξ, as the type of basis functions used in (3.3) depends on this distribution. For simplicity a Gaussian distribution for ξ was assumed and following the Askey scheme mentioned above, Hermite polynomials were chosen as the basis functions. In general, a formal identification of the distribution of the random variables could be obtained by off-line Monte Carlo based identification of the parameters and uncertainty bounds and then different basis functions could be chosen to match the obtained distributions however this is beyond the scope of the current study which focuses on the control strategy.

To illustrate on how the uncertainty in the model parameters is propagated into the output variance, the method is illustrated for the first prediction interval and then is generalized to all intervals along the prediction horizon. The calculation of the first plant output prediction along the prediction horizon, $\hat{y}_\chi(k+1)$ with the uncertain model described using Eq. 3.3 and 3.4 is given as follows:

$$\hat{y}_\chi(k+1) = \sum_{q=1}^{n_{ARX}} \left(\sum_{l=0}^{P_{PCE}-1} h_{q_\chi,l}\Phi_l \right) \hat{y}_\chi(k+1-q) + \sum_{n=0}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{n(\chi,1),l}\Phi_l \right) u_1(k+1-n)$$
$$+ \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{i,j(\chi,1),l}\Phi_l \right) u_1(k+1-i)u_1(k+1-j) + \cdots$$
$$+ \sum_{n=0}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{n(\chi,n_u),l}\Phi_l \right) u_{n_u}(k+1-n) \qquad 3.4$$
$$+ \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{i,j(\chi,n_u),l}\Phi_l \right) u_{n_u}(k+1-i)u_{n_u}(k+1-j) + w_k$$

To propagate the uncertainty in parameters to the output, the plant output, $\hat{y}_\chi(k+1)$ is also represented by a PCE based on the same basis functions Φ_l chosen to describe the uncertain parameters of the Volterra model. Then, the uncertain output represented by a PCE is equated to the uncertain Volterra model given in (3.5) resulting in the following equation:

$$\sum_{l=0}^{P_{PCE}-1} \hat{y}_{\chi,l}(k+1)\Phi_l$$
$$= \sum_{q=1}^{n_{ARX}} \left(\sum_{l=0}^{P_{PCE}-1} h_{q_\chi,l}\Phi_l \right) \hat{y}_\chi(k+1-q)$$
$$+ \sum_{n=0}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{n(\chi,1),l}\Phi_l \right) u_1(k+1-n)$$
$$+ \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{i,j(\chi,1),l}\Phi_l \right) u_1(k+1-i)u_1(k+1-j) + \cdots \qquad 3.5$$
$$+ \sum_{n=0}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{n(\chi,n_u),l}\Phi_l \right) u_{n_u}(k+1-n)$$
$$+ \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} \left(\sum_{l=0}^{P_{PCE}-1} h_{i,j(\chi,n_u),l}\Phi_l \right) u_{n_u}(k+1-i)u_{n_u}(k+1-j) + w_k$$

In Eq. 3.5, the LHS represents the PCE used to model the first output prediction along the prediction horizon. Similarly the PCE model can be developed for further predictions along

the prediction horizon. Each PCE coefficient in the LHS expansion for the output is denoted as $\hat{y}^{\chi}_{k+p,l}$, where p is the prediction horizon and k is the current time interval. To calculate these PCE coefficients, a Galerkin projection is used as follows. For simplicity of notation, let the Volterra series model in Eq. 3.5 RHS be represented by $f\left(\hat{y}^{\chi}_{k}, \dots, \hat{y}^{\chi}_{k+1-n_{ARX}}, u_{k+1}, u_k, \dots u_{k+2-M}, \sum_{l=0}^{P_{PCE}-1} h_{ij,l}\Phi_l\right)$. Then to determine each PCE coefficient for the model output associated with basis function Φ_l, a Galerkin projection of the Eq. 3.5 is calculated with respect to each basis function (also shown in Eq. 2.16)

$$\left\langle \left(\sum_{l=0}^{P_{PCE}-1} y_{k+1,l}\Phi_{\mathrm{l}} \right), \Phi_l \right\rangle = \left\langle f\left(\hat{y}^{\chi}_{k}, \dots, \hat{y}^{\chi}_{k+1-n_{ARX}}, u_{k+1}, u_k, \dots u_{k+2-M}, \sum_{l=0}^{P_{PCE}-1} h_{ij,l}\Phi_l\, , w_k \right), \Phi_l \right\rangle \tag{3.6}$$

where $\langle x, y\rangle$ denotes inner product. Then, for the first prediction along the horizon, due to orthogonality of the basis functions, the PCE coefficient for the output is obtained as follows:

$$y_{k+1,l} = \left\langle f\left(\hat{y}^{\chi}_{k}, \dots, \hat{y}^{\chi}_{k+1-n_{ARX}}, u_{k+1}, u_k, \dots u_{k+2-M}, \sum_{l=0}^{P_{PCE}-1} h_{ij,l}\Phi_l\, , w_k\right), \Phi_l \right\rangle \Big/ \langle \Phi_l, \Phi_l \rangle \tag{3.7}$$

$$\forall l \in [0, P_{PCE}-1]$$

where in Eq. 3.7 the output predictions at previous sampling intervals, $\hat{y}^{\chi}_{k}, \dots, \hat{y}^{\chi}_{k+1-n_{ARX}}$ are also uncertain and are each represented by a corresponding PCE, thus resulting in the following general expression:

$$\hat{y}_{k+1,l} = \frac{\left\langle f\left(\begin{matrix} \sum_{l=0}^{P_{PCE}-1}\left(\hat{y}^{\chi}_{k,l}\Phi_l\right), \dots, \sum_{l=0}^{P_{PCE}-1}\left(\hat{y}^{\chi}_{k+1-n_{ARX},l}\Phi_l\right), u_{k+1}, u_k, \dots, u_{k+2-M}, \\ \sum_{l=0}^{P_{PCE}-1}\left(h_{ij,l}\Phi_l\right) \end{matrix} \right), \Phi_l \right\rangle}{\langle \Phi_l, \Phi_l \rangle} \tag{3.8}$$

$$\forall l \in [0, P_{PCE}-1]$$

For subsequent predictions, expressions similar to Eq. 3.8 can be developed but are not shown for brevity. The calculations of the inner products between basis functions in Eq. 3.8 require evaluation of n_0-dimensional integrals which can be done using Gaussian

quadratures. Although Gaussian quadratures are computationally expensive with increasing number of dimensions of the PCE, in the current work these integrals need to be computed only once and thus they can be conducted offline. Then, analytical expressions for every PCE coefficient of the model output for all output predictions along the prediction horizon, can be determined as a function of the control inputs, $u_{k+1}, \dots, u_{k+1}, u_k, \dots, u_{k+2-M}$, the PCE coefficients representing each of the Volterra series parameters $h_{ij,l}$, and the feedback error, $w_k$as per Eq. 3.8.

3.3 Robust controller formulation and cost function

Using a Volterra series model, Diaz-Mendoza and Budman, 2010b developed a test to find the model ,within a family of models representing the actual process, for which a worst closed loop performance is obtained. The rationale for this test is that if the worst performance is bounded by a performance index then all other possible models within the uncertainty set of models will satisfy it as well. The key disadvantage of this formulation was its conservatism, as the possibility of the worst model behavior occurring in the actual process might be low as compared to the nominal model. To remove some of this conservatism, the controller in the current work is designed based on a variance calculated for all possible models within the uncertainty set rather than on a worst error.

In section 3.2 it was shown how parametric uncertainty can be propagated onto the output using PCE expansions. Since the basis functions for the spectral expansion are orthogonal, the $\mathcal{L}^2$-norm of each prediction represented using PCE ($y_{k+i} = \sum_{l=0}^{P_{PCE}-1} y_{k+i,l}\Phi_l$, $\forall i \in [1, p]$), is the sum of squares of its PCE coefficients as follows:

$$\left\|\hat{y}^{\chi}_{k+i,l}\right\|_{\mathcal{L}^2} = \sum_{l=0}^{P_{PCE}-1} \left(\hat{y}^{\chi}_{k+i,l}\right)^2, \forall i \in [1,p] \tag{3.9}$$

$$\sum_{i=1}^{p}\left\|\hat{y}^{\chi}_{k+i,l}\right\|_{\mathcal{L}^2} = \sum_{i=1}^{p}\sum_{l=0}^{P_{PCE}-1} \left(\hat{y}^{\chi}_{k+i,l}\right)^2 \tag{3.10}$$

and correspondingly an analytical expression can be obtained for $\hat{y}^{\chi}_{k+i,l}$ as shown in Section 3.2. Based on this analytical calculation of variance it is possible to design a control law based on the on line solution of the following optimization problem:

$$\min_{U} J \quad J = \left\| \begin{matrix} \sum_{i=1}^{p} \left\| \hat{y}^{\chi}_{k+i,l} \right\|_{\mathcal{L}^2} \\ W\Delta U \\ k_{SSV} \end{matrix} \right\|_{\infty} \qquad 3.11$$

$$\max_{wrt\ H^L, H^{NL}} \left\| \begin{matrix} k_{SSV} * u/u_{limits} \\ tc \end{matrix} \right\|_{\infty} \Leftrightarrow \max_{\substack{wrt\ k_{SSV} \\ st\ \mu_{\Delta}(M) \geq k_{SSV}}} (k_{SSV})$$

According to problem 3.11, the first two terms in the cost function J to be minimized at every time step includes the $\mathcal{L}^2$-norm of the outputs over the prediction horizon and weighted input changes, $W\Delta U$. $\mathcal{L}^2$-norm of the outputs over the prediction horizon can be calculated by an analytical expression given by Eq. 3.10. Weighted input changes, $W\Delta U$, is computed as it is generally done in other predictive control formulation, as a product of weighting factor and difference in successive control actions (3.12)

$$W\,\Delta\,U = \begin{bmatrix} W_1^{\Delta u_1}[u_1(k) - u_1(k-1)] \\ \vdots \\ W_{\mathrm{m}}^{\Delta u_1}[u_1(k+m) - u_1(k+m-1)] \\ \vdots \\ W_1^{\Delta u_{n_u}}\left[u_{n_u}(k) - u_{n_u}(k-1)\right] \\ \vdots \\ W_{\mathrm{m}}^{\Delta u_{n_u}}\left[u_{n_u}(k+m) - u_{n_u}(k+m-1)\right] \end{bmatrix} \qquad 3.12$$

The third term in the cost function 3.11 is a calculated bound denoted as k_{SSV}, which is used to impose constraints on manipulated variables and to enforce a terminal condition represented by tc which corresponds to the last deviation between the output and the set-point along the prediction horizon as per the Eq. 3.13. This bound (k_{SSV}), is calculated by a Skewed Structure Singular Value calculation (skewed μ, Braatz *et al.*, 1994) for which details are given in the Appendix A, The last equation in problem (3.11) ensures that in the limit either u/u_{limits} or tc can be at most equal to k_{SSV}. By including this bound within the cost function J, the aim is to minimize it along with the variance and the weighted input changes.

$$tc = \begin{bmatrix} tc_1 \\ \vdots \\ tc_\chi \end{bmatrix}$$

$$tc_\chi = \hat{y}_\chi(k+p)\frac{k_{SSV}}{\epsilon}$$

$$\begin{aligned} &= \frac{k_{SSV}}{\epsilon}\Bigg[\sum_{q=1}^{n_{ARX}}(h_{q_\chi} \pm \delta h_{q_\chi})\hat{y}_\chi(k+p-q) + \sum_{n=0}^{M-1}\big(h_{n(\chi,1)} \pm \delta h_{n(\chi,1)}\big)u_1(k+p-n) \\ &+ \sum_{i=0}^{M-1}\sum_{j=i}^{M-1}\big(h_{i,j(\chi,1)} \pm \delta h_{i,j(\chi,1)}\big)u_1(k+p-i)u_1(k+p-j) + \cdots \\ &+ \sum_{n=0}^{M-1}\big(h_{n(\chi,n_u)} \pm \delta h_{n(\chi,n_u)}\big)u_{n_u}(k+p-n) \\ &+ \sum_{i=0}^{M-1}\sum_{j=i}^{M-1}\big(h_{i,j(\chi,n_u)} \pm \delta h_{i,j(\chi,n_u)}\big)u_{n_u}(k+p-i)u_{n_u}(k+p-j) \; + \; w_k\Bigg] \end{aligned} \qquad 3.13$$

However, to calculate (3.11), bounds for the autoregressive term $\hat{y}^{\chi}_{k+p-1}$ needs to be determined. These bounds are determined using the corresponding PCE for $\hat{y}^{\chi}_{k+p-1}$. The key difference between the current approach and the previous work of Diaz-Mendoza and Budman, 2010b is, in that work worst bounds k_{SSV}'s were computed for each one of the output predictions along the horizon (Refer to Table 3.1 which summarizes different controllers compared in the current work), whereas in the present work the bound was computed only for the terminal condition. The performance cost related to the intermediate outputs along the horizon was quantified in the current study with an analytical expression for variance based on the PCE approach thus dramatically reducing the computational cost as compared to the algorithm used by Diaz-Mendoza and Budman, 2010b.

Offset Removal: As with other minmax approaches involving the minimization of worst case scenarios the current controller does not eliminate offset. To illustrate the occurrence of offset consider the scenario, when steady state is reached and input constraints are not active, i.e. $y^{\chi}_{k+1} = y^{\chi}_{k}$ and $u_1(k+1) = u_1(k), u_2(k+1) = u_2(k) \ldots u_{n_u}(k+1) = u_{n_u}(k)$. In that case the cost function reduces to the following equation,

$$J = \left|\hat{y}(k+1)_\delta - y^{sp}(k) + y^{plant}(k) - \hat{y}(k)\right|^2 \qquad 3.14$$

where $\hat{y}(k+1)_{\delta}$, represents the prediction of robust model (based on Eq.3.4) at time instant $k+1$ when uncertainty is considered, and $\hat{y}(k)$ is the nominal model prediction (based on Eq.3.1) at time instant k. Since these two quantities, i.e., $\hat{y}(k+1)_{\delta}$ and $\hat{y}(k)$, are never equal hence $y^{plant}(k)$ is never equal to $y^{sp}(k)$, consequently an offset is observed even when J in 3.11 is driven to zero. To remove this offset the concept of dual controller proposed by Chen and Allgower, 1998 is used whereby whenever the system approaches steady state a controller based on a nominal model is used, i.e. at steady state $\hat{y}(k+1)_{\delta} = \hat{y}(k)$, hence if the cost J is driven to zero $y^{plant}(k) = y^{sp}(k)$. To assess the closeness to steady state, an ad-hoc test is applied whereby a difference in plant output for three successive time steps are compared to a predefined small number; e.g.at time k, if $\left|y_k^{real} - y_{k-1}^{real}\right|, \left|y_{k-1}^{real} - y_{k-2}^{real}\right|, \left|y_{k-2}^{real} - y_{k-3}^{real}\right| \leq \delta, \delta = 1e-3$, then a nominal controller is used.

3.4 Case Study

The PCE-based RNMPC (RNMPC) controller was applied to a pH-neutralisation process and then compared to a robust controller based on a Structured Singular Value test (μ-based RNMPC), to a non robust controller based on a nominal Volterra model (non-robust NMPC) and to a non-robust controller based on a nominal first principle model (FP-NMPC) corresponding to the differential equations model given in Table 3.2.

The NMPC schemes used in the comparisons are summarized in Table 3.1 in terms of cost, model used for output prediction and whether an observer is used for estimation of unmeasured states. The last row in Table 3.1 is related to how the uncertainty is considered

Table 3.1 Other NMPC schemes

	FP-NMPC	non-Robust NMPC	μ-based RNMPC	PCE-based RNMPC
Model Predictions	$x(k+1) = f(x_k, u_k)$ $y_k = g(x_k, u_k)$	Nominal volterra series model (Eq.3.1)	Uncertain Volterra series prediction (Eq.3.2)	Uncertain Volterra series prediction with PCE (Eq.3.5)
Observer Equations	$\hat{x}(\hat{k}+1) = f(\hat{x}_k, u_k)$ $\hat{y}_k = g(\hat{x}_k, u_k) + w_k$ $w_k = y_{k-1}^{plant} - \hat{y}_{k-1}$	None	None	None
Cost Function	$\min\limits_{wrt\ U} \left(\left\| \begin{matrix} \hat{Y} - Y^{sp} \\ W\Delta U \\ k_{SSV} \end{matrix} \right\|_\infty \right)$ $\max\limits_{wrt\ H^L, H^{NL}} \left\| \begin{matrix} k * u/u_{limits} \\ tc \end{matrix} \right\|_\infty$ $= \max\limits_{\substack{wrt\ k_{SSV} \\ st\ \mu_\Delta(M) \geq k_{SSV}}} (k_{SSV})$	$\min\limits_{wrt\ U} \left(\left\| \begin{matrix} Y^{nominal} - Y^{sp} \\ W\Delta U \\ k_{SSV} \end{matrix} \right\|_\infty \right)$ $\max\limits_{wrt\ H^L, H^{NL}} \left\| \begin{matrix} k * u/u_{limits} \\ tc \end{matrix} \right\|_\infty$ $= \max\limits_{\substack{wrt\ k_{SSV} \\ st\ \mu_\Delta(M) \geq k_{SSV}}} (k_{SSV})$	$\min\limits_{wrt\ U} \left(\max\limits_{wrt\ H^L, H^{NL}} \left\| \begin{matrix} \hat{Y}_{k+i}^{unc} - Y^{sp} \\ W\Delta U \\ k * u/u_{limits} \\ tc \end{matrix} \right\|_\infty \right)$ $= \min\limits_{wrt\ U} \left(\max\limits_{\substack{wrt\ k_{SSV} \\ st\ \mu_\Delta(M) \geq k_{SSV}}} (k_{SSV}) \right)$	$\min\limits_{wrt\ U} \left(\left\| \begin{matrix} \sum_{i=1}^{p} \left\| \hat{y}_{k+i,l}^{\chi} \right\|_{L_2} \\ W\Delta U \\ k_{SSV} \end{matrix} \right\|_\infty \right)$ $\max\limits_{wrt\ H^L, H^{NL}} \left\| \begin{matrix} k * u/u_{limits} \\ tc \end{matrix} \right\|_\infty$ $= \max\limits_{\substack{wrt\ k_{SSV} \\ st\ \mu_\Delta(M) \geq k_{SSV}}} (k_{SSV})$
Terminal Cost	Uncertain Volterra Series Prediction (Eq. 3.13)	Uncertain Volterra Series Prediction (Eq. 3.13)	Uncertain Volterra Series Prediction (Eq. 3.13)	Uncertain Volterra Series Prediction (Eq. 3.13)
Uncertainty in $\hat{y}_{k+p-1}^{\chi}$ (autoregressive term for calculation of terminal cost in Eq. 3.13)	• Nominal Prediction using First Principles. • Feedback is based on First Principles	• Nominal Penultimate prediction from Volterra Series Model (Eq. 3.1)	• Uncertain Penultimate Prediction using Volterra Series (Eq. 3.2) • It is part of μ-calculation.	• Bounds on Penultimate Prediction, determined using PCE for output (Eq. 3.5)

(or ignored) for the autoregressive term in equation (3.11) used for the calculation of the terminal condition.

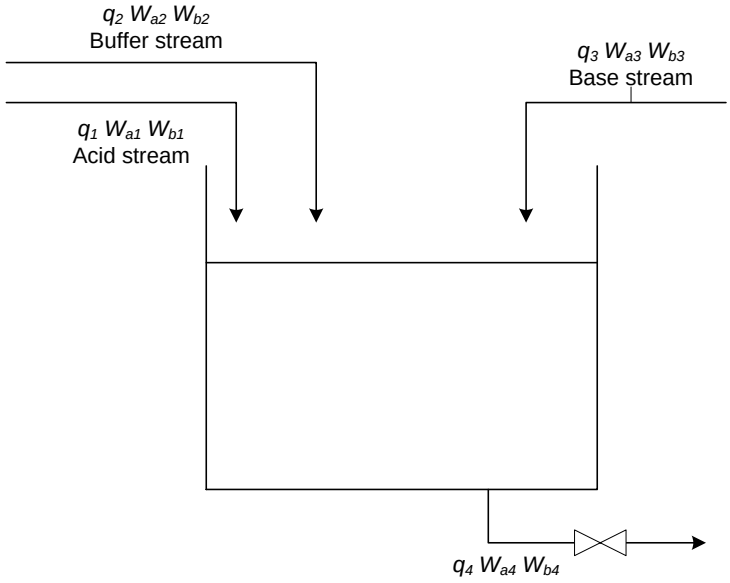


Figure 3.1 pH neutralisation system

The pH-neutralization system is shown in Figure 3.1, in which an acid, base and buffer streams enter the tank and are mixed uniformly. The control problem consists of maintaining the height of liquid, y_1, in the tank and pH of the effluent, y_2, at their set-point, by manipulating the flowrates of acid and base flows, u_1 and u_2, respectively. Changes in the buffer flowrate, q_2, are assumed to be the sole source of disturbances and changes to valve characteristics, C_v and n, during operation were considered to be the sources of unmodeled dynamics. Thus, nominal values in C_v and n were chosen to define the nominal model and deviations from these nominal values were considered in the numerical simulations to test for robustness. The ODEs governing the process are shown in Table 3.2. The operating conditions for the process are listed in Table 3.3. The Volterra series model and the controller schemes were designed using variables normalized with respect to the values presented in Table 3.4.

Table 3.2 Process dynamics for pH neutralisation system

$A\frac{dh}{dt} = q_1 + q_2 + q_3 - q_4$	$q_4 = C_v(h)^n$
$Ah\frac{dW_{a4}}{dt} = q_1(W_{a1} - W_{a4}) + q_2(W_{a2} - W_{a4}) + q_3(W_{a3} - W_{a4})$	$W_{ai} = [H^+]_i - [OH^-]_i - [HCO_3^-]_i - 2[CO_3^{2-}]_i$
$Ah\frac{dW_{b4}}{dt} = q_1(W_{b1} - W_{b4}) + q_2(W_{b2} - W_{b4}) + q_3(W_{b3} - W_{b4})$	$W_{bi} = [H_2CO_3]_i + [HCO_3^-]_i + [CO_3^{2-}]_i$
$W_a + 10^{pH-14} + W_b\frac{1 + 2 * 10^{pH-pK_2}}{1 + 10^{pK_1-pH} + 10^{pH-pK_2}} - 10^{-ph} = 0$	

The identification of nominal and uncertain Volterra series parameters is performed following the approach in Diaz-Mendoza and Budman, 2010a which is briefly reviewed here. Since the goal of this work is to assess robustness of the NMPC algorithms where the model error is related to changes in valve characteristics, different combinations of C_v and n values were considered in the simulations of the closed loop system. Correspondingly, a total of $n_{op} = 9$ operating regions were considered corresponding to all combinations of 3 different levels of values of C_v and n. For each such combination PRMS with $N + 1$ levels was applied to the system r_{seq}times, where $r_{seq}$is the number of times PRMS is applied on each operating region.

Table 3.3 Operating Conditions

A	207cm^2	W_{a1}	3×10^{-3}
pK_1	6.35	W_{a2}	-3×10^{-2}
pK_2	10.25	W_{a3}	-3.05×10^{-3}
q_1	3×10^{-3} M HNO_3	W_{b2}	3×10^{-2}
q_2	3×10^{-2} M $NaHCO_3$	W_{b3}	5×10^{-5}
q_3	3×10^{-3} M NaOH + 5×10^{-5} M $NaHCO_3$		

Table 3.4 Values for Normalisation

Variable	**V_p**	**V_d**
y_1, height	14	5
y_2, pH	7	3
q_1, acid flow	q_1	$0.2q_1^{ss}$
q_3, base flow	q_3	$0.225q_3^{ss}$

In the current work a 2nd order Volterra series was used to model the system, hence 3 levels of PRMS were deemed sufficient for identification. For each PRMS, the corresponding Volterra series parameters were obtained using nonlinear optimization. Thus a total of

$n_{op}r_{seq}$ parameters were identified. The standard deviation identified for each parameter was assumed to be equal to the uncertainty and it was used to determine the corresponding PCE coefficients.

The identification of PCE expansion's coefficients used to represent each one of the Volterra series parameters was based on the following assumptions: (i) the number of terms in the PCE expansion used to describe each Volterra series parameter is assumed to be 2, hence only a first order PCE is developed, $q = 1$, (ii) ξ is assumed to be normally distributed and correspondingly Hermite Polynomials were chosen as basis functions (Xiu and Karniadakis, 2002). Following these assumptions each of the Volterra series parameters was represented by a PCE with only two terms with corresponding coefficients denoted as $h_{ij,0}$ and $h_{ij,1}$ respectively. Thus the first term $h_{ij,0}$ corresponds to the nominal part of the parameter and the second term $h_{ij,1}$ is associated with the variance according to $h_{ij,1} = \sqrt{var(h_{ij})}$.

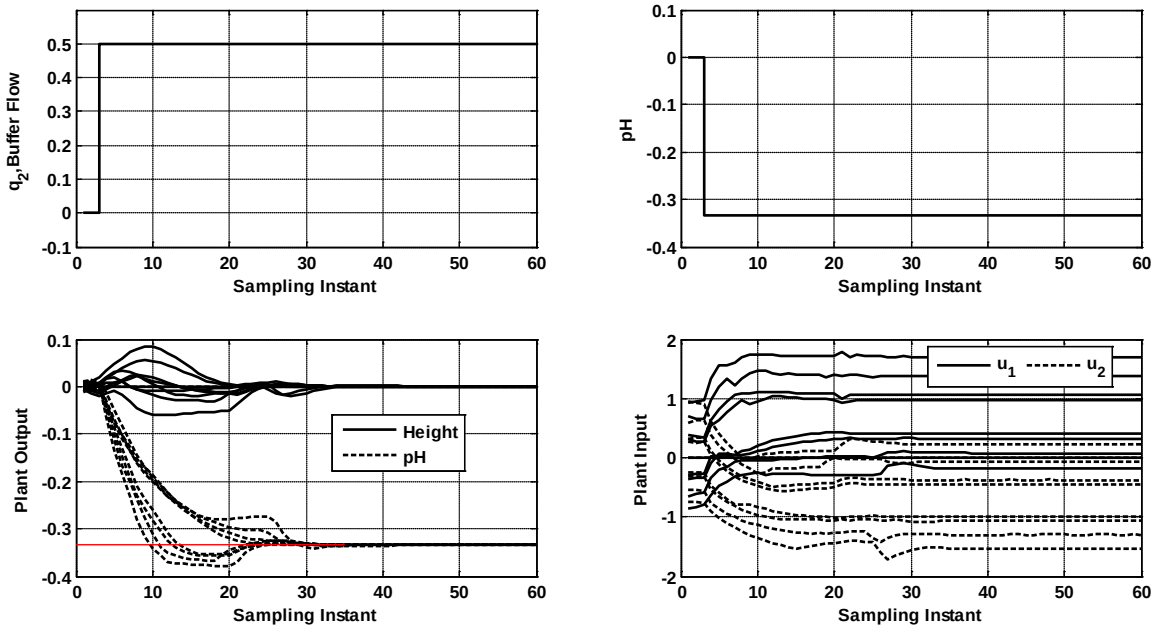


Figure 3.2 Setpoint tracking and Disturbance rejection at different operating conditions

As preliminary test of the proposed algorithm, its set-point tracking and disturbance rejection capabilities were tested around different operating conditions corresponding to different combinations of the valve parameters, C_v and n. The set-point change in pH were from 7 to 6

and the disturbance was a step change in the buffer flowrate from q_2 =0.55 to 0.825. Figure 3.2a and Figure 3.2b shows the disturbance and set point step-like changes respectively considered in the simulations. Controlled and manipulated variable response for different operating conditions i.e. different combinations of C_v and n are shown in Figure 3.2c and Figure 3.2d respectively. The bound on the terminal condition used is $\varepsilon = 0.4$ in terms of the normalized output variables defined in Table 3.4, i.e.$|y_{normalised}| < \varepsilon$. These simulations show that after the plant output approaches a steady state, the controller switches to a nominal model based controller, which is the non-Robust NMPC presented in Table 3.1, thus resulting in zero offset, irrespective of the operating condition as explained at the end of section 3.3.

Next the controller performance was tested in the presence of input constraints. The controller was simulated for $C_v = 8.25$ and $n = 0.5$, with a step-like disturbance for q_2 changing from 0.55 to 0.8, and $|u_2(k)| \leq 0.3$. Figure 3.3 shows the performance of controller with and without input constraints. These simulations verified that the formulation of the skewed-μ formulation given by (3.11) ensures compliance with input constraints.

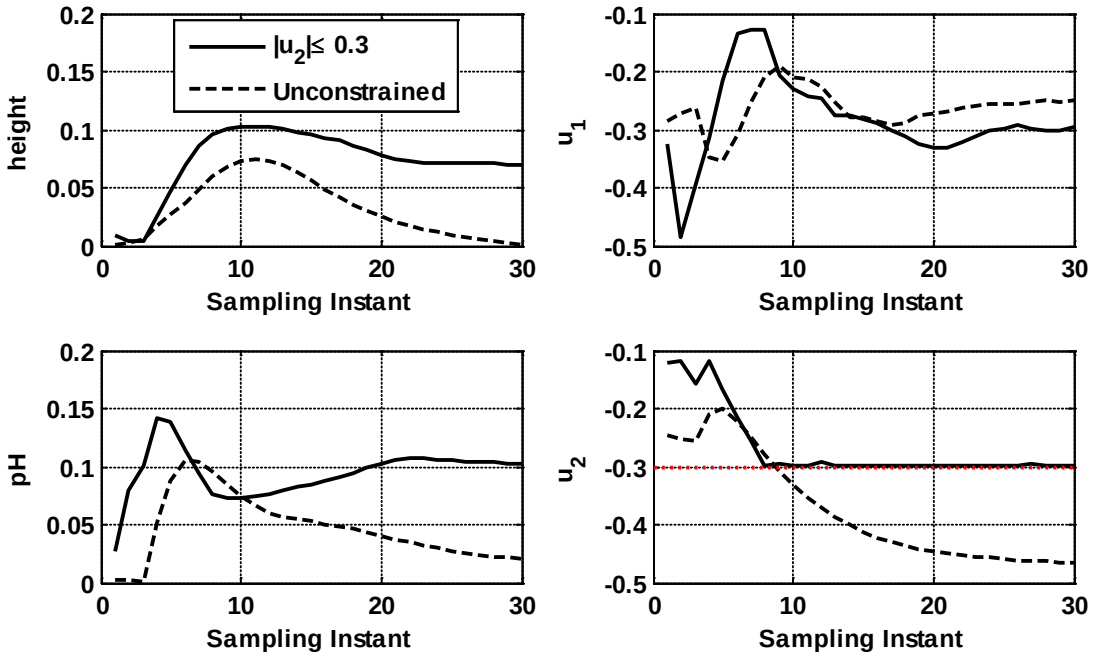


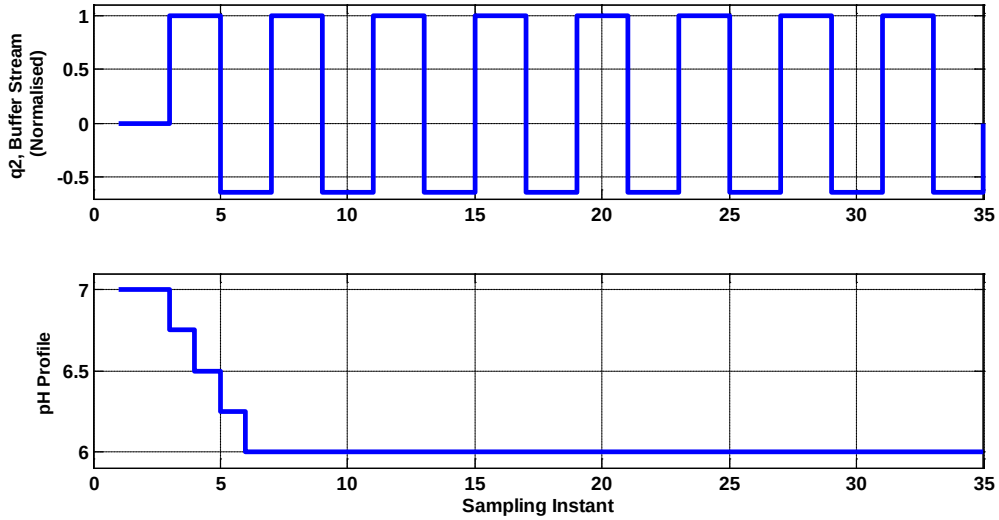
Figure 3.3 $[C_v, n] = 8.25, 0.5$, PCE-RNMPC with input constraints

As mentioned in the section 3.1 a key motivation of the current work was to improve over a previously proposed robust NMPC algorithm referred to as μ-based RNMPC (Diaz-Mendoza and Budman, 2010b) that was found to be prohibitive in terms of computational load and often conservative. Hence, the performance of the currently proposed technique PCE-RNMPC was compared to μ-based RNMPC in terms of computational effort and conservativeness. The two controllers were compared for the disturbance and pH set-point profiles in Figure 3.2a and Figure 3.2b for different operating conditions corresponding to the different combinations of C_v and n values. In terms of computational time, PCE-RNMPC required an average of 1hr to complete the entire calculation as compared to >5 days required for the μ-based RNMPC. Both the simulations were started from the same steady state condition at $ph = 7$ and $h = 14cm$, and the same initial guesses for optimization were used for both algorithms. The IAE for PCE-RNMPC was 2.29 and 2.7 for the μ-based RNMPC (average of completed runs). Thus beyond the dramatic reduction in computations, the PCE-RNMPC turns out to be less conservative.

Subsequently PCE was compared to 2 additional algorithms defined in the first two columns in Table 3.1, i.e. FP-NMPC based on a first principle (differential equations') model of the system and non-robust NMPC based on the nominal Volterra series model identified for the system under study. To conduct this comparison the 9 valve's parameters' values C_v and n combinations were varied to assess controller performances under varying operating conditions. The rationale for simulating the system for different combinations of the parameters' values was to assess the performance of robust and non-robust controllers in the presence of model error since the model used for control was based on nominal values of C_v and n whereas the plant was simulated for different values of these parameters. The parameter values used for the model simulating the actual plant are referred to as $C_{v,plant}$ and n_{plant} whereas the nominal values used in the nominal model are $C_{v,model} = 8.75$ and $n_{model} = 0.5$. Each NMPC algorithm accounted for these nominal values in a different manner. The FP-NMPC algorithm, being based on the differential equations describing the process, used these values explicitly. On the other hand, for PCE-RNMPC and non-Robust NMPC based on empirical Volterra series' models, $C_{v,model}$and n_{model} were accounted for

implicitly through the identification of the average values of the Volterra model coefficients from input-output data of the simulated plant. The Integral of Absolute Error (IAE) was used to measure the controller performance in the simulations for different combinations of input movement weighting factors' values. Six combinations of weighting factors were studied as follows: case 1 [0.20, 0.25], case 2 [0.20, 0.35], case 3 [0.20, 0.40], case 4 [0.25, 0.20], case 5 [0.35, 0.20] and case 6 [0.40, 0.20]. Figure 3.4, presents the disturbance (buffer flow, q_2) and pH set-point changes considered for these simulations. A terminal region was chosen as ε=0.8, i.e. $|y_{normalised}| < \varepsilon$ for all the controllers. Table 3.5 shows the IAE values for the aforementioned 6 cases.

Figure 3.4 Disturbance and Set-point profile used for testing robustness of different controllers



Out of the 6 cases that were studied, in 5 of these cases PCE-RNMPC performed better, in terms of the IAE averaged over all simulations, than FP-NMPC. Also, the average over the 6 cases $\overline{IAE}_{FP-NMPC}$ is higher than $\overline{IAE}_{PCE-RNMPC}$. This is attributed to the rapid disturbance changes in buffer flow due to which a significant plant-model mismatch occur and consequently the robust PCE-based RNMPC algorithm performs better than the non-robust FP-NMPC. As compared to the non-Robust NMPC, PCE-RNMPC always performed better, which means that considering uncertainty along the prediction horizon is essential even when

operating in the neighborhood of the nominal operating conditions. For example, Figure 3.5 compares the responses for PCE-RNMPC with FP-NMPC and non-Robust NMPC, for $w_1 = 0.25$ and $w_2 = 0.2$ around the nominal operating condition.

For all the case studies corresponding to different combinations of input weights it was observed that the average $\overline{IAE}_{PCE-RNMPC}$ was consistently lower than that of $\overline{IAE}_{FP-NMPC}$ and $\overline{IAE}_{non-Robust\ NMPC}$. At lower values of these weights, it was observed that the difference between non-Robust NMPC and PCE-RNMPC was very high (~20% and ~ 30% for case study 1 and 4 respectively) while, as expected, these differences decrease with increasing weights to ~10% for both case studies 3 and 6. This corroborates the fact that as the two controllers, i.e. non-Robust NMPC and PCE-RNMPC, become more aggressive for lower input weights values they turn increasingly more sensitive to plant-model mismatch.

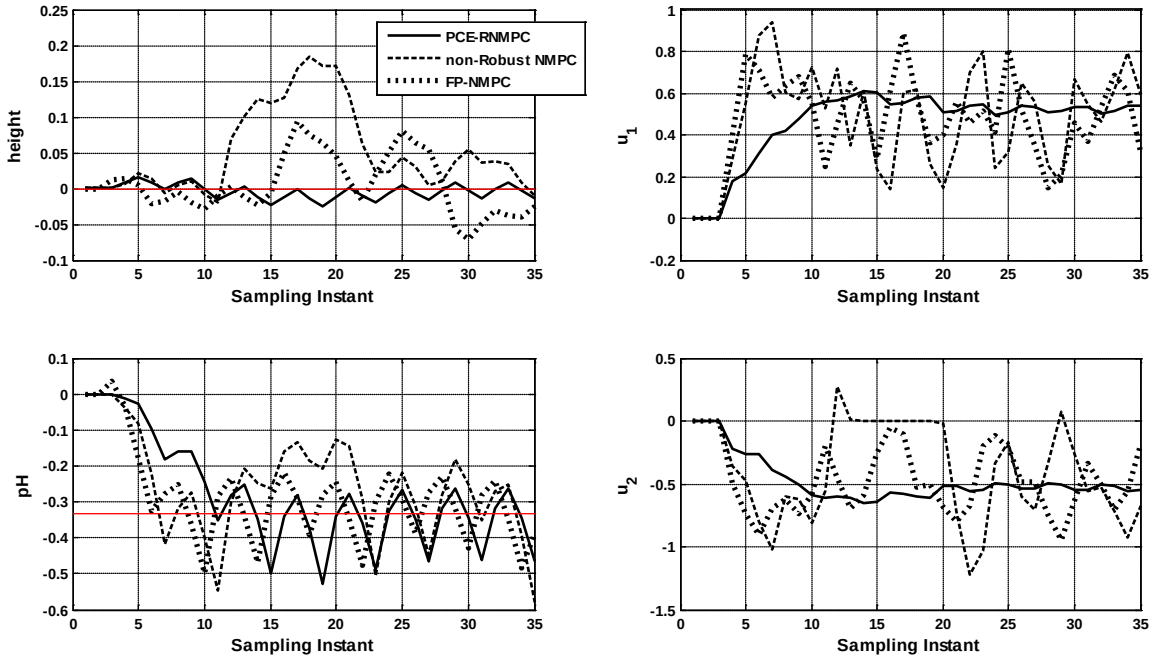


Figure 3.5 $[C_v, n] = 8.75, 0.5,\ [w_1, w_2] = [0.25, 0.2]$, Comparison of different controllers at nominal operating conditions

For PCE-RNMPC, the $\overline{IAE}_{PCE-RNMPC}$ was not very sensitive to the changes in the input weighting factors. On the other hand the effect of the input weights on the difference in performance between PCE-RNMPC and FP-NMPC was less consistent although PCE-

RNMPC performed consistently better. For instance, the difference between the two controllers appears to be the largest for intermediate values of the input weights (cases 2 and 5).

The PCE-RNMPC performed better than the non-Robust NMPC in 32/48 cases and in 39/48 cases when compared to FP-NMPC. The importance of accounting for robustness is best demonstrated in case study 3, with $w_1 = 0.2, w_2 = 0.4$ and for $C_{v,plant} = 9.25$ and $n_{plant} = 0.55$, where the difference in performance of PCE-based RNMPC and FP-NMPC was found to be ~80%. The corresponding responses for this latter case are shown in Figure 3.6.

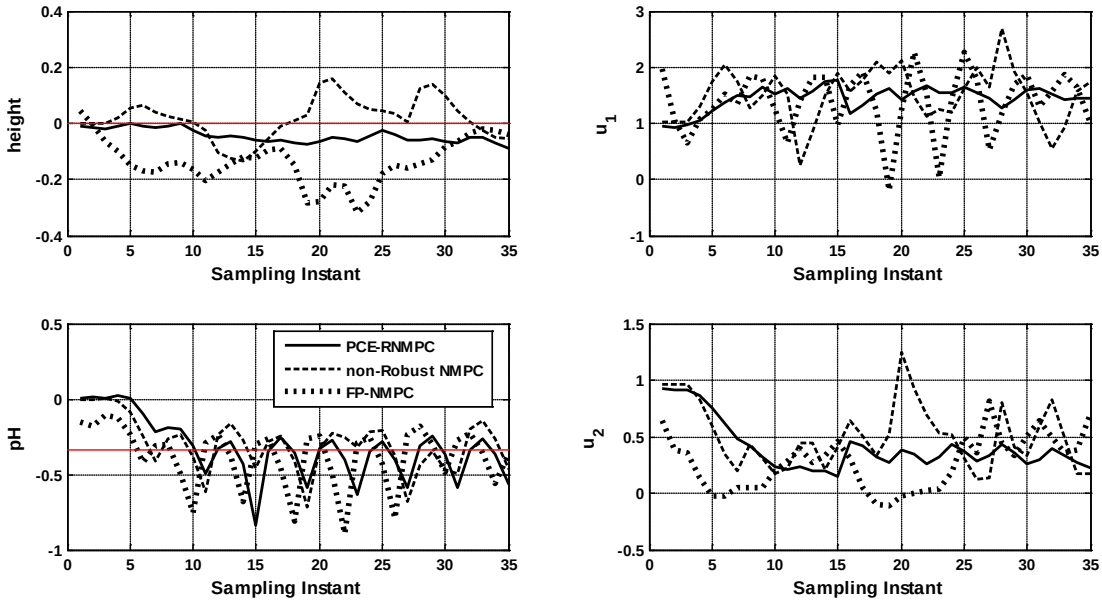


Figure 3.6 $[C_v, n] = [9.25, 0.55]$, $[w_1, w_2] = [0.2, 0.4]$, Comparison of different controllers

In summary, the average difference in performance of PCE-based RNMPC and FP-NMPC is ~20% and between PCE-RNMPC and non-Robust NMPC is ~10%. Also in comparison to the previously proposed μ-based RNMPC, the PCE-based RNMPC was shown to be both less conservative and less computationally onerous.

3.5 Conclusion

A new Robust-NMPC algorithm was developed based on Polynomial Chaos Expansions of the uncertain parameters of a Volterra series model. The key advantage of the algorithm is

that the variance along the prediction horizon can be quickly calculated using analytical formulae. This variance calculation is combined with an SSV-based norm calculation to enforce robust constraints for manipulated variable suppression and a terminal condition at the end of the control horizon. The average performance of the proposed controller is compared in simulations with other non-Robust controllers based on the IAE over a wide range of model parameter uncertainty and for different input weights values. The comparative study showed that $\overline{IAE}_{PCE-RNMPC}$ is consistently lower than $\overline{IAE}_{FP-NMPC}$ and $\overline{IAE}_{non-Robust}$. Also by comparing PCE-RNMPC to SSV-based RNMPC, where in the latter a worst error is calculated for the entire prediction horizon by an SSV-test, it was shown that PCE-RNMPC is both computationally less expensive and less conservative.

Table 3.5 Comparison of IAE for different controllers

Weighting Factor	w_1 = 0.2, w_2 =0.25			w_1 = 0.2, w_2 =0.35			w_1 = 0.2, w_2 =0.40		
Plant Valve Characteristics	non-Robust	PCE-RNMPC	FP-NMPC	non Robust	PCE-RNMPC	FP-NMPC	non Robust	PCE-RNMPC	FP-NMPC
Cv = 8.25, n =0.45	5.40	3.52	3.70	4.89	4.54	7.24	3.48	4.77	4.35
Cv = 8.25, n =0.5	3.66	3.26	3.58	3.39	3.31	3.76	3.00	3.30	3.99
Cv = 8.25, n =0.55	3.93	3.37	4.00	4.10	3.78	3.85	3.98	3.52	3.91
Cv = 8.75, n =0.45	3.58	4.73	5.10	3.45	4.42	4.20	3.75	3.70	4.14
Cv = 8.75, n =0.5	5.92	3.12	3.26	3.94	3.36	5.99	5.34	3.43	3.35
Cv = 8.75, n =0.55	6.29	5.15	5.78	5.01	4.44	5.79	4.73	4.43	4.88
Cv = 9.25, n =0.45	3.29	3.39	3.14	3.68	3.47	3.61	3.38	3.34	3.41
Cv = 9.25, n =0.5	5.12	3.39	4.70	4.06	3.11	3.55	4.63	3.90	4.72
Cv = 9.25, n =0.55	6.13	6.46	5.63	4.77	5.14	8.66	5.57	5.17	9.73
Average	4.81	4.04	4.32	4.14	3.95	5.19	4.21	3.95	4.72

Weighting Factor	w_1 = 0.25, w_2 =0.2			w_1 = 0.35, w_2 =0.2			w_1 = 0.40, w_2 =0.2		
Plant Valve Characteristics	non Robust	PCE-RNMPC	FP-NMPC	non Robust	PCE-RNMPC	FP NMPC	non Robust	PCE-RNMPC	FP-NMPC
Cv = 8.25, n =0.45	7.53	3.59	3.93	3.99	3.85	3.95	3.32	3.65	4.04
Cv = 8.25, n =0.5	4.95	2.90	3.21	3.58	3.23	3.34	5.15	3.31	3.90
Cv = 8.25, n =0.55	4.31	3.17	4.47	3.61	3.99	4.45	4.37	3.96	3.90
Cv = 8.75, n =0.45	3.38	4.23	3.76	3.44	3.66	4.23	3.08	3.52	4.14
Cv = 8.75, n =0.5	5.08	2.79	3.52	4.70	3.31	3.49	4.68	3.54	3.58
Cv = 8.75, n =0.55	7.35	4.62	4.59	3.86	4.20	5.58	4.91	5.14	4.76
Cv = 9.25, n =0.45	3.93	2.59	3.12	3.63	2.93	3.81	4.18	3.55	3.88
Cv = 9.25, n =0.5	4.31	3.36	4.56	6.47	3.50	3.99	3.98	3.41	5.19
Cv = 9.25, n =0.55	6.46	7.82	7.85	6.53	7.71	13.90	5.07	5.31	9.31
Average	5.25	3.90	4.33	4.42	4.04	5.19	4.30	3.93	4.74
Overall Average	4.52	3.97	4.75						

Chapter 4

Applications of Polynomial Chaos Expansions in optimization and control of bioreactors based on Dynamic Metabolic Flux Balance models[2]

4.1 Introduction

This work proposes model-based control and optimization approaches for bioreactor processes that are robust to model error. The key challenge in addressing robustness to model error is to propagate the uncertainty in model parameters onto the control or optimization objectives. When using nonlinear dynamic first principles' models such propagation requires the use of Monte Carlo algorithms which are computationally demanding. To reduce the computational load we propose the use of Polynomial Chaos Expansions that permit quick calculation of the variance resulting from the process model mismatch. Two different problems are tackled: i- on-line robust predictive control with an economic objective and ii- off-line robust optimization of an end point property.

Most of the reported studies on optimal operation of bioreactors involve offline model based optimization (Banga *et al.*, 1997), without accounting for feedback corrections, (Frahm *et al.*, 2002, Hjersted and Henson, 2006, Banga *et al.*, 1997) or for robustness to model errors. The objective of these optimization strategies have been generally the maximization of a property at the end of the batch such as the productivity.

Traditionally, studies of optimization of bioreactors have used unstructured models that are based on simplistic substrate and biomass balances thus not accounting for detailed interactions between different nutrients. On the other hand, structured models that explicitly account for detailed interactions between nutrients and products, have gained increasing acceptance in the pharmaceutical industry motivating their use for control and optimization. For example, Dynamic Flux Balance Modeling (DFBM) has been applied successfully by Mahadevan *et al.*, 2002, as an extension of MFA to describe the dynamic growth of *E.coli* on

[2] Part of this work has been adapted from Kumar, D. & Budman, H. 2015

glucose and acetate. Hjersted and Henson, 2006 used DFBM models representing the growth of *Saccharomyces Cerevisae* and Ethanol production on glucose for offline optimization of the fed-batch operation by implementing an optimal substrate feeding policy while reducing batch time or/and increasing productivity. A key advantage of DFBM models is that they require solving an LP problem with a relatively small number of rate limiting kinetic constraints as compared to other unstructured models that require the calibration of a larger number of kinetic expressions with many corresponding parameters. Thus, DFBMs are potentially less sensitive to experimental noise than other models but they can be sensitive to parametric uncertainty. Also, during fed-batch operation, a combination of factors such as non-ideal mixing or the presence of froth may contribute to additional model error and process disturbances. The importance of robustness for fed-batch bioreactor control has been stressed in (Kuhlmann *et al.*, 1998), due to i) time varying behavior, ii) un-modeled dynamics and iii) large disturbances occurring in the process.

Nagy and Braatz, 2007 have shown that Polynomial Chaos Expansions (PCE) is a computationally efficient alternative to Monte Carlo simulations for propagating uncertainty in dynamic models. The computational advantages of PCEs for robust control and optimization (Kumar and Budman, 2014, Nagy and Braatz, 2007, Kim *et al.*, 2012) derive from the availability of analytical formulae to compute the statistical moments (mean, variance, etc.) of variables described by such expansions.

In the current study, two applications of Polynomial Chaos Expansions to propagate uncertainty onto a quality of interest are pursued: i) an on-line robust optimal control for a bioreactor in which the economic objective is to maximize the amount of biomass at the end of the batch and ii) an off-line robust optimization of a fed-batch bioreactor using a probabilistic objective function. For both applications the process dynamics are modelled using DFBM and the parametric uncertainty is propagated using a PCE based approach. Since the DFBM model involves an LP, the resulting control strategy is obtained from the solution of a bi-level optimization problem involving the maximization of the economic objective subject to the LP solution. This bi-level optimization formulation poses challenges to the design of a robust strategy; and a PCE based approach is proposed to address them.

The proposed controller can be used in real-time application due to the low computational complexity resulting from the use of PCEs.

The manuscript is organized as follows. Section 4.2 introduces background material on DFBM and PCE which are then used in Section 4.3 to develop the robust-model predictive controller and Section 4.5 to formulate the robust optimization problem. Section 4.4 presents the control case study and Section 4.6 presents the robust optimization case study.

4.2 Mathematical Background

4.2.1 Dynamic Flux Balance Model

DFBM is based on an a priori known network of m metabolites, $\boldsymbol{z_{mx1}}$, participating in n different reactions. Each reaction is associated to a flux, $\boldsymbol{\nu_{nx1}}$ given in units of mM of metabolite/hr/mM of cell. This network of reactions can be mathematically expressed in terms of a stoichiometric matrix ($\boldsymbol{A_{mXn}}$) for the corresponding vector of reaction fluxes ($\boldsymbol{\nu_{nx1}}$). The DFBM approach assumes that the cell acts as an agent that strives to optimally allocate available resources (nutrients) to maximize a given objective, e.g. the cellular growth rate μ. Other optimization objectives have also been reported, e.g. the redox potential, but this study considers only the cell growth. Using the defined stoichiometric matrix and fluxes it is assumed that the cell maximizes the growth subject to constraints on fluxes or metabolites' concentrations as follows:

$$\max_{\boldsymbol{X,\nu,z}} \mu = \boldsymbol{w}^T \boldsymbol{\nu} \tag{4.1}$$

$$s.t. \ \frac{d\boldsymbol{z}}{dt} = \boldsymbol{A\nu}X, \qquad \frac{dX}{dt} = \mu X, \qquad \boldsymbol{A\nu} \leq \boldsymbol{b}$$

$$|\dot{\boldsymbol{\nu}}| \leq \dot{\boldsymbol{\nu}}_{\boldsymbol{max}}, \qquad \boldsymbol{\nu, z} \geq 0$$

where $\boldsymbol{b}_{mx1}$represents a vector of bounds on consumption or production rates of extracellular metabolites $\boldsymbol{z}$, i.e. nutrients and by-products, X, is the concentration of biomass, $\boldsymbol{w}$ is the contribution of each reaction flux towards cell growth (Mahadevan *et al.*, 2002). Constraints related to the flux rate or change of flux rate can be introduced so as to obtain better fitting of

the model to data. Dynamic flux models have not been used before for predictive control as proposed in the current work.

4.2.2 Polynomial Chaos Expansion

A Polynomial Chaos Expansion (PCE) describes a random process as a spectral expansion of random variables, (θ_i), using orthogonal basis functions, Φ_i (Ghanem and Spanos, 1990). For example, a second-order (finite variance) random variable y^d, is described using a PCE in (2), where a_{i1}^d are deterministic coefficients for each term in the expansion. Since the basis functions Φ_i, are orthogonal, the first term in ((4.2), a_0^d is the output (y^d) mean and the output variance can be obtained from $\sum_{i=1}^{\infty}(a_{i1}^d)^2 + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}(a_{i_1 i_2}^d)^2 +$ ... (Ghanem and Spanos, 1990, Ghanem and Spanos, 1997).

$$y^d = a_0^d\Phi_0 + \sum_{i_1=1}^{\infty} a_{i1}^d\Phi_1(\theta_{i_1}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2}^d\Phi_2(\theta_{i_1},\theta_{i_2}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3}^d\Phi_3(\theta_{i_1},\theta_{i_2},\theta_{i_3}) \quad (4.2)$$

For practical applications, random variables are approximated using truncated PCEs: $y^d \approx \sum_{i=1}^{n_{PCE}} a_i^d\Phi_{\mathrm{i}}(\theta)$. The truncated series are defined by: *i*) dimensionality, n_0, number of independent sources of random variables $(\theta_{\mathrm{i}_1}, \theta_{\mathrm{i}_2}, \theta_{\mathrm{i}_3})$ and *ii*) maximum polynomial order for the basis function, q, (dependent on the nonlinearity of the random process). The number of terms in the expansion n_{PCE}, is then given by $n_{PCE} = (n_0+q)!/(n_0!\,q!) - 1$. The basis functions Φ_i are chosen from the Askey scheme (Xiu and Karniadakis, 2002) depending on the type of stochastic distribution of the random variables (θ_i) that is considered in the model, e.g. Hermite functions are selected if θ_i is normal, so as to preserve orthogonality.

Given a process model with an uncertain output, $y = f(x;\lambda)$, where x is an input and λ is an uncertain parameter, the aim is to propagate the uncertainty in $\lambda(\theta)$ onto $y(\theta)$using the process model. Assuming that PCE's of $\lambda(\theta)$, are known a priori or are identified from data (Xiu and Karniadakis, 2002), a corresponding PCE expansion for the output can be calculated by a projection (inner product) operation with respect to each of the orthogonal

basis functions, Φ_i, (Xiu and Karniadakis, 2002). For example, after substitution of a PCE of $\lambda(\theta)$, into the equation $y = f(x;\lambda)$ and assuming $y(\theta) = \sum_{i=1}^{n_{PCE}} y_i \Phi_{\mathrm{i}}(\theta)$, the inner product of $y(\theta)$ with respect to each basis functions Φ_i is used to determine the i^{th}- PCE coefficient y_i as follows:

$$y_i = \frac{\langle y\Phi_{\mathrm{i}}\rangle}{\langle \Phi_{\mathrm{i}}^2\rangle} = \frac{\langle f(x;\lambda)\Phi_{\mathrm{i}}\rangle}{\langle \Phi_{\mathrm{i}}^2\rangle} \tag{4.3}$$

Two approaches referred to as non-intrusive and intrusive can be used to evaluate the inner product $\langle y\Phi_{\mathrm{i}}\rangle$. The non-intrusive method does not explicitly uses the model f and it only uses specific input and corresponding output values that define an empirical input-output mapping referred to as a surrogate model (Najm, 2009). In contrast, in the intrusive method, the integral $\langle f(x;\lambda)\Phi_{\mathrm{i}}\rangle$ in (4.3) is evaluated by using Galerkin projections where the PCE coefficients of $f(x;\lambda)$ in (4.3) are calculated analytically using the nonlinear dynamic model *f*. Both intrusive and non-intrusive methods are used in different steps in the current study. Additional details on the use of each method are provided in the next section.

4.3 Robust Control

4.3.1 Modeling with uncertainty

The goal of the current study is to develop a robust-MPC for a bioreactor operated both with feeding and perfusion; based on a DFBM model given in (4.1). During perfusion there is a continuous effluent stream from the bioreactor which only consists of the metabolites while all the biomass is retained in the bioreactor. To this purpose, dynamic mass balances that account for the feeding rate F, perfusion rate P and resulting volume changes can be written in terms of the fluxes' vector $\boldsymbol{\nu}$ (Eq.(4.4)- (4.6)), where, V is the volume of the reactor, $\boldsymbol{z}_{feed}$ is the concentration of metabolites in the feed and functions f and g are the RHS of ODE's for metabolites, $\boldsymbol{z}$ (Eq.(4.5)) and biomass, X(Eq.(4.6)) respectively. To solve for the fluxes' vector $\boldsymbol{\nu}$, a DFBM model is posed as an LP that can be solved at each time interval k

$$\frac{dV}{dt} = F - P, \tag{4.4}$$

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{A\nu}X + \frac{F\left(\boldsymbol{z_{feed}} - \boldsymbol{z}\right)}{V} = f(\boldsymbol{A}, F, V, P, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{\nu}, X, \boldsymbol{z}) \tag{4.5}$$

$$\frac{dX}{dt} = \mu X - \frac{X(F-P)}{V} = g(\boldsymbol{A}, F, V, P, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{\nu}, X, \boldsymbol{z}) \tag{4.6}$$

as shown in (4.7). The positivity constraints are obtained from the discretized versions of the ODE's in (4.4)- (4.6).

$$\max_{\boldsymbol{\nu}} \mu(k) = \boldsymbol{w}^T \boldsymbol{\nu}(k) \tag{4.7}$$

$$s.t.\ \boldsymbol{A\nu}(k) \leq \boldsymbol{b}(\boldsymbol{z}(\boldsymbol{k}-\boldsymbol{1}), \boldsymbol{\beta})$$

$$\left|\frac{\nu(k) - \nu(k-1)}{\Delta t}\right| \leq \dot{\nu}_{max},$$

$$\boldsymbol{\nu}(k), \boldsymbol{z}(k), X(k) \geq 0$$

The constraints in problem (4.7) involve bounds on flux rates, positivity of species' concentrations calculated from the discretized form of the process equations given in (4.4)-(4.6), Δt is a discretization time step, k, is the current time interval. The functional form of $\boldsymbol{b}$ depends on the type of metabolite; $\boldsymbol{b}$ can be zero implying that there is no external exchange nor accumulation of the metabolite, and in other cases it is a function of parameters such as uptake rates, substrate inhibition constant**,** and concentration of metabolites $\boldsymbol{z}$. As a result $\boldsymbol{b}(k-1) = b(\boldsymbol{\beta}, \boldsymbol{z})$ is a function of $\boldsymbol{z}(k-1)$. The material balance of $\boldsymbol{z}$ and X is used in continuous form (4.4)-(4.6) for calculating predictions for the controller until the end of the batch; and in their discretized form within the LP (4.7) for formulating the positivity constraints on metabolite concentrations, *i.e.* $\boldsymbol{z}(\boldsymbol{k}) >= 0$.

Rather than tracking a prescribed trajectory, an economic objective is used for control in this study consisting of maximizing the cellular amount at the end of the fed-batch $XV(t_f)$, where t_f represents batch end time. Also it is assumed that the biomass $X(k)$ and glucose, the main

metabolite, can be measured online. Thus, Eqs. (4.4)-(4.6) are used to predict $X(k)$ until the end of the batch and is solved in 2 steps; *1)* the LP is solved at every time step to determine $\boldsymbol{\nu}(k)$. *2)* Then, the ODE's in (4.4)-(4.6) are solved using the ode45 solver in MATLAB for calculating output predictions until the end of the batch.

In the current study the parameters' vector $\boldsymbol{\beta}$ is assumed as the main source of uncertainty in the model and hence it is characterized as a random variable by a PCE as $\boldsymbol{\beta} = \boldsymbol{\beta_0}\phi_0 + \boldsymbol{\beta_1}\phi_1 + \boldsymbol{\beta_2}\phi_2 = \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\beta_i}\phi_i$. In correspondence with the two step solution explained above for Problem (4.4)-(4.6) and (4.7), the uncertainty propagation is also done in two steps by noticing that $\boldsymbol{\beta}$ impacts the ODE's predictions through the LP in (4.7) as follows: step 1- a PCE expansion for the reaction fluxes can be determined, $\boldsymbol{\nu} = \boldsymbol{\nu_0}\phi_0 + \boldsymbol{\nu_1}\phi_1 + \boldsymbol{\nu_2}\phi_2 = \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i}\phi_i$ from the LP with a non-intrusive approach, step 2- The PCE expansions in $\boldsymbol{\nu}$ is substituted into the system of ODE's in (4.4)-(4.6) to solve for PCE expansions of $\boldsymbol{z}, X$ using an intrusive method. From this substitution it is possible to obtain ODE's for the PCE coefficients of $\boldsymbol{z}$ and X, $\boldsymbol{z_0}, \boldsymbol{z_1}, \boldsymbol{z_2}, X_0, X_1, X_2$ respectively. These two steps for uncertainty propagation are further described below.

4.3.1.1 Propagation of uncertainty onto fluxes (non-intrusive PCE approach) using LP in (4.7)

Because the constraints in the LP problem are nonlinear with respect to $\boldsymbol{\beta}$, the solution of the LP is nonlinear with respect to the coefficients of the PCEs representing these parameters $\boldsymbol{\beta}$. Hence, in the current study it is proposed to replace the LP with a surrogate model (non-intrusive PCE approach) for directly relating the PCE expansion of the uncertain parameter $\boldsymbol{\beta}$ to a PCE expansion of flux vector $\boldsymbol{\nu} = \boldsymbol{\nu_0}\phi_0 + \boldsymbol{\nu_1}\phi_1 + \boldsymbol{\nu_2}\phi_2$**.** This surrogate model needs to be developed in real-time for every time step in the prediction horizon. To this purpose, an input ($\boldsymbol{\beta}$) and outputs $\boldsymbol{\nu}$ mapping is created using samples of input values $\boldsymbol{\beta_j}$**,** and solving the LP_j, as shown in Eq. (4.8) derived from (4.7) for each of those values to determine corresponding output, $\boldsymbol{\nu_j}$. $\boldsymbol{\nu_j}$ are PCE coefficients of $\boldsymbol{\nu}$ as per the equality constraint in Eq. (4.8), where $\boldsymbol{\beta}_{j,Gauss}$ are specific collocation points necessary for Gaussian Quadrature,

$w_{j,i,Gauss}$ are standard Gaussian quadrature weights corresponding to $\boldsymbol{\beta}_{j,Gauss}$, n_{Gauss} is dependent on dimensionality of $\boldsymbol{\beta}$ and nonlinear dependence of $\boldsymbol{\nu}$ on $\boldsymbol{\beta}$.

$$LP_j = \max_{\boldsymbol{\nu_j}} \mu(k) = \boldsymbol{w}^T \boldsymbol{\nu}(k) \tag{4.8}$$

$$s.t. \boldsymbol{A\nu}(k) \leq \boldsymbol{b}\left(\boldsymbol{z}(\boldsymbol{k}-\boldsymbol{1}), \boldsymbol{\beta_j}\right)$$

$$\boldsymbol{\nu}(k), \boldsymbol{z}(k) \geq 0$$

$$\nu_i = \sum_{j=1}^{n_{Gauss}} w_{j,i,gauss} \nu_j(\boldsymbol{\beta_j}), i \in [1,..,n_{PCE}], j \in [1,2,\dots,n_{Gauss}]$$

4.3.1.2 Propagation of uncertainty in fluxes into the predictions of $\boldsymbol{z}$ and X

Assuming PCEs of $\boldsymbol{z}$ and X can be described by $\boldsymbol{z} = \boldsymbol{z_0}\phi_0 + \boldsymbol{z_1}\phi_1 + \boldsymbol{z_2}\phi_2$ and $X = X_0\phi_0 + X_1\phi_1 + X_2\phi_2$, this step consists of obtaining expressions for the PCE coefficients $\boldsymbol{z_i}, X_i, i \in [0,1,\dots n_{PCE}-1]$, by using the ODE's in (4.4)-(4.6). To this end, ODE's for each of $\boldsymbol{z_i}, X_i, i \in [0,1,\dots n_{PCE}-1]$, are obtained by substituting the PCE's for $\boldsymbol{\beta}$ (given), $\boldsymbol{\nu}$ from (4.8) and assuming PCEs for X and $\boldsymbol{z}$ into the functions f and g in equations (4.5) and (4.6) as follows:

$$\frac{d\boldsymbol{z}}{dt} = f\left(\boldsymbol{A}, F, P, V, \boldsymbol{z}_{feed}, \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{z_i}\phi_i \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i}\phi_i, \quad \sum_{i=0}^{n_{PCE}-1} X_i\phi_i, \boldsymbol{w}\right) \tag{4.9}$$

$$\frac{dX}{dt} = g(\boldsymbol{w}, \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i}\phi_i, \sum_{i=0}^{n_{PCE}-1} X_i\phi_i, F, P, V) \tag{4.10}$$

$$\frac{d\boldsymbol{z_0}}{dt} = \frac{\langle f\left(\boldsymbol{A}, F, P, V, \boldsymbol{z}_{feed}, \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{z_i}\phi_i \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i}\phi_i, \sum_{i=0}^{n_{PCE}-1} X_i, \boldsymbol{w}\right), \phi_0 \rangle}{\langle \phi_0^2 \rangle} \tag{4.11}$$

$$\frac{d\boldsymbol{z_i}}{dt} = \frac{\langle f\left(\boldsymbol{A}, F, P, V, \boldsymbol{z}_{feed}, \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{z_i}\phi_i \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i}\phi_i, \sum_{i=0}^{n_{PCE}-1} X_i, \boldsymbol{w}\right), \phi_i \rangle}{\langle \phi_i^2 \rangle} \tag{4.12}$$

Similar equations can be formulated to calculate the PCE coefficients for the biomass X.

$$\frac{dX_0}{dt} = \frac{\langle g\left(\boldsymbol{w}, \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i \phi_i}, \sum_{i=0}^{n_{PCE}-1} X_i, F, P, V\right), \phi_0 \rangle}{\langle \phi_0^2 \rangle} \tag{4.13}$$

$$\frac{dX_i}{dt} = \frac{\langle g\left(\boldsymbol{w}, \sum_{\boldsymbol{i=0}}^{\boldsymbol{n_{PCE}-1}} \boldsymbol{\nu_i} \phi_i, \sum_{i=0}^{n_{PCE}-1} X_i, F, P, V\right), \phi_\mathrm{i} \rangle}{\langle \phi_\mathrm{i}^2 \rangle} \tag{4.14}$$

4.3.1.3 Prediction with uncertainty until the end of the batch by combining step 1 and step 2 above

At a given time step k, provided that the feeding rate $F(k+l|k)$ and perfusion rate $P(k+l|k)\ \forall l \in [0,1,2 \ldots, n_f]$ are available, where $t_f = k + n_f$ and t_f corresponds to the end of the batch, then the steps for uncertainty propagation until the end of batch are as follows:

1. $i = 0, F(k+l|k)$, and $P(k+l|k)$ are known, so are $\boldsymbol{\beta}, \boldsymbol{\nu_i}(\boldsymbol{k-1})$, $\mu(k-1)$, $\boldsymbol{z_i}(k-1), X_i(k-1)$.
2. Determine $\boldsymbol{\nu_i}(\boldsymbol{k+l})$, $\mu_i(k+l)$ using $\boldsymbol{\beta}, \boldsymbol{\nu_i}(\boldsymbol{k+l-1})$, $\mu(k+l-1)$, $\boldsymbol{z_i}(k+l-1), F(k+l), P(k+l), X_i(k+l-1)$ by replacing LP with a surrogate model using Gaussian quadratures (section 4.3.1.1).
3. Determine $\boldsymbol{z_i}(\boldsymbol{k+l}), X_i(k+l)$ using $\boldsymbol{\beta}, \boldsymbol{\nu_i}(\boldsymbol{k+l})$, $\mu(k+l)$, $\boldsymbol{z_i}(k+l-1), X_i(k+l-1)$ and $F(k+l|k)$, $P(k+l|k)$ from step 2 above and section 4.3.1.2
4. If $l \geq n$, *then* break; *else* $l = l + 1$ and go to Step 2 and 3; *end if.*

4.3.2 Nominal Control Formulation

In the current study an economic objective function is used for fed-batch control that consists of maximizing the amount of biomass at the end of the batch, time t_f, $X\left(t_f = k + n_f\right) * V(t_f)$ by manipulating the nutrient feed-rate $F(k+l|k), \forall l \in [1,2,3 \ldots n_f]$ and the perfusion rate, $P(k+l|k), \forall l \in [1,2,3 \ldots n_f]$, for the process model in Eqs. (4.4)-(4.6). In the model (4.4)-(4.6), it is assumed that the amount of biomass, $X^{ms}(k)$ and the p^{th} nutrient manipulated by the controller $z_p^{ms}(k)$, can be measured at each time interval Δt, and are available for use as feedback by the controller. Thus, the open-loop prediction model can be updated with the feedback values $[X^{ms}(k)\ z_p^{ms}(k)]$, at every time step. The feedback,

$fb_k = X^{ms}(k) - X(k|k-1)$ is used to update predictions: $X(k+l) = g(k+l) + fb_k$, and the manipulated nutrient p is updated using the current measurement, $z_p(k|k) = z_p^{ms}(k)$. The resulting equations for the Nominal Control Problem, *i.e.* a controller that does not consider model error, with feedback are shown in (4.15). The problem posed in Eq. (4.15) is a bi-level optimization problem where the inner level (problem (4.7)) solves for the model fluxes and the outer level solves the control problem to determine the optimal feeding and perfusion rates.

$$\max_{F(k+l|k),P(k+l|k),k+l\leq t_f} XV(t_f) \tag{4.15}$$
$$s.t.\ Problem\ ((4.4)-(4.7))$$
$$fb_k = X^{ms}(k) - X(k|k-1)$$
$$X(t_f|k) = g(t_f|k) + fb_k$$
$$z_p(k|k-1) = z_p^{ms}(k)$$

With respect to time along the batch, it should be noticed that the outer level optimization seeks to maximize the biomass at the end of the batch whereas in the inner optimization level the growth is maximized at each instant. In the current work, the outer level is solved using *fmincon* in MATLAB, and the inner level (problem (4.7)) is solved with *linprog* in MATLAB.

4.3.3 Robust Control Formulation

The robust optimization involves a modified cost consisting of a weighted sum of the expectation and variance of a cost function. In the current work, since the uncertainty is propagated using PCE, both the expectation and variance can be quickly calculated online using analytical expressions. Similar to the nominal control formulation given in (4.15), the biomass prediction is updated using feedback, $fb_k = X^{ms}(k) - X_0(k|k-1)$ with only the nominal prediction $X(k+l) = g(k+l) + fb_k$ and assuming remaining PCE coefficients $X_i(k|k) = 0$, where $i \in [1,2,\dots n_{PCE}-1]$; the manipulated nutrient p is updated using the current measurement, $z_{p,0}(k|k) = z_p^{ms}(k)$ and similarly $z_{p,i}(k|k) = 0$.

$$\max_{F(k+l|k),P(k+l|k)} w_1 E\left(XV(t_f)\right) - w_2 Var\left(XV(t_f)\right) \quad (4.16)$$

$$s.t. Eqs.(4.7) - (4.14)$$

$$fb_k = X^{ms}(k) - X_0(k|k-1)$$

$$X_i(k|k-1) = 0, i \in [1,2,\dots n_{PCE} - 1]$$

$$X_0(t_f|k) = X_0(t_f|k)\ from\ (5-12) + fb_k$$

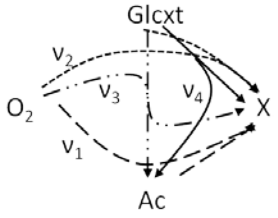
$$\hat{z}_{p,0}(k|k-1) = z_p(k)$$

$$\hat{z}_{p,i}(k|k-1) = 0, i \in [1,2,\dots n_{PCE} - 1]$$

Eq. (4.16) presents the robust control problem formulation with economic objective function and uncertain model predictions where w_1, w_2, represents the weights assigned to the nominal and robust performance respectively. The problems in (4.15) or (4.16) are solved with a combination of *linprog* and *fmincon* in MATLAB for the inner and outer optimizations respectively.

4.4 Case Study on Robust Control

To illustrate the proposed controller, a simplified DFBM model developed by Mahadevan *et al.*, 2002 for growth of *E.coli* on glucose is used. Figure 4.1 shows the simplified metabolic network, with glucose (Glcxt), acetate (Ac) and oxygen (O_2) as the input and biomass (X) as the output.



$$\nu_1 \quad 39.43\ Ac + 35\ O_2 \rightarrow X$$

$$\nu_2 \quad 9.46\ Glcxt + 12.92\ O_2 \rightarrow X$$

$$\nu_3 \quad 9.84\ Glcxt + 12.73\ O_2 \rightarrow 1.24\ Ac + X$$

$$\nu_4 \quad 19.23\ Glcxt \rightarrow 12.12 Ac + X$$

Figure 4.1 Simplified Metabolic Network for *E.Coli* growth on Glucose: Flux balances and stoichiometric coefficients

It consists of 4 fluxes given by the vector $\boldsymbol{\nu}$ and 3 metabolites given by the vector $\boldsymbol{z}$ (Glcxt, Ac, O_2). The growth rate, μ as a function of the fluxes and A_{3x4}, the stoichiometric matrix related to the 3 metabolites participating in the reactions leading to the biomass growth, are presented as follows:

$$\boldsymbol{A} = \begin{bmatrix} 0 & -9.46 & -9.84 & -19.23 \\ -35 & -12.92 & -12.73 & 0 \\ -39.43 & 0 & 1.24 & 12.12 \end{bmatrix} \tag{4.17}$$

and, $\mu = \sum_{i=1}^{4} \nu_i$

Eq. (4.18) and (4.19) represents the corresponding mass balances and DFBM model for *E.Coli* growth on Glucose and Acetate. In a batch reactor, there are 3 distinct growth phases of *E.coli*, viz. i) Aerobic growth on Glucose, ii) Anaerobic growth on Glucose, and iii) Anaerobic growth on a second metabolite, acetate. In the current study the original model (Mahadevan et al., 2002), has been expanded with two additional parameters: glucose inhibition constant K_I, and the effect of perfusion rate P. As generally reported in the literature the parameter K_I is used to describe the inhibitory effect of high concentration of glucose on growth. The perfusion rate P ensures that the negative impact on growth by accumulation of high levels of acetate and glucose is avoided. Eq. (4.18) and (4.19) can then be used with the economic objective function proposed in (4.15) to formulate a nominal controller that uses feedback signals of $X^{ms}(k)$ and $Z_{gl}^{ms}(k)$. The uncertainty in the model is assumed for simplicity to be associated to the maximum uptake rate constraints, GUR_{max}. The robust controller is then developed by extending the nominal model of Eq. (4.18 and (4.19)) to account for robustness as described insection4.3.

$$\frac{dz_{Gl}}{dt} = \boldsymbol{A\nu}X + F\left(\boldsymbol{z_{Gl,in}} - z_{Gl}\right) \tag{4.18}$$

$$\frac{dz_{O2}}{dt} = A^{O2}\boldsymbol{\nu}X - \frac{Fz_{O2}}{V} + k_L a(0.21 - z_{O2})$$

$$\frac{dz_{Ac}}{dt} = A^{Ac}\boldsymbol{\nu}X - \frac{Fz_{Ac}}{V}$$

$$\frac{dX}{dt} = \mu X - \frac{X(F-P)}{V},$$

$$\frac{dV}{dt} = F - P$$

$$\max_{X,\boldsymbol{\nu}_i} \quad \mu = \sum_{i=1}^{4} \nu_i \tag{4.19}$$

$$z_{i,} \geq 0, \forall i \in [1,3], \quad \nu_i \geq 0, \forall i \in [1,4]$$

$$|A^{Glcxt}\boldsymbol{\nu}| \leq \frac{GUR_{max}\, Z_{Glcxt}}{K_m + Z_{Glcxt} + \frac{Z_{Glcxt}^2}{K_I}} \frac{mmol}{gdw - hr}$$

$$-A^{O2}\boldsymbol{\nu} \leq OUR_{max},$$

$$A^{Ac}\boldsymbol{\nu} \leq 100$$

To test the controller in terms of its ability to reject disturbances, changes in the mass transfer coefficient $k_L a$ are considered as unmeasured disturbances to the process. Then, the objective is defined as to maximize the biomass at the end of the batch by manipulating the glucose feeding and the perfusion rate in the presence of disturbances in $k_L a$. To develop the robust controller, a PCE for the uncertain parameter, GUR_{max}, is assumed to be a priori known. Hermite Polynomials are chosen as the basis functions for the PCE expansion of GUR_{max}which is assumed to be a normally distributed variable with PCE coefficients β_0 , β_1 that can be shown to be equal to the mean and variance of GUR_{max} respectively (Ghanem and Spanos, 1990). Then, $n_{dim} = 1$ and $n_{order} = 2$ are used for uncertainty propagation using PCE. Table 4.1 shows the nominal parameter values along with the variance used for the case study. The changes in $k_L a$ are assumed to be of sinusoidal form with amplitude of 0.05 and with different mean values defined below.

The goal of the case study is to compare the performance of nominal and robust controller in terms of their disturbance rejection ability. The weights in the cost function in Eq. (4.16) were kept constant to $w_{robust} = 10, w_{nominal} = 20$. Three different mean values of disturbances are used to compare the controllers' performances (2.4, 4.0 and 5.6). To check for the effect of GUR_{max}, 6 different values of $GUR_{max}$are considered where each of these values remains constant along a fermentation i.e. during one run of the batch GUR_{max}does not changes its value.

Table 4.1 : Process Parameters for *E.Coli* growth on Glucose and Acetate used for Robust/Nominal Controller

Name	Value
$[\overline{k_L a}, \sigma_{k_L a}$	$[4.0, 0.8]hr^{-1}$
K_m	$0.015\ mM$
$[GUR_{max}, \sigma]$	$[6.5, 1.3]mM/g$
OUR_{max}	$12.0\ mM/g - dw/hr$
$t_f, \Delta t$	$[11.0, 0.5]hr$
$z_{Gl,in}, z_{Ac,in}, z_{O2,in}$	$[5.00, 0, 0]$
$[V_{min}, V_{max}]$	$[0.2L, 0.4L]$
$[z_{Gl,0}, z_{O2,0}, z_{Ac,0}]$	$[0.40, 0.21, 0.20]$
$[X_0, V_0]$	$[1e-3, 0.3\ L]$

Table 4.2: Robust Controller vs Nominal Controller Performance

GUR_{max}	$k_{La} = 2.4$		$\overline{k_{La}} = 4.0$		$k_{La} = 5.6$	
	Robust Cost	Nominal Cost	Robust Cost	Nominal Cost	Robust Cost	Nominal Cost
3.90	0.0222	0.0214	0.0202	0.0204	0.0250	0.0211
5.20	0.0532	0.0534	0.0653	0.0558	0.0524	0.055
5.85	0.0732	0.0748	0.0877	0.0931	0.0855	0.0899
6.50	0.1015	0.1063	0.123	0.1213	0.1227	0.1268
7.15	0.1153	0.1198	0.1114	0.0958	0.1385	0.1108
7.80	0.1255	0.0791	0.1407	0.0805	0.1284	0.0812
Ratio	1.09		1.17		1.15	

In the closed-loop control calculations the robust model prediction uses the nominal value and the associated uncertainty information whereas the nominal model, used for the nominal controller, only uses the nominal value of 6.5 for GUR_{max}. Both the nominal and robust controllers use the nominal mean value of the disturbance i.e. $\overline{k_{LA}} = 4.0$. Thus, a total of 18 cases are simulated for the nominal and robust controller corresponding to the different combinations of values of $GUR_{max}$and $k_L a$. Table 4.2 lists the amount of biomass produced at the end of the batch using the nominal and robust controller respectively. It is evident looking from the biomass values, that the biomass production is more sensitive to GUR_{max} than to $k_L a$. Also when increasing the value of $GUR_{max}$for the plant, the total biomass production ($X_{t_f} V_{t_f}$) increases, irrespective of the kind of controller. Accordingly, it is important that when the actual value of $GUR_{max}$is low, the biomass production will meet certain production targets, or else the entire batch production might have to be discarded. In that case the robust controller becomes particularly effective since it predicts the possibility that productivity will be low thus increasing the glucose feeding beyond the amount calculated by the nominal controller. For instance, when $GUR_{max} \approx 3.9$, the average performance of the robust controller for three disturbances is better than the nominal controller by 7%. For the 6 cases where $GUR_{max} > GUR_{max,nominal}$, the robust controller performs better than nominal controller. Also, since the robust controller accounts for the case that GUR_{max} can be large, it avoids overfeeding of glucose that results in growth inhibition. For the remaining cases where $GUR_{max} \approx GUR_{max,nominal}$, the robust controller performance is similar to that of the nominal controller. The ratio of the average cost of the robust and the nominal controllers for each mean value of the disturbance is shown at the bottom of Table 4.2 indicating consistently better productivity (cost) for the robust controller with an average improvement of ~15%, that can be very significant in bio-manufacturing operations.

Figure 4.2 shows a typical Feeding and Perfusion profile for both the robust and nominal controllers for $GUR_{max} = 5.2$ and for the process disturbance corresponding to a mean value of $k_L a = 4.0$. In the initial phase until $t < 6\,hr$, biomass growth occurs on glucose and the

nominal and robust controllers show similar performance. During the interval $6 \leq t \leq 9\ h$, the nominal controller starts both feeding and perfusion, resulting in high z_{Gl} for that time period, as compared to the robust controller, which only starts feeding at comparatively lower rates so as to avoid glucose inhibition. Also, during $6 \leq t \leq 9\ h$, the robust controller maintains a constant level of z_{Gl}, indicating a balance between the metabolic glucose uptake rate and glucose from the feed while the nominal controller results in higher glucose levels which are inhibiting the growth. In the last phase of the batch, $t \geq 9h$, both the robust and the nominal controller increase feeding rate of glucose significantly in order to maximize the final biomass amount.

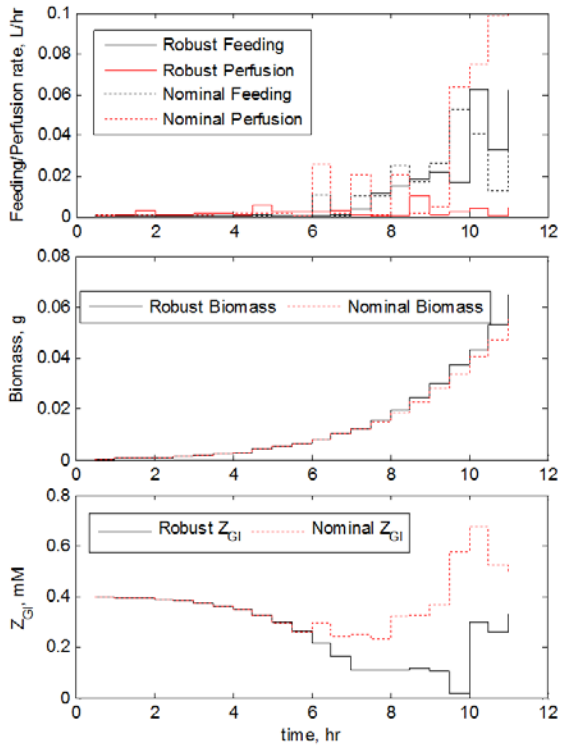


Figure 4.2: Robust vs. Nominal Controller: Feeding, Perfusion, Biomass and Glucose trajectories

4.5 Robust Optimization

An off-line robust optimization approach is proposed for a fed-batch bioreactor using probabilistic objective function. Due to the plant-model mismatch and the disturbances

occurring in batch processes, robust optimization has become important for improving process productivity in the presence of model error (Srinivasan et al., 2003). Different approaches to robust optimization have been reported depending on the availability of measurements along the batch (Srinivasan et al., 2003). In the absence of measurements a single offline-robust optimization calculation can be performed to obtain an optimal feeding recipe. On the other hand if measurements are available along the batch, they can be used to counter the effect of uncertainties for adapting the model to be used for subsequent batches or for on-line feedback calculations (Mandur and Budman, 2015, Srinivasan and Bonvin, 2007, Srinivasan et al., 2003) as done for the robust controller presented above.

In this section we assume that measurements are not available during the batch and therefore a single off-line robust optimization calculation is performed to obtain an optimal feeding recipe. For most of the off-line robust optimization techniques uncertainty propagation methods are required and then a probability distribution of the objective function is used to define the cost. Studies have been proposed where the objective function consists of i) the expected or extremum value of a terminal property Dewasme et al., 2011, ii) a worst-case scenario of the cost, Ma et al., 1999, iii) a weighted function of the expected value and variance of the terminal property, Nagy and Braatz, 2003, Nagy and Braatz, 2004, iv) a probabilistic objective function to meet a certain quality criteria, Terwiesch et al., 1998, or v) a linearization of the objective function around the nominal conditions combined with bounds of the model uncertainties Logist et al., 2011. Most of these mentioned studies either rely on first-principles model for uncertainty quantification and propagation or use Monte Carlo sampling methods (which is computationally heavy). All constraints in these formulations are transformed to corresponding robust counterparts. On the other hand if measurements are available along the batch, they can be used to counter the effect of uncertainties by adapting the model to be used for subsequent batches (Mandur and Budman, 2015, Srinivasan and Bonvin, 2007, Srinivasan et al., 2003). In the current work, PCE's is used for uncertainty quantification and propagation, which are known to facilitate quick computation of statistical measures and uncertainty propagation.

In the current study a probabilistic based objective function is used to provide a minimum amount of biomass at the end of the batch. A surrogate model for the biomass amount is developed using a non-intrusive PCE approach, which is then used to define the statistical measures required for the objective function. Section 4.2.1 and 4.2.2 already introduced the background material on DFBM and PCE which is used to develop the robust-model for optimization. Section 4.6 presents a comparison of the robust and nominal optimization performances for an *E.Coli* fermentation case study.

4.5.1 Modeling with uncertainty

The goal of the robust optimization is to compute optimal recipes for both feeding and perfusion rates whereby a DFBM model given in (4.1) is used to model the process dynamics. The DFBM model for the bioreactor has been described by equations (4.4)-(4.7).

As shown in section 4.3, this model can be turned into a robust model using PCE via 2 steps, *i)* first developing non-intrusive models for the fluxes $\boldsymbol{\nu}$ at every time step k, *ii)* And then propagating the uncertainty into ODE's for $\boldsymbol{z}, X, V$ using intrusive methods. However, for the purpose of robust optimization (RO), if the objective is solely to optimize an end-point property then it is sufficient to formulate a robust model relating the end-point property of interest to the decision variables and uncertainties thus by-passing the need to develop PCE models for each of the metabolites as done above for robust control. In the current case study the probability for meeting a minimum biomass amount, $XV(t_f)$, is maximized. Towards this goal, a robust model is developed using a non-intrusive PCE to determine $XV(t_f)$ for a given combination of the decision variables, i.e. feeding (F) and perfusion (P) rates of glucose to the bioreactor, as a function of the parametric uncertainties. This non-intrusive PCE is often referred in the literature as a surrogate model since it replaces the original first-principles based model.

In the current study variations in elements of the parameters' vector $\boldsymbol{\beta}$ are assumed to be the main source of uncertainty in the model. To this purpose, a map between the input ($\boldsymbol{\beta}$) and outputs X is created using samples of input values $\boldsymbol{\beta_j}$, and solving the Eqs. (4.4)-(4.7) for each of those $\boldsymbol{\beta_j}$, to determine the corresponding output, X_j. X_i are PCE coefficients of X

calculated by Eq. (4.20), where $\boldsymbol{\beta}_{j,Gauss}$ are specific collocation points necessary for Gaussian Quadrature, $w_{j,i,Gauss}$ are standard Gaussian quadrature weights corresponding to $\boldsymbol{\beta}_{j,Gauss}$, n_{Gauss} is chosen based on the dimensionality of $\boldsymbol{\beta}$ and nonlinear dependence of X with respect to $\boldsymbol{\beta}$. Equation (4.21) represents the surrogate model h relating end-point biomass with the uncertain parameters $\boldsymbol{\beta_j}$, for given combinations of feeding and perfusion rates. Figure 4.3 pictorially depicts how the distribution of uncertain parameters $\boldsymbol{\beta_j}$ is used to develop the surrogate model h for $X(t_f)$. The calculation of X and V in the cost $XV(t_f)$ is done separately. A surrogate model is created as explained above (and shown in section 4.3.1.1) for a given combination of F and P, between the uncertain parameters $\boldsymbol{\beta}$ and the output $X(t_f)$ whereas the volume V is calculated separately with the overall mass balance Eq. (4.18) as a function of $F$and P.

$$X_i = \sum_{j=1}^{n_{Gauss}} w_{j,i,gauss} X_j(\boldsymbol{\beta_j}), i \in [1,..,n_{PCE}], j \in [1,2,\dots,n_{Gauss}] \tag{4.20}$$

$$X(t_f) = h(\boldsymbol{\beta_j}, F, P) \tag{4.21}$$

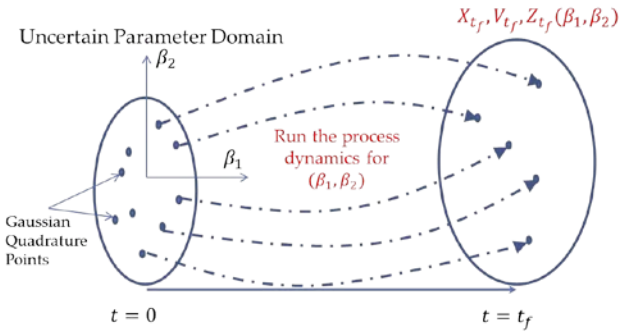


Figure 4.3: Input-output mapping to develop PCE for X_{t_f}

4.5.2 Nominal Optimization Formulation

The objective is to maximize the amount of biomass at the end of the batch, i.e $XV(t_f)$. Hence in the case of nominal optimization, it is same as the nominal control problem solved

in section 4.3.2 at the first time step when no feedback is considered. The equations for nominal optimization problem are as follows:

$$\max_{F(k),P(k),0\leq k\leq t_f} XV(t_f)$$
$$s.t.\ Problem(4.4)-(4.7) \tag{4.22}$$

Problem (4.22) is a bilevel optimization problem, where the inner level problem solves for the model fluxes and the outer level maximizes production to determine the optimal feeding and perfusion rates' profiles. It should be noticed that the outer level optimization seeks to maximize the biomass at the end of the batch whereas in contrast, in the inner level problem the growth is maximized at every time step. In the current work, the outer level optimization (4.22) is solved using *fmincon* in MATLAB, and the inner level problem (4.7) is solved with *linprog* in MATLAB.

4.5.3 Robust Optimization Formulation

Since bioreactors' in pharmaceutical industries are generally costly to operate, it becomes critical to meet certain minimum productivity at the end of each batch. Based on this target a probabilistic objective function is used for the purposes of robust optimization, whereby the probability of producing a minimum amount of product is maximized with respect to the feeding F and perfusion P profiles. It is assumed in this work that the biomass is the product of the process under consideration. The objective function used for robust optimization is derived from a Chebyshev Inequality, as shown in (4.23).

$$Pr[|Y-E[Y]|\geq\lambda]\leq\frac{Var[Y]}{\lambda^2}\quad or,$$
$$Pr[Y\geq\lambda]\leq\frac{Var[Y]}{(E[Y]-\lambda)^2} \tag{4.23}$$

Where, Y is a random variable and λ is a threshold chosen to be greater than $(Var[Y])^{0.5}$. In the current work, since the uncertainty is propagated using PCE, both the expectation and variance can be quickly calculated using analytical expressions as compared to alternative Monte Carlo approaches that are computationally costly. The random variable in the current

study is the end-point total biomass $XV(t_f)$ and the threshold value λ is the biomass amount at the end of the batch $\lambda = XV_{min}(t_f)$. Thus the robust optimization problem can be formulated as follows:

$$\max_{F(k),P(k),0\leq k\leq t_f} \frac{Var[XV(t_f)]}{\left(E[XV(t_f)] - \lambda\right)^2} \quad (4.24)$$

$$s.t.\, Eq.\,(4.20)\; and\; (4.21)$$

In Eq. (4.24), $\lambda = XV_{min}(t_f)$ is chosen based on process knowledge. To solve problem (4.24) a surrogate model h needs to be developed at every iteration of the optimization search. Robust optimization problem involves two important steps. First, for a given combination of F and P, develop a surrogate model h for $XV(t_f)$ using the information about the uncertain parameter $\boldsymbol{\beta}$. The number of input-output samples that should be used for developing h, depends on nonlinearity of the model and the sources of uncertainty $\boldsymbol{\beta}$. Second, analytical formulae for the statistical means are used in the objective function. In the current study, only the parameters are considered uncertain. However in general bioreactor applications the initial concentration of biomass in the reactor is often uncertain due to lack of sensor sensitivity and the proposed approach can be extended to include this uncertainty as well. Based on the surrogate model, problem (4.24) can be solved to determine the probability of meeting the minimum threshold λ. This problem is solved using *fmincon* in MATLAB.

4.6 Case Study on Robust Optimization

To illustrate the proposed robust optimization scheme, the simplified DFBM model developed by Mahadevan *et al.*, 2002 for growth of *E.coli* on glucose presented in Section 4.4 is used.

Two parameters, mass transfer coefficient $k_L a$, and maximum glucose uptake rate GUR_{max} are considered to be uncertain and as a result $n_{dim} = 2$. To account for nonlinearity between the uncertain parameters, $k_L a$ and GUR_{max}, and $X(t_f)$, the order for PCE is chosen as $n_{order} = 2$. Hermite polynomials are chosen as the basis functions for the PCE expansion of

$X(t_f)$. Table 4.3 shows the nominal parameter values along with their variances used for the case study.

Table 4.3 : Process Parameters for *E.Coli* growth on Glucose and Acetate for Robust/Nominal Optimization

Name	Value
$[\overline{k_L a}, \sigma_{k_L a}]$	$[4.0, 0.8] hr^{-1}$
K_m,	$[0.015\ mM]$
$[GUR_{max}, \sigma]$	$[6.5, 1.3] mM/g$
OUR_{max}	$12.0\ mM/g - dw/hr$
$t_f, \Delta t$	$[11.0, 0.1] hr$
$z_{Gl,in}, z_{Ac,in}, z_{O2,in}$	$[5.00, 0, 0] mM$
$[V_{min}, V_{max}]$	$[0.2L, 0.4L]$
$[z_{Gl,0}, z_{O2,0}, z_{Ac,0}]$	$[0.40, 0.21, 0.20] mM$
$[X_0, V_0]$	$[1e-3 g/L, 0.3\ L]$
K_I	$[1.0\ mM/g - dw/hr]$
λ	$[0.03]\ g$

The goal of the case study is to compare the performance of the nominal and robust optimization formulations in terms of number of cases for which either the robust or nominal optimal feeding/perfusion profiles failed to meet the minimum biomass amount. The nominal optimization problem (4.22) is solved to maximize the amount of biomass at the end of the batch using nominal operating conditions and hence disregarding any kind of uncertainty in the system. The robust optimization problem (4.24) is solved to maximize the number of batches meeting the minimum biomass at the end of the batch condition, while accounting for the uncertainty in $k_L a$ and GUR_{max}. $F_{nominal}$and $P_{nominal}$ denote the solution of Problem (4.22) for feeding and perfusion respectively, and similarly $F_{robust}$and P_{robust} denote the solution to Problem (4.24). These solutions are shown in Figure 4.6 for the parameters shown in Table 4.3. Both the solutions are then used to determine the distribution of $X(t_f)$. The latter is calculated by first creating sample input space of uncertain parameters $\boldsymbol{\beta_j}$consisting of 1000 points, and then using the simulated model equations for the calculated optimal feeding and perfusion profiles. The histograms of $X(t_f)$ for the robust and nominal optimizations respectively are shown in Figure 4.5. It can be observed that the distribution of

$X(t_f)$ for nominal optimization is near uniform but for the robust optimization the distribution is closer to a normal distribution. The amount of violation of the lower constraint imposed on biomass can be assessed from the cumulative distributions which are shown in Figure 4.4. In the case of the nominal optimization $X(t_f) \geq 0.03$, the feeding profiles resulted in a violation of the lower constraint in ~18% (Figure 4.4) of the runs while the robust optimization resulted in violation of this constraint for only 4% (Figure 4.4) of the cases.

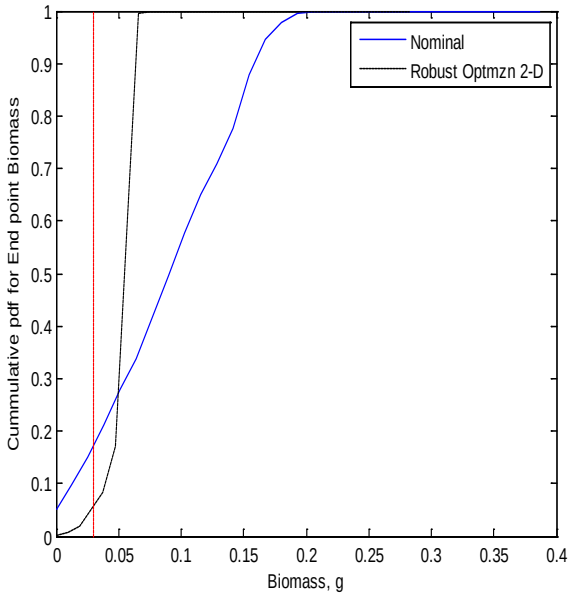


Figure 4.4: Cumulative pdf of $X(t_f)$ for Nominal and Robust Optimization

The improved performance of the robust optimization versus the nominal optimization can also be interpreted from the differences in the feeding and perfusion profiles for the two approaches (shown in Figure 4.6). In both the cases there are three distinct phases of time to characterize the process behavior. In the initial phase $t \leq 6h$, both for the nominal and the robust optimization there is negligible amount of feeding and perfusion since glucose is present and biomass growth is primarily occurring on the available glucose. During the second phase $6 \leq t \leq 9h$, $F_{nominal}$shows a sudden increase to compensate for the nominal depletion of glucose while $P_{nominal}$ is kept at low levels. On the other hand for the solution of the robust optimization there are minor changes in $F_{robust}$and P_{robust}, and both are kept

close to each other. High $F_{nominal}$ during this second phase has a significant impact on the distribution of $X(t_f)$. Since GUR_{max} is assumed to be uncertain, for the cases when

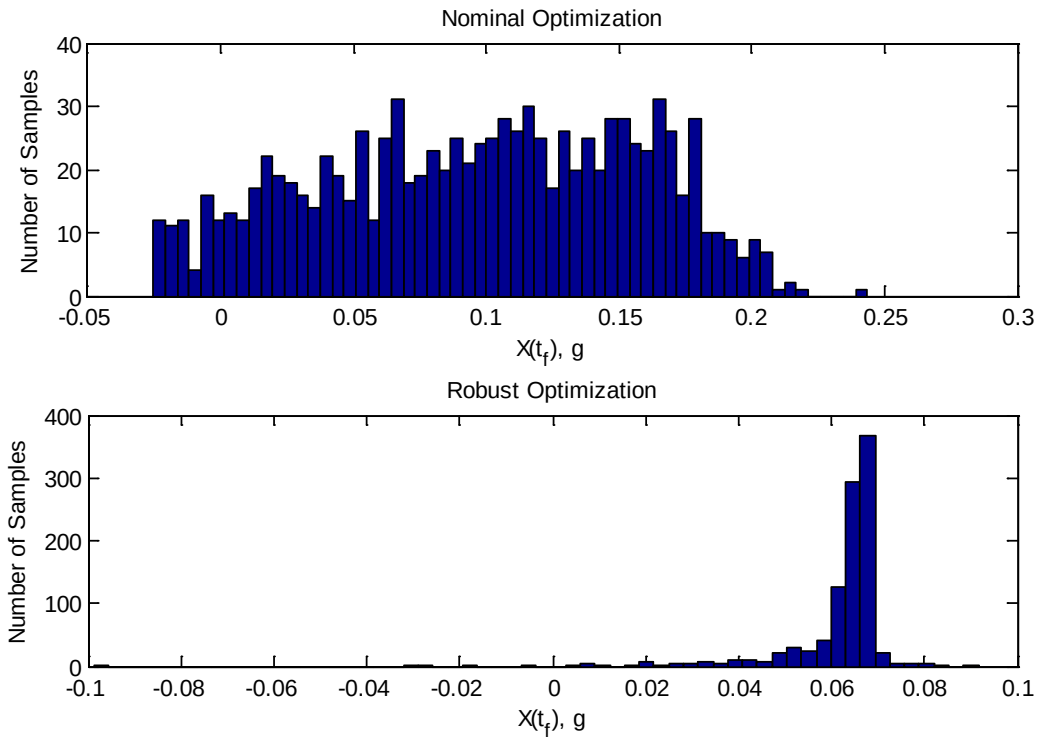


Figure 4.5 Histogram of $X(t_f)$ for Nominal (top) and Robust (bottom) Optimization

$GUR_{max} < GUR_{max,nom}$; the remaining glucose concentration Z_{gl} may be very high and additional glucose feeding will lead to significant growth inhibition and reduced amount of biomass at the end. The robust algorithm correctly predicts the possibility of glucose overfeeding thus the glucose feeding required by the robust solution is lower. However, high $F_{nominal}$during second phase, may lead to gains in the maximum amount of biomass that can be produced using $F_{nominal}$ and $P_{nominal}$ for the cases that GUR_{max} is close to $GUR_{max,nom}$.

During the third phase $t \geq 9h$, all the rates $P_{robust}, F_{robust}, P_{nominal}$ and $P_{nominal}$, increase significantly since by then the initial glucose amount has been depleted thus necessitating additional feeding in order to maximize the biomass amount at the end of the batch.

4.7 Conclusions

Robust and nominal control algorithms are presented for a fed-batch reactor modeled using DFBM model for maximizing an economic objective function. The economic objective was chosen as the final biomass amount. The robust controller showed better final productivity (biomass level) than the nominal controller by an average of 13% that can be significant in bio-manufacturing. The use of PCE to propagate parametric uncertainty is computationally very efficient as compared to Monte Carlo and thus it is instrumental for online implementation of the proposed robust algorithm. The controller requires the use of feedback corrections in glucose and biomass. The implementation of the algorithm requires the use of both an intrusive PCE model to predict the effect of uncertainty on the metabolites and a non-intrusive PCE model for end of batch biomass.

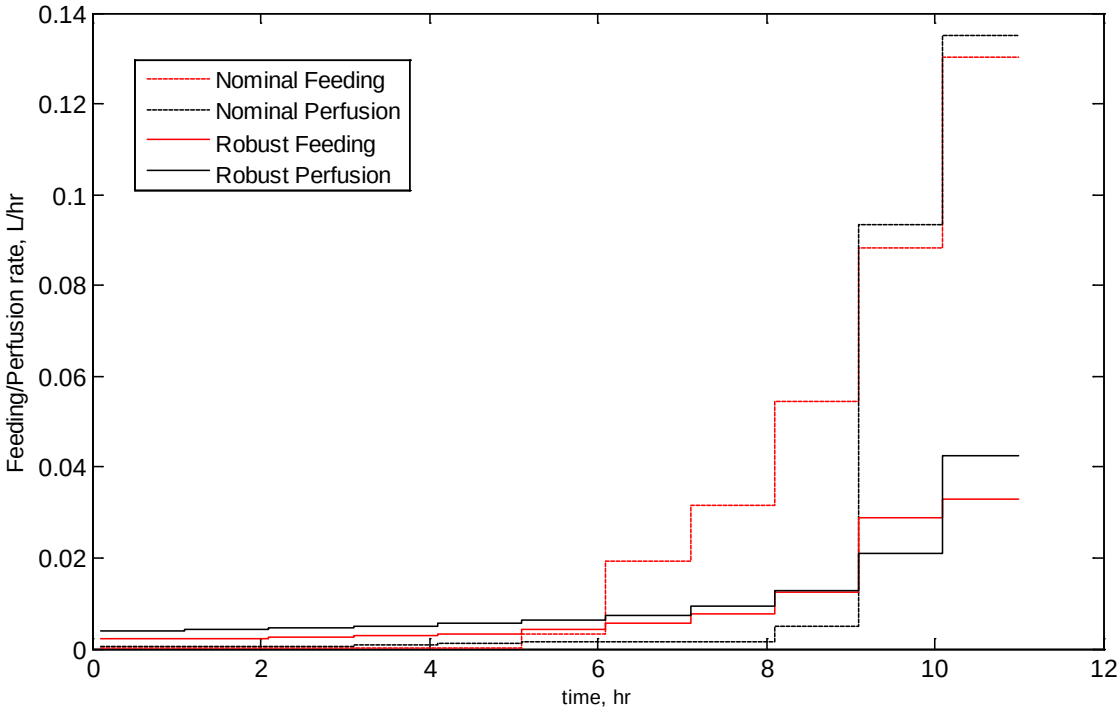


Figure 4.6 Feeding and Perfusion profiles for Robust and Nominal Optimization

The offline-robust optimization approach presented is based on the minimization of the violation of a lower bound on end point biomass level. A surrogate model of end-point biomass is developed using non-intrusive PCE approach which permitted quick calculation

of the variance by analytical formula. The nominal optimization failed to meet the minimum end-point biomass lower limit for 18% of the cases as opposed to 4% violation for robust optimization.

Both the robust optimization and control algorithms have shown superior performance to their nominal counterparts. In practice the choice between robust control and robust optimization will depend on the degree of confidence of plant personnel on the available measurements and on process certification constraints. In the past, pharmaceutical companies have generally avoided on line control in manufacturing operations due to safety concerns and to comply with tight operating procedures that resulted from long certification procedures. In these cases, a robust optimization approach will be more suitable for implementation. However, recent changes in FDA guidelines have increased the acceptability of on-line control strategies of pharmaceutical processes such as the one proposed in this work. The approach presented in robust optimization can also be extended to optimize for a quality property along the process, however, this will require a combination of both intrusive and non-intrusive propagation of uncertainty using PCE.

Chapter 5
Robust Distributed MPC using robust observer during communication loss[3]

The initial motivation for doing robust-distributed MPC was to implement the PCE-based RNMPC presented in Chapter 3 for a nonlinear system in a distributed fashion, where the robust control would account for the uncertainty in plant-model mismatch and also the errors in model interactions occurring between each of the subsystems. However since the PCE-based RNMPC turned out to be computationally efficient the work in this Chapter was done for linear systems but it was not expanded further for the PCE based NMPC algorithm. Thus, the integration of a robust observer with PCE-based RNMPC, where the robust observer can be used to determine the bounds on model interactions has been left for future work. Part of the work presented in this Chapter has been accepted as a refereed publication in the conference proceedings of ADCHEM 2012. That paper included a simplified version of the current Chapter and it only presented the distillation column example. Since that conference, the solution of the robust observer has been improved by integrating MATLAB with an external bilinear solver from YALMIP. Also, the methodology based on the bilinear optimization has been applied to a larger system of 9 states and 4 inputs in a reactor-separator system to illustrate the computational feasibility of the approach for systems with several states and inputs.

5.1 Introduction

The use of one central controller to control highly interconnected process units in chemical plants is often computationally challenging and difficult to implement. Hence, a more practical approach is to partition the process into smaller subsystems and to design lower dimensional controllers for each subsystem (Scattolini, 2009). This distributed control approach referred to as Distributed Model Predictive Control (DMPC) has gained significant attention from the research community with various algorithms being proposed which can be

[3] Part of this work is adapted from Kumar, D., Al-Gherwi, W. & Budman, H. 2012

broadly classified according to different features: i) type of cost function (local vs. global), ii) solution procedure being used (non-iterative, single iteration) and iii) degree of exchange of information between the subsystems (Scattolini, 2009). Venkat *et al.*, 2005 proposed a method of cooperative distributed control that permits to recover the performance of a centralized controller when the proposed iterative algorithm converges. Other DMPC algorithms have been reported (Zhang and Li, 2007, Liu *et al.*, 2009, Scheu and Marquardt, 2011).

Considering the importance of the model for MPC algorithms in general and DMPC in particular, it is also important to consider robustness to plant-model mismatch. To provide for robustness Al-Gherwi *et al.*, 2011 assumed the plant model of each subsystem to be included within a polytopic model and the control action was based on the minimization of a robust performance bound where this latter minimization is conducted iteratively for every subsystem in a cooperative manner. Other robust DMPC algorithms use the "tubes" concept (Trodden and Richards, 2006) that consists of developing invariant regions (tubes) at each time instant for linear time invariant models with interactions between subsystems treated as bounded disturbances. However, plant-model mismatch has not been explicitly included as yet in the tubes' based approach.

A key requirement in most previously proposed DMPC algorithms is the exchange of state information at the beginning of all iterations. Since this exchange is required at every time step, a loss in communication due to dropped packets or poor signal needs to be explicitly addressed (Rawlings and Stewart, 2008). Using a nominal model based estimator, de la Pena and Christofides, 2008 accounted for communication loss within the cost function of an MPC algorithm with guaranteed stability and included constraints for the length of the data loss period. Maestre *et al.*, 2009 designed DMPC where each agent (sub-system) calculated and communicated various options for future control actions and these actions were then coordinated based on the solution of a central optimization problem. However, during communication loss, the system acted like a decentralized controller. Heidarinejad *et al.*, 2011 developed a scheme to ensure stability with communication loss by assuming zero control actions for other subsystems in case of communication loss. Sun and El-Farra, 2008,

proposed a model-based control method based on a decentralized approach and low communication requirements. In this scheme subsystems would interact with each other at predetermined time instants, and for remaining time instants a nominal model is used as state estimator thus essentially converting the strategy into an open loop one.

The goal of the current work is to address the issue of communication loss within Robust DMPC by explicitly accounting for model plant mismatch. When communication is lost between subsystems, the new algorithm computes state bounds to account for model errors and includes these in the formulation of Linear Matrix Inequalities (LMI) that are used to calculate an optimal state feedback predictive control law. When there is a prolonged loss of communication the state bounds calculated at a previous time instant are used as estimates to recursively determine new states' bounds in the presence of model errors. Then, using the bounds on states' estimates corresponding bounds on closed loop performance are minimized based on the worst performing plant. A brief description of the Robust DMPC developed by Al-Gherwi *et al.*, 2011, that was used as the basis of the current approach, is presented in Section 5.2.1, followed by the development of a robust observer to be used in the presence of loss of communication (Section 5.2.2) and a discussion on robust stability (Section 5.2.4) of the proposed strategy. Section 5.3 presents case studies to illustrate the use of algorithm involving a high purity distillation column example (Skogestad *et al.*, 1988) with a high condition number and a reactor separator process (Liu *et al.*, 2009), in the presence of intermittent communication losses. Conclusions are presented in Section 5.4.

Al-Gherwi *et al.*, 2011 developed robust DMPC in which the plant models' parameters of each subsystem lie within a polytopic model and the control action was based on the minimization of a robust performance bound where this latter minimization step is conducted iteratively for every subsystem where the subsystems exchange state information at the beginning of all iterations. To address the issue of communication loss, the new algorithm summarized in this chapter computes state bounds in the presence of model errors and includes these in the formulation of Linear Matrix Inequalities (LMI) that are used to calculate an optimal state feedback predictive control law. Then, while communication is absent, state bounds at previous time instant are used as estimates to recursively determine

new states' bounds, in the presence of model errors. Then, based on these states' bounds, a robust performance bound which corresponds to the worst performing plant is minimized.

5.2 Definitions and Methodology

5.2.1 Robust DMPC Algorithm (Al-Gherwi *et al.*, 2011)

The process is represented by a linear time varying (LTV) model of the form,

$$\boldsymbol{x(k+1)} = \boldsymbol{A(k)x(k)} + \boldsymbol{B(k)u(k)} \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}, \boldsymbol{u} \in \mathbb{R}^{\boldsymbol{m}}$, are the process states and inputs respectively. It is assumed that the time varying behaviour of the process can be effectively described by representing the state matrices by a family of time invariants plants as per the following convex hull:

$$[\boldsymbol{A(k)B(k)}] = \sum_{\boldsymbol{l=1}}^{\boldsymbol{L}} \beta_l \left[\boldsymbol{A^l B^l}\right] \; ; \; \sum_{l=1}^{L} \beta_l = 1; \beta_l \geq 0 \tag{5.2}$$

where each vertex *l* corresponds to a linear model identified from data or obtained from linearization around different operating conditions. This description implies that at any moment the plant can be modelled by any convex combination of these *l* vertices or models. In the original algorithm of Al-Gherwi *et al.*, 2011 it was assumed that all the states are known to all subsystems either through measurements or through estimation thus ignoring the possibility of communication loss.

The system, states and inputs, can be divided into *N* subsystems each represented by following equation

$$\boldsymbol{x_i(k+1)} = \boldsymbol{A_i(k)x_i(k)} + \boldsymbol{B_i(k)u_i(k)} + \sum_{\substack{\boldsymbol{j=1} \\ \boldsymbol{j \neq i}}}^{\boldsymbol{N}} \boldsymbol{B_j(k)u_j(k)} \tag{5.3}$$

where, $\boldsymbol{x_i}$ and $\boldsymbol{u_i}$ include all the states and inputs local to subsystem *i*, and also states and inputs of other subsystems *j*, that affect subsystem *i*, which values are communicated between the subsystems. Hence, $\boldsymbol{x_i} = [\boldsymbol{x'_{11}}, \dots, \boldsymbol{x'_{ii}}, \dots, \boldsymbol{x'_{NN}}]'$, where $\boldsymbol{x_{ii}}$ are states locally

measured within subsystem *i*. Similar to (5.2), the model of each subsystem is given by a polytope defined as follows:

$$\left[\boldsymbol{A_i(k)B_i(k)}..\boldsymbol{B_j(k)}..\right] = \sum_{\boldsymbol{l=1}}^{\boldsymbol{L}} \boldsymbol{\beta_l}\left[\boldsymbol{A_i^{(l)}B_i^{(l)}} .. \ \boldsymbol{B_j^l} ..\right] \ \forall \boldsymbol{j} \in \{\boldsymbol{1}, \ldots \boldsymbol{N}\}, \boldsymbol{j} \neq \boldsymbol{i} \qquad 5.4$$

Kothare et al., 1996 proposed the minimization of a robust performance objective for a centralized formulation where the plant was represented by a polytope. Al-Gherwi *et al*., 2011 extended that formulation by simultaneously minimizing robust performance objectives in a distributed fashion for each of the *N* subsystems as per the following constrained min-max problem:

$$\min_{\boldsymbol{u_i(k+n|k)}} \max_{[\boldsymbol{A_i(k+n)B_i(k+n)B_j(k+n)}],n\geq 0} J_i(k) \qquad 5.5$$

$$\boldsymbol{s.t.} \qquad |\boldsymbol{u_i(k+n|k)}| \leq \boldsymbol{u_i^{max}}, \boldsymbol{n} \geq \boldsymbol{0}$$

where $J_i(k)$ is local cost function for each subsystem defined as

$$J_i(k) = \sum_{\mathrm{n=0}}^{\mathrm{n=\infty}} \Big\{ \boldsymbol{x_i'(k+n|k)S_i x_i(k+n|k)} + \boldsymbol{u_i'(k+n|k)R_i u_i(k+n|k)} + \sum_{\substack{i=1 \\ i\neq j}}^{N} \boldsymbol{u_j^{\bullet'}(k+n|k)R_j u_j^{\bullet}(k+n|k)} \Big\} \qquad 5.6$$

where $\boldsymbol{S_i} > \boldsymbol{0}, \boldsymbol{R_i} > \boldsymbol{0}, \boldsymbol{R_j} > \boldsymbol{0}$ and $\boldsymbol{u_j^{\bullet}}$ is the control action calculated for subsystems j in the previous iteration and remains constant within the current iteration. The local objective function $\boldsymbol{J_i}$, can be formulated to consider different objectives: i- a global objective function, referred to as cooperative control, that corresponds to the case where $\boldsymbol{S_i, R_i, R_j} > \boldsymbol{0}$ and these matrices are all diagonal for each subsystem or ii- a strictly local objective function, e.g. as needed to achieve a Nash equilibrium, that corresponds to the case of $\boldsymbol{R_j} = \boldsymbol{0}$.

The problem of finding the control action $\boldsymbol{u_i} = \boldsymbol{F_i x_i}$ was then formulated as finding a control law $\boldsymbol{F_i}$that is determined by simultaneously minimizing for all subsystems a robust

performance criteria, $\boldsymbol{\gamma_i}, \boldsymbol{i} \in \{\mathbf{1}, \dots, \boldsymbol{N}\}$ subject to a set of LMI's constraints corresponding to stability and input constraints for the polytopic model in (5.4):

$$\min_{\gamma_i, \boldsymbol{Q_i}, \boldsymbol{Y_i}} \gamma_i \qquad \textbf{5.7}$$

$$s.t. \begin{bmatrix} 1 & \boldsymbol{x_i'(k)} \\ \boldsymbol{x_i(k)} & \boldsymbol{Q_i} \end{bmatrix} \geq 0 \qquad \textbf{5.7a}$$

$$\begin{bmatrix} \boldsymbol{Q_i} & \boldsymbol{Q_i \tilde{A}_i'^{(l)} + Y_i' B_i'^{(l)}} & \boldsymbol{Q_i \tilde{S}_i^{\frac{1}{2}}} & \boldsymbol{Q_i' R_i^{\frac{1}{2}}} \\ * & \boldsymbol{Q_i} & \mathbf{0} & \mathbf{0} \\ * & * & \boldsymbol{\gamma_i I} & \mathbf{0} \\ * & * & * & \boldsymbol{\gamma_i I} \end{bmatrix} \geq \mathbf{0} \;\; \forall \boldsymbol{l} \in \{\mathbf{1}, \dots, \boldsymbol{L}\} \qquad \textbf{5.7b}$$

$$\begin{bmatrix} \boldsymbol{(u_i^{max})^2 I} & \boldsymbol{Y_i} \\ \boldsymbol{Y_i'} & \boldsymbol{Q_i} \end{bmatrix} \geq \mathbf{0} \qquad \textbf{5.7c}$$

where $\boldsymbol{F_i = Y_i' Q_i^{-1}}$, and $\boldsymbol{Q_i = \gamma_i P_i^{-1}}$, and the matrix $\boldsymbol{P_i}$ (is p.d.) is used to bound the cost function of each subsystem according to $J_i(k) \leq \boldsymbol{x_i' P_i x_i} = V_i(k)$. The control laws $\boldsymbol{F_i's}$, calculated for each subsystem from the minimization problem above, are then exchanged among the subsystems at each iteration of a Jacobi iterative solution scheme with relaxation until the calculations converge for all the subsystems. The exchange of information among the subsystems is conducted at each time step. Additional details on this iterative algorithm are given in Al-Gherwi *et al.*, 2011.

5.2.2 Loss of Communication

Since the above methodology assumes perfect communication for all iterations at each time step, to address the effect of communication loss it is necessary to modify the above control scheme. The basic idea proposed in this study is that each subsystem is equipped with an observer, based on the closed loop model, which is used to recursively provide bounds on the states while communication is absent. Thus, during periods of communication loss, the robust observer of each subsystem computes bounds for each of the plant states. For each plant state, a bound vector, $\boldsymbol{x_{b,i}(k)} = \left[x_{i,l}(k)\; x_{i,h}(k)\right]', \forall i \in [1, \dots, n]$ is defined where $x_{i,l}(k)$ and $x_{i,h}(k)$ are lower and higher bounds on i^{th} plant state ($x_{p,i}$) respectively and are

based on state bounds calculated in the previous time interval according to the following constrained optimization problems:

$$\boldsymbol{x_{i,l}(k+1)} = \underset{\boldsymbol{A(k),B(k),x_{b,1}(k),\ldots,x_{b,n}(k)}}{\boldsymbol{min}} \boldsymbol{x_{p,i}(k+1)} \tag{5.8}$$

$$\boldsymbol{x_{i,h}(k+1)} = \underset{\boldsymbol{A(k),B(k),x_{b,1}(k),\ldots,x_{b,n}(k)}}{\boldsymbol{max}} \boldsymbol{x_{p,i}(k+1)} \;\; \forall \boldsymbol{i} \in [\boldsymbol{1},\ldots,\boldsymbol{n}]$$

$$s.t. \;\; \boldsymbol{x(k+1)} = (\boldsymbol{A(k)} + \boldsymbol{B(k)F})\boldsymbol{x(k)}$$

$$[\boldsymbol{A(k)B(k)}] = \sum_{\boldsymbol{l=1}}^{\boldsymbol{L}} \beta_l \left[\boldsymbol{A^l B^l}\right] \; ; \; \sum_{l=1}^{L} \beta_l = 1; \beta_l \geq 0$$

$$x_{i,l}(k) \leq x_{p,i}(k) \leq x_{i,h}(k)$$

Where, the decision variables are the bounds at the previous interval $\boldsymbol{x_{b,i}(k)}$ and the elements of the system matrices within the polytopic model (5.2) which result in lower or upper values of each state when solving (**5.8**) above. In the time interval occurring immediately following communication loss, the last available measurement of the state $\boldsymbol{x(k)}$ is used for computing new bounds $\boldsymbol{x_{b,i}(k+1)}$. The optimisation problems in (**5.8**) have a bilinear cost with respect to the decision variables according to the first constraint in (**5.8**), i.e. closed loop model, that involves products between decision variables and thus the problem becomes NP-hard. Bilinear solver algorithm based on "mountain climbing" methods have been proposed (Konno, 1976) or, alternatively, global solvers can be used. In the current work, the DMPC problem is solved using the LMI toolbox of MATLAB and the bilinear solver of YALMIP (Löfberg, 2004) is used to solve (5.8).

The calculated state bounds, $\boldsymbol{x_{b,i}(k)}, \forall i \in [1,\ldots,n]$, are then included as additional LMI constraints for computing the controllers $\boldsymbol{F_i}$'s according to (5.7). The additional LMI's corresponding to the states' bounds are kept fixed during all the iterations at a given time step. It is important to note that the additional LMI's included in the optimization problem (5.7) solved within each subsystem are not only based on bounds of local subsystem states $\boldsymbol{x_i}$, but includes the bounds on all the plant states $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ and since each state has a lower and an

upper bound, the number of additional LMI constraints becomes, 2^n-1. Thus, instead of a single LMI in (5.7a), 2^n LMIs are used as follows:

$$\begin{bmatrix} 1 & \boldsymbol{x_p^j(k)'} \\ \boldsymbol{x_p^j(k)} & \boldsymbol{Q_i} \end{bmatrix} \geq 0 \; \forall j \in [1, \dots, 2^n], \boldsymbol{x_p^j(k)} \in \mathbb{R}^n \qquad 5.9$$

where, $\boldsymbol{x_p^j(k)}$ is a total of 2^n different vectors corresponding to all possible combinations of the lower and upper bounds' values contained in the vectors $\boldsymbol{x_{b,i}(k)}, \forall i \in [1, \dots, n]$.

The key idea behind the combined robust controllers and robust observers proposed here is that since the states' values used by a robust observer in each subsystem *i* are initialized with the same plant measurements when communication is lost, the state bounds determined by the observers during periods of communication loss will be the same across all subsystems. Furthermore, since during periods of communication loss all subsystems use the same set of LMI's with the same bounds' values on all the states for determining control laws $\boldsymbol{F_i}, i \in \{1, \dots, N\}$ all control laws are identical to each other despite the loss in communication.

To account for the limited states' measurements available to each subsystem during communication loss, the control actions $\boldsymbol{u_i}$ for each subsystem *i* are based on the measured local states,$\boldsymbol{x_{ii}}$, and the means of the state bounds for the states of the other subsystems, $\boldsymbol{x_{ij}}, \boldsymbol{j} \in \{\boldsymbol{1}, \dots, \boldsymbol{N}\}, \boldsymbol{j} \neq \boldsymbol{i}$ that are unavailable due to communication loss as follows:

$$\boldsymbol{u_i(k) = F_{ii} x_{ii}(k) + \sum_{\substack{i=1 \\ i \neq j}}^{N} F_{ij} \left(x_{ij,l}(k) + x_{ij,h}(k) \right) / 2} \qquad 5.10$$

Where, $\boldsymbol{F_{ii}}$ and $\boldsymbol{F_{ij}}$ are defined as the sub-matrices of $\boldsymbol{F_i}$'s that relate the control actions of the local subsystem *i* to the local states $\boldsymbol{x_{ii}}$ and the other states $\boldsymbol{x_{ij}}$ respectively.

5.2.3 Summary of the Robust DMPC Algorithm with loss of communication

The robust DMPC algorithm when communication is lost between subsystems is summarized as follows:

Algorithm 1 (RDMPC with loss of communication)

Step0 (initialization)*: at control interval k=0 set* $\boldsymbol{F_i = 0}$

Step1 (updating)

if (beginning of loss of communication)

at the beginning of control interval (k), if there is no communication then solve all 2n optimization problems to determine $\boldsymbol{x_{b,i}(k)}, \forall i \in [1, \dots, n]$,(5.8)

else

at the beginning of control interval (k) all the controllers exchange their local states measurements and initial estimates F_i*'s via communication,*

end if

set iteration t = 0 and $\boldsymbol{F_i = F_i^{(0)}}$.

Step2 (iterations)

while $t \leq t_{max}$

Solve all N LMI problems in parallel to obtain the minimizers $\boldsymbol{Y_i^{t+1}, Q_i^{t+1}}$ *used to estimate the feedback solutions* $\boldsymbol{F_i^{t+1} = Y_i'^{(t+1)} Q_i^{-1(t+1)}}$. *If problem in a particular iteration is infeasible set,* $\boldsymbol{F_i^{(t)} = F_i^{(t-1)}}$. *Check the convergence for a specified error tolerance* ε_i *for all the controllers*

if $\left\| \boldsymbol{F_i^{(t+1)} - F_i^{(t)}} \right\| \leq \varepsilon_i \; \forall i \in \{1, \dots, N\}$

break

end if

Exchange the solutions $\boldsymbol{F_i}$*'s and set t = t + 1*

end while

Step3 (implementation)

if (loss of communication)

apply the control actions $\boldsymbol{u_i} = \boldsymbol{F_{ii}x_{ii}} + \sum_{\substack{i=1 \\ i\neq j}}^{N} \frac{\boldsymbol{F_{ij}}(\boldsymbol{x_{ij,l}}+\boldsymbol{x_{ij,h}})}{2}$ *to the corresponding subsystems, increase the control interval k = k + 1, return to step1 and repeat the procedure.*

else

apply the control actions $\boldsymbol{u_i} = \boldsymbol{F_i x_i}$ *to the corresponding subsystems, increase the control interval k = k + 1, return to step1 and repeat the procedure.*

end if

5.2.4 Convergence and Robust Stability Analysis of Robust-DMPC Algorthm with loss of communication

Lemma 1 (Al-Gherwi *et al.*, 2011). For a cooperative control objective, defined in section 5.2, each one of the *N* convex problems defined in *Algorithm1* will converge to the same solution which is the solution of the centralized problem, i.e $\gamma_1 = \cdots = \gamma_i = \cdots = \gamma_N = \gamma$, where γ, is the performance upper bound of centralized MPC (see Al-Gherwi *et al.*, 2011).

In the absence of communication at each time step, identical controllers $\boldsymbol{F_i}$*'s* are computed within each subsystem from the minimization of robust performance criterion in each subsystem, i.e. $\gamma_1 = \cdots = \gamma_i = \cdots = \gamma_N = \gamma$. However, in this case γ is a performance upper bound calculated based on bounds rather than on the actual state measurements.

For the purpose of proving the robust stability of the proposed algorithm the following definitions are needed:

Definition 1 (*Invariant Set for Quadratic Stability*) Boyd, 1994. The set $\varepsilon = \{\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}} | \boldsymbol{x'Q^{-1}x} \leq 1\}$ is said to be an invariant set for $\boldsymbol{x(k+1)} = \boldsymbol{\Phi(k)x(k)}$ where $\boldsymbol{\Phi(k)} = (\boldsymbol{A(k)} + \sum_{i=1}^{N} \boldsymbol{B_i(k)F_i(k)})$, iff $\boldsymbol{Q^{-1}}$ satisfies $\boldsymbol{Q^{-1}} - \boldsymbol{\Phi'^{(l)}Q^{-1}\Phi^{(l)}} \geq 0, l \in \{1, \ldots, L\}$. As a result if $\boldsymbol{x(k)} \in \varepsilon$, then $\boldsymbol{x(k+1)} \in \varepsilon$.

Definition 2 (*Intersection of Invariant sets*). If the sets $\varepsilon_i = \{\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}} | \boldsymbol{x}' \boldsymbol{Q_i^{-1}} \boldsymbol{x} \leq 1\} \forall i \in \{1, \ldots, N\}$, exist then there is a set $\varepsilon = \bigcap_{i=1}^{N} \varepsilon_i$ defined as $\varepsilon = \{\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}} | \boldsymbol{x}' \boldsymbol{Q^{-1}} \boldsymbol{x} \leq 1\}$ where $0 < \boldsymbol{Q^{-1}} \leq \sum_{i=1}^{N} \tau_i \boldsymbol{Q_i^{-1}}, \sum_{i=1}^{N} \tau_i = 1, 0 \leq \tau_i \leq 1$.

Then, the robust stability of *Algorithm1* is given in Theorem 1.

Theorem 1. *At sampling time k and any iteration t > 0, the state feedback solutions* $\boldsymbol{F_i^{(t)}}(\boldsymbol{k}) = \boldsymbol{Y_i'^{(t)}}(\boldsymbol{k}) \boldsymbol{Q_i^{-1(t)}}, i \in \{1, \ldots, N\}$, *obtained from Algorithm 1, robustly stabilize the closed loop system* $\boldsymbol{x}(\boldsymbol{k}+\boldsymbol{1}) = \boldsymbol{\Phi}(\boldsymbol{k}) \boldsymbol{x}(\boldsymbol{k})$ *where* $\boldsymbol{A}(\boldsymbol{k})$ *and* $\boldsymbol{B}(\boldsymbol{k})$ *belong to the polytopic description defined in (2).*

Proof. In the presence of communication the stability proof provided in Al-Gherwi *et al.*, 2011 applies. When communication is lost if the problem posed in (5.7) is feasible and all the process disturbances can be bounded by the parametric uncertainties considered in modeling, since $\boldsymbol{Q_i^{-1}} > 0$ and (5.9) is satisfied i.e. $\boldsymbol{x_p^j}(\boldsymbol{k})' \boldsymbol{Q_i^{-1}} \boldsymbol{x_p^j}(\boldsymbol{k}) \leq 1, \forall j \in \{1, \ldots, 2^n\}$, then due to convexity and the bounds of the states $\boldsymbol{x}' \boldsymbol{Q_i^{-1}} \boldsymbol{x} \leq \boldsymbol{x_p^j}(\boldsymbol{k})' \boldsymbol{Q_i^{-1}} \boldsymbol{x_p^j}(\boldsymbol{k}) \leq 1, \forall j \in \{1, \ldots, 2^n\}$, and the condition of definition 2 (*intersection of invariant sets*) is satisfied. Thus, the robust stability criterion based on the intersection of invariant sets given by definition 2 and used in the robust stability proof of Al-Gherwi *et al.*, 2011 is also satisfied here. Accordingly, $\boldsymbol{Q^{-1}} - \boldsymbol{\Phi}^{(l)'} \boldsymbol{Q^{-1}} \boldsymbol{\Phi}^{(l)} \geq 0 \; \forall l \in \{1, \ldots, N\}$, which satisfies the conditions of definition 1 and thus $\boldsymbol{x}(\boldsymbol{k}+\boldsymbol{n}), n > 0$ belong to the invariant set $\varepsilon = \{\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}} | \boldsymbol{x}' \boldsymbol{Q^{-1}} \boldsymbol{x} \leq 1\}$.

One key aspect to note is that although the controller action is implemented using the mean of the bounds, as determined by the robust observer, the resulting controller is stable since the actual state of the other subsystem is bounded within its respective upper and lower bound; and both lower/upper bound has been included in the formulation of robust controller as a state constraint.

5.3 Case Studies

5.3.1 Case Study 1

To illustrate the proposed algorithm in the absence of communication between subsystems, a high purity distillation column example from Skogestad *et al.*, 1988 was studied. The system consists of two inputs, reflux and boil-up ratio, and two outputs, composition of top and bottom products. Due to the high condition number this process has been often used to study the effects of uncertainty on closed loop control stability and performance. The state space model consists of 2 states, 2 inputs and 2 outputs, and the actual plant is being represented by a polytope with 4 vertices. $\boldsymbol{B}^{(1),(2),(3),(4)}$ represents the 4 vertices of convex hull within which the plant lies.

$$A^{(1)} = A^{(2,3,4)} \begin{bmatrix} 0.9231 & 0 \\ 0 & 0.9231 \end{bmatrix}, \tag{5.11}$$

$$B^{plant} = \begin{bmatrix} 0.1013 & 0.0332 \\ 0.1248 & 0.0421 \end{bmatrix} * \begin{bmatrix} 1 \pm \Delta_1 & 0 \\ 0 & 1 \pm \Delta_2 \end{bmatrix} \tag{5.11a}$$

Where, $\Delta = [\Delta_1, \Delta_2], \Delta_1 \in [-0.75, 0.75], \Delta_2 \in [-0.8, 1.4]$

$$B^1 = \begin{bmatrix} 0.1755 & 0.0066 \\ 0.2163 & 0.0084 \end{bmatrix}, \quad B^2 = \begin{bmatrix} 0.1755 & 0.0797 \\ 0.2163 & 0.1011 \end{bmatrix} \tag{5.11b}$$

$$B^3 = \begin{bmatrix} 0.0270 & 0.0066 \\ 0.0333 & 0.0084 \end{bmatrix}, \quad B^4 = \begin{bmatrix} 0.0270 & 0.0797 \\ 0.0333 & 0.1011 \end{bmatrix}$$

Both inputs were assumed to be constrained by, $|u_1(k)| < 4.5,\ |u_2(k)| < 4.0$. The system was subdivided into two subsystems as $u_1 - y_1$ and $u_2 - y_2$. The importance of considering robustness of a centralized controller to model errors for this particular process has been demonstrated in previous studies Skogestad *et al.*, 1988 and it is not illustrated here.

Instead, since the key point of the proposed algorithm is to provide for robustness in the presence of communication loss, it is relevant to compare a robust controller combined with a robust observer to a robust controller that is combined with a nominal observer where the latter is solely based on a nominal model thus ignoring model errors. Accordingly, these two controllers will be referred heretofore by the observer type that is used, i.e. robust observer

versus nominal observer respectively, where the robust observer based configuration is the one proposed in the current study (equations 5.8-5.10 above).

To compare the performance of these controllers, two cost functions were used first one related to the overall control effort, $J_u = \sum_{j=1}^{N_s}\sum_{i=1}^{N}(u_i'(j)u_i(j))$, and the second one related to the output variables, $J_y = \sum_{j=1}^{N_s}\sum_{i=1}^{N}(y_i'(j)S_i y(j))$, where N_S is the simulation time.

Nine different plants realizations were chosen by simulating the plant model with different $\boldsymbol{B}$ matrices included within the uncertainty values used for the robust controller given by (11) above: Δ=[-0.7,0.7], [-0.7,0], [-0.7,0.4], [0,0.7], [0,0], [0,-0.4], [0.7,0.7], [0.7,0], [0.7,-0.4]. It should be emphasized that for the simulations the robust controller is always based on the same uncertainty bounds values given in (5.11), whereas the model used to emulate the actual process is varied according to the nine combinations of $\Delta's$ given above to simulate the operation of the robust controller around different operating conditions. To test the effect of duration of communication loss, this loss was assumed to be periodic as shown in Figure 5.1 where presence or loss of communication between the subsystems is

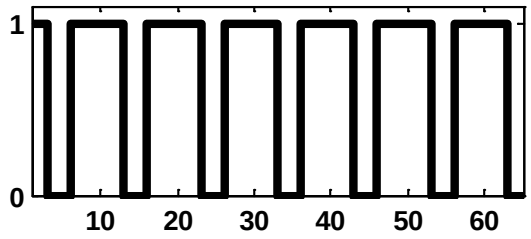


Figure 5.1 Communication Loss Profile, Lost = 0

indicated by values of 1 or 0 respectively. The controller parameters used in simulation for comparing robust control and observer with the controller with the non-robust observer are: $S_1 = 1, S_2 = 1, R_1 = 2, R_2 = 2,\ \varepsilon_1\ and\ \varepsilon_2 = 10^{-2}$. A set-point change of [-15,-20] is conducted for the plant outputs, y_1 and y_2, respectively. The results for these two configurations were compared for a set-point change based on the two cost functions J_u and J_y given above.

Table 5.1 presents results for the set-point change with varying periods of communication loss and various plant realizations. As shown in Table 5.1 when the loss of communication period T is 6 or 5, and for any of the plants' uncertainty realizations considered in the simulation, the control actions' cost $J_{u,non-robust}$ is 6.0-7.0 times more than that of $J_{u,robust}$. For $T = 3$, both non-Robust and Robust perform similarly for 6 out of 9 cases and for 3 out of 9 cases $J_{u,non-robust}$ is again 1.7-1.8 times of $J_{u,robust}$. Thus, as the period of communication loss increases, the difference in performance between the two controllers is very significant.

Also, the differences in output performance, given by $J_{y,}$ between the two controllers changes significantly with respect to the particular realization of the plant and it was noted that the ratio of the two cost values defined for comparing the performance remain insensitive to changes in Δ_2, but significantly increase with respect to changes in Δ_1. Figure 5.2 compares the responses (plant $\Delta_1, \Delta_2 = 0.7$) for communication loss periods of $T = 3$ and 5 respectively. For a loss period of $T = 3$, the controlled variables do not show a significantly different behavior ($J_{y,non\text{-}robust}/J_{y,robust} \sim 1.1$) but the plant inputs show a significantly different profile. The robust controller with the non-robust observer exhibits a highly oscillatory behavior as communication is lost and established back again. For communication loss period of 5, both plant inputs and outputs, have drastically different closed loop performance for the two controllers resulting in significant improvement of the performance when using the robust controller with the robust observer. It should not be surprising that for $\Delta = [0.0, 0.0]$, the robust observer based controller performs better than the nominal counterpart even though the nominal observer model matches the plant. The reason for this is that even the robust controller with the nominal observer is tuned for the worst case model, which is generally different from the nominal model used by the nominal observer.

Table 5.1: Robust Observer vs Nominal Observer

Plant	$\Delta = [-0.7, 0.7]$		$\Delta = [-0.7, 0.0]$		$\Delta = [-0.7, -0.4]$		$\Delta = [0.0, 0.7]$		$\Delta = [0.0, 0.0]$	
Loss	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust
Period	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost
3	245	235	244	236	238	231	198	193	189	183
	3209	3200	3188	3186	3195	3196	2562	2573	2555	2567
4	282	847	264	791	251	768	239	627	209	566
	3285	3767	3228	3550	3222	3448	2669	3194	2623	2997
5	254	1896	229	1594	220	1469	203	1588	180	1259
	3254	4538	3237	3967	3239	3711	2685	4135	2658	3495
6	245	4351	221	2970	209	2506	206	5010	182	2982
	3311	6253	3280	4618	3273	4082	2680	6982	2645	4749

Plant	$\Delta = [0.0, -0.4]$		$\Delta = [0.7, 0.7]$		$\Delta = [0.7, 0.0]$		$\Delta = [0.7, -0.4]$	
Loss	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust
Period	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost
3	184	177	153	269	142	259	140	255
	2556	2569	2052	2396	2026	2334	2022	2301
4	197	541	140	784	130	659	127	606
	2609	2909	2092	3337	2083	2965	2080	2804
5	170	1147	123	2105	122	1575	118	1316
	2651	3342	2032	4669	2023	3871	2039	3400
6	172	2387	126	9013	124	4793	121	3511
	2633	4071	2001	10623	1992	6526	2007	5150

Table 5.2: Robust Observer vs Non-robust for Nash Scheme

Plant	$\Delta = [-0.7, 0.7]$		$\Delta = [-0.7, 0.0]$		$\Delta = [-0.7, -0.4]$		$\Delta = [0.0, 0.7]$		$\Delta = [0.0, 0.0]$	
Loss	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust
Period	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost
3	153 3485	366 6316	135 3446	346 6205	126 3449	336 6121	145 3173	308 6117	129 3187	296 6000
5	106 3393	2080 7950	95 3376	1556 7017	91 3400	1336 6634	94 3001	2484 8760	87 2999	1812 7513

Plant	$\Delta = [0.0, -0.4]$		$\Delta = [0.7, 0.7]$		$\Delta = [0.7, 0.0]$		$\Delta = [0.7, -0.4]$	
Loss	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust	Robust	Non-Robust
Period	Cost	Cost	Cost	Cost	Cost	Cost	Cost	Cost
3	119 3183	284 5896	146 2937	275 6001	130 2871	328 5805	120 2843	314 5759
5	81 3020	1496 6945	87 2683	3541 10707	82 2671	2433 8546	77 2668	1952 7752

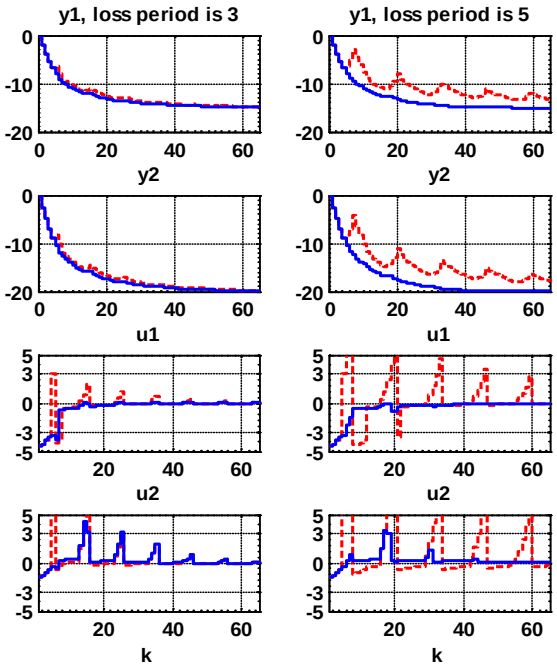


Figure 5.2 Comparison of Robust Observer (Solid) vs Non-Robust (Dashed)

Since in many industrial implementations of distributed MPC, local control objectives are enforced for each subsystem, comparison of input and output response was carried out using a Nash equilibrium based scheme which is based on local control objectives that involve local variables only, i.e. $\boldsymbol{R_j} = 0$, and $\boldsymbol{S_j}$ is a diagonal weighting matrix where all the elements are set to zero except for those elements which multiply the states which are measured locally in each subsystem. Controller parameters used for this study are $S_1 = 1, S_2 = 1, R_1 = 0.7, R_2 = 0.7, \alpha = 1,\ \varepsilon_1\ and\ \varepsilon_2 = 10^{-2}$.

Table 5.2 presents $J_{u,robust}$, $J_{y,robust}$ and $J_{u,non-robust}$, $J_{y,non-robust}$ for a loss period of $T = 3,5$ and for nine plant realizations. Even for the communication loss period $T = 3$, the ratio of the costs related to the control effort $J_{u,non-robust}/J_{u,robust}$ is between 1.8-2.6. And this difference in performance monotonically grows with increasing loss period. Similarly for $J_{y,non\text{-}robust}/J_{y,robust}$ is between 1.7-2.0 for $T = 3$. Hence for the Nash equilibrium based scheme, where only local objective functions are optimized in each subsystem, the robust controller with the robust observer provides significant improvement over the performance of the robust controller with the nominal observer even for short periods of communication loss.

5.3.2 Case Study 2

To test the practicality of the proposed robust observer based DMPC methodology to larger systems we consider a reactor-separator process, where there are 3 process units, 2 CSTR's followed by a flash separator (see Figure 5.3). A polymerization reaction occurs in both the CSTR's, $A -> B -> C$, where A is the reactant, B is the desired product and C is an undesired side product. Most of the vapors from the flash separator are condensed, recycled and then a small part is purged out. The bottoms from the separator form the product. CSTR1 receives fresh feed, F_{10} comprising of the reactant A, the effluent from CSTR1 then flows into the CSTR2 along with fresh feed F_{20}consisting of the reactant A, and finally the effluent from CSTR2 goes to the flash separator. Each of the process units is equipped with an external heat input (Q_1, Q_2, Q_3) in order to maintain the temperature conditions. This type of process system has been previously used by Liu *et al.*, 2009 with different steady-state conditions to develop a stabilizing distributed MPC algorithm referred to as Lyapunov-based MPC. The entire system consists of 9 states $(x_{A1}, x_{B1}, T_1, x_{A2}, x_{B2}, T_2, x_{A3}, x_{B3}, T_3)$ and 4 inputs (Q_1, Q_2, Q_3, F_{20}).

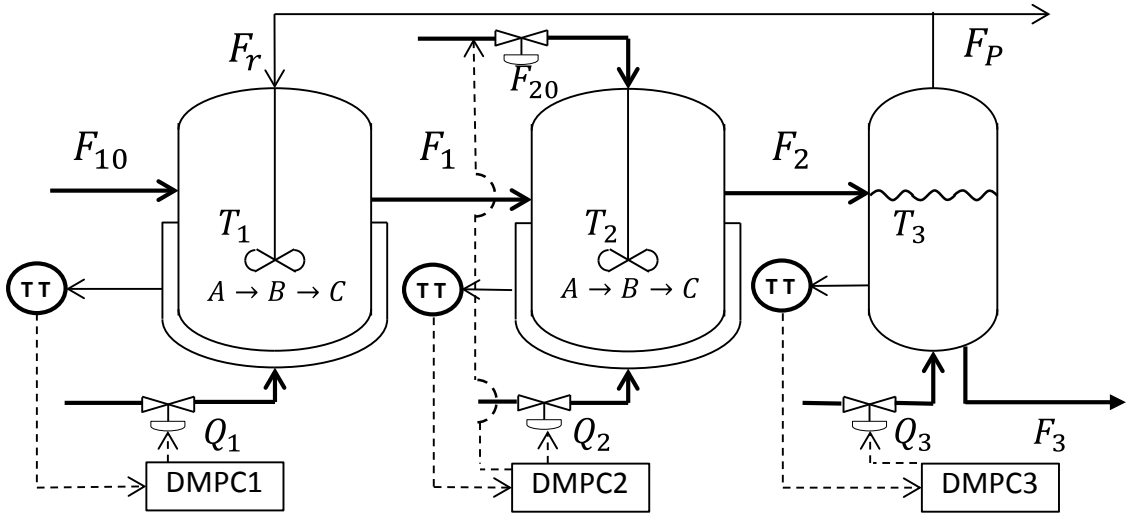


Figure 5.3: DMPC Scheme of Reactor-Separator Case Study[4].

[4] Figure 5.3 has been adapted from Liu, J. *et al*., 2009..

Figure 5.3 shows the distributed control scheme assuming the partition of the process into three subsystems. The input-output pairing of the first subsystem consists of the local states and the heat input to the CSTR1: x_{A1}, x_{B1}, T_1, Q_1; similarly for subsystem 3, the local states and the heat input in the flash separator are the outputs and input respectively: x_{A3}, x_{B3}, T_3, Q_3. Subsystem 2 consists of local states (x_{A1}, x_{B1}, T_1) along with the heat input and the amount of fresh feed supplied to CSTR2 (Q_2, F_{20}) as inputs. This type of system is highly nonlinear and for the purposes of this study is linearized around a steady state condition (provided in Table 5.3), and then discretized in order to derive the LTI model matrices of the entire system[$\boldsymbol{A}, \boldsymbol{B}$]. The robust model is then derived by assuming uncertainty in $\boldsymbol{A}$, and is defined by the convex hull of 4 matrices [$\boldsymbol{A}^{(1),(2),3,(4)}$] and [$\boldsymbol{B}^{(1),(2),3,(4)}$]**.** The values of these matrices have been provided in Appendix B. The structure of the robust model as derived from the nominal model is presented in Eq. 5.12, where $\Delta_1, \Delta_2, \Delta_3$, define the uncertainty in respective subsystems, and the bounds on these uncertainty elements are shown in Eq. 5.13.

$$\boldsymbol{A} = \boldsymbol{A}_{nominal} * \left\{ \boldsymbol{I}_{9x9} + \begin{bmatrix} \Delta_1 \boldsymbol{I}_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & \Delta_2 \boldsymbol{I}_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & \Delta_3 \boldsymbol{I}_{3x3} \end{bmatrix} \right\} \tag{5.12}$$

$$\left[\boldsymbol{B}^{(1),(2),3,(4)}\right] = \boldsymbol{B}^{(1)} = \boldsymbol{B}_{nominal}$$

$$-0.5 \leq \Delta_1 \leq 0.15, -0.5 \leq \Delta_2 \leq 0.3, \Delta_3 = 0 \tag{5.13}$$

Linearization is done by using first-order Taylor Series expansion around steady-state condition, followed by discretization using MATLAB command *c2d* with a time step of 3 minutes. The [$\boldsymbol{A}^{(1),(2),3,(4)}$] and [$\boldsymbol{B}^{(1),(2),3,(4)}$] matrices are determined by using different pairings of, $[\Delta_1, \Delta_2] = \{[-0.5, -0.5], [-0.5, 0.3], [0.15, -0.5], [0.15, 0.3]\}$, respectively.

In order to compare the performance of robust observer based DMPC with the nominal observer based DMPC, the system was disturbed from the steady-state and then the DMPC scheme was investigated as for the previous case study for 3 different communication loss profile, that differ by the duration of the communication loss period as $T = 3,4,5,$ (see Figure

5.4) and for 6 different plant configurations within the convex hull defined by $[\boldsymbol{A}^{(1),(2),3,(4)}]$ and $[\boldsymbol{B}^{(1)}]$**.** The cost function for comparison is defined as the weighted sum of

Table 5.3: Process Parameters and corresponding Steady-State values

Process Variables	Description	Steady-State Values
x_{A1}, x_{A2}, x_{A3}	Mass Fractions of A in vessels 1,2,3	[0.865, 0.881, 0.750]
x_{B1}, x_{B2}, x_{B3}	Mass Fractions of B in vessels 1,2,3	[0.121, 0.117, 0.244]
x_{C1}, x_{C2}, x_{C3}	Mass Fractions of C in vessels 1,2,3	[0.014, 0.002, 0.006]
T_1, T_2, T_3	Temperatures in vessels 1,2,3	[380.2, 376.07, 385.55] (K)
T_{10}, T_{20}	Feed Stream Temperatures to vessels 1,2	$[300, 300]K$
F_1, F_2	Effluent flowrate from vessels 1,2	$[20.08, 25.12]m^3/min$
F_{10}, F_{20}	Steady-state feed stream flowrates to vessels 1, 2	$[5.04, 5.04]m^3/min$
V_1, V_2, V_3	Volumes of vessels 1, 2	$[1.0, 0.5, 1.0]m^3$
E_1, E_2	Activation Energy for the reactions 1,2	$[5x10^4, 6x10^4]KJ/kmol$
k_1, k_2	Pre-exponential values for reactions 1, 2	$[2.77x10^3, 2.5x10^3]s^{-1}$
$\Delta H_1, \Delta H_2$	Heats of reactions for reactions 1, 2	$[6x10^4, 7x10^4]\ KJ/kmol$
$\alpha_A, \alpha_B, \alpha_C$	Relative volatilities of A, B, C	[3.5, 1.0, 0.5]
Q_1, Q_2, Q_3	Heat inputs to vessels 1, 2, 3	$[1.08x10^6, 1.14x10^6]KJ/h,$ $[1.0x10^6]\ KJ/h$
C_p, R, ρ	Heat Capacity, solution density, Gas Constant,	[4.2 $KJ/kg/K$, 1000 $\frac{kg}{m3}$, 8.314 $KJ/kmol/K$]

$J = \sum_{j=1}^{N_s}\sum_{i=1}^{N_y}(y_i'(j)S_i y(j)) + \sum_{j=1}^{N_s}\sum_{i=1}^{N_u}(u_i'(j)R_i u_i(j))$, where in this case study $S = 0.3I_{9x9}$ and $R = 0.6I_{9x9}$. Also for the purposes of this case study, J_{robust}refers to the value of the cost for the robust observer based DMPC and $J_{nominal}$ refers to the cost function value

for the nominal observer based DMPC. The process system of Figure 5.3 has 3 subsystems, with uncertainty in 2 subsystems, resulting in 4 matrices which defines the convex hull however, since the total number of states are 9, while solving Problem 5.7 with augmented state constraints (Eq. 5.9), there are additional $2^9 - 1 = 511$, LMI constraints added. The robust control problem 5.7 is solved locally for each subsystem at every time step during the loss of communication.

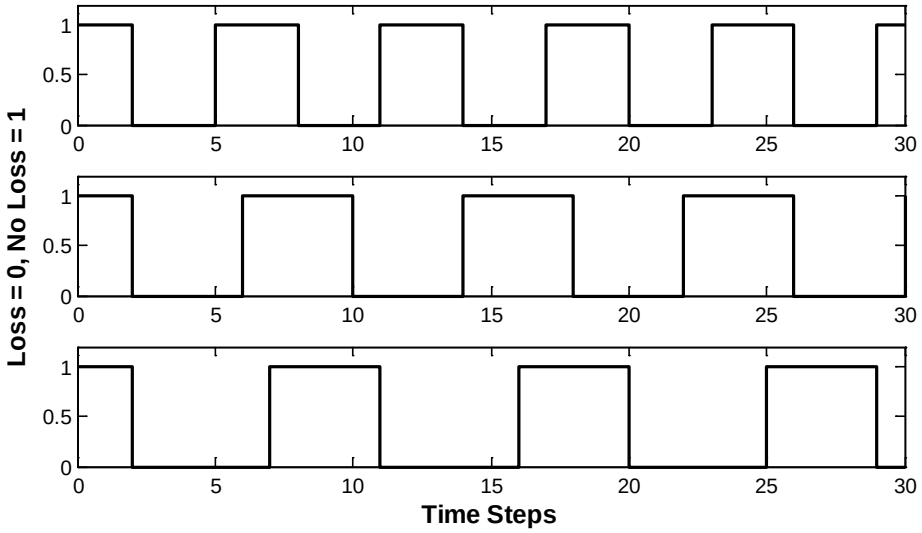


Figure 5.4: Communication Loss Profile, $T = 3, 4, 5$

Table 5.4 compares the value of the cost function for 6 different plants, for 3 different communication loss profiles. Robust observer consistently does better than nominal observer for all the cases considered by ~20%, i.e $J_{robust} > J_{nominal}$. On increasing the communication loss period from $T = 3\ to\ 5$, there is a marginal improvement in $\bar{J}_{robust}$over $\bar{J}_{nominal}$, where $\bar{J}$ represents the cost averaged over all the 6 plants for each of the communication loss period. It can also be observed that as the value of the uncertainty grows, the ratio of $J_{nominal}/J_{robust}$increases, for example for all the communication loss periods considered, the robust observer based DMPC had the greatest positive impact on the plant defined by $\Delta_1, \Delta_2 = [-.4, -0.25]$ and the least impact on the plant defined by $\Delta_1, \Delta_2 = [-0.1, 0.25]$.

Table 5.4: Cost function value for Robust vs. Non-robust Observer in Case Study 2

State wt = 0.3, Input wt = 0.6			J, loss period = 3			J, loss period = 4			J, loss period = 5		
Plant #	Δ1	Δ2	Nominal	Robust		Nominal	Robust		Nominal	Robust	
1	-0.1	0.25	6.99	6.66	1.05	7.38	6.92	1.07	7.41	6.87	1.08
2	-0.1	-0.4	5.29	4.47	1.18	5.83	4.86	1.20	5.92	4.92	1.20
3	-0.4	-0.25	4.25	3.26	1.30	4.84	3.74	1.29	4.95	3.79	1.31
4	-0.4	0.25	4.67	3.72	1.25	5.29	4.18	1.26	5.43	4.25	1.28
5	-0.4	-0.4	4.15	3.16	1.31	4.72	3.64	1.30	4.83	3.68	1.31
6	-0.1	-0.25	5.56	4.79	1.16	6.07	5.16	1.18	6.16	5.20	1.19
					1.21			1.22			1.23

Figure 5.5 and Figure 5.6 compares the controller action and states for nominal and robust observer for the plant defined by $\Delta_1, \Delta_2 = [-0.4, -0.25]$, with communication loss period $T = 5$. The value of all the states and control actions has been normalized. Since the system is not starting from a steady-state condition, all the states and control actions exhibit a sudden jump towards the origin, and then during communication loss they exhibit additional jumps towards the steady-state before finally reaching the steady-state. While the controller is trying to bring back the plant to steady-state, the nominal observer based DMPC algorithm

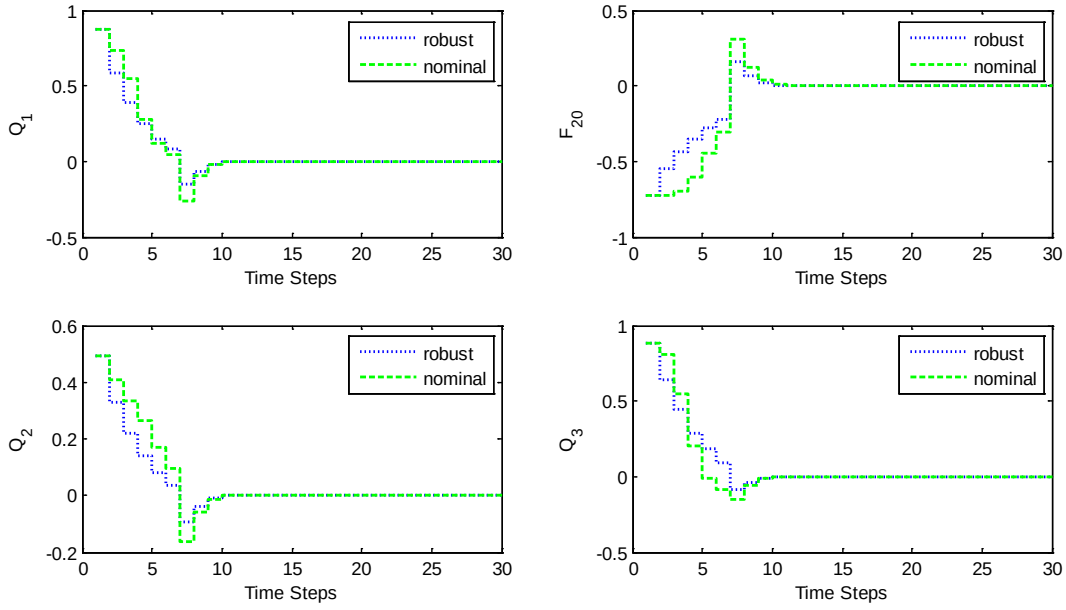


Figure 5.5: Control Actions for Plant $\Delta_1, \Delta_2 = [-0.4, -0.25]$ and loss period = 5

results in a bigger jump away from the steady when compared with the robust observer based algorithm. During steady-state operation both the nominal and robust observer based algorithms results in similar control action. This can be attributed to the fact that during steady-state operation, both the nominal observer and robust observer based DMPC provide very similar solutions in terms of state estimates and state bounds respectively.

5.4 Conclusions

In this work a robust DMPC algorithm supplemented by a robust observer has been proposed to handle communication loss. It is shown that a robust DMPC algorithm can be effectively combined with a robust observer that is formulated by additional LMI constraints. The key idea is that the robust observer is incorporated in each subsystem to determine bounds for all the states, and since the starting point of the state estimates used by robust observer for each subsystem after communication loss is same, it computes identical bounds for all subsystems during periods loss of communication thus resulting in the same controller being calculated for each subsystem. To account for the limited information available to each subsystem during communication loss the control action for each subsystem is based on the local measured states and on the mean values of the bounds of the unmeasured states. The performance of the robust controller combined with the robust observer was shown to be significantly better than that of a robust controller that uses a nominal model based observer for two different case studies.

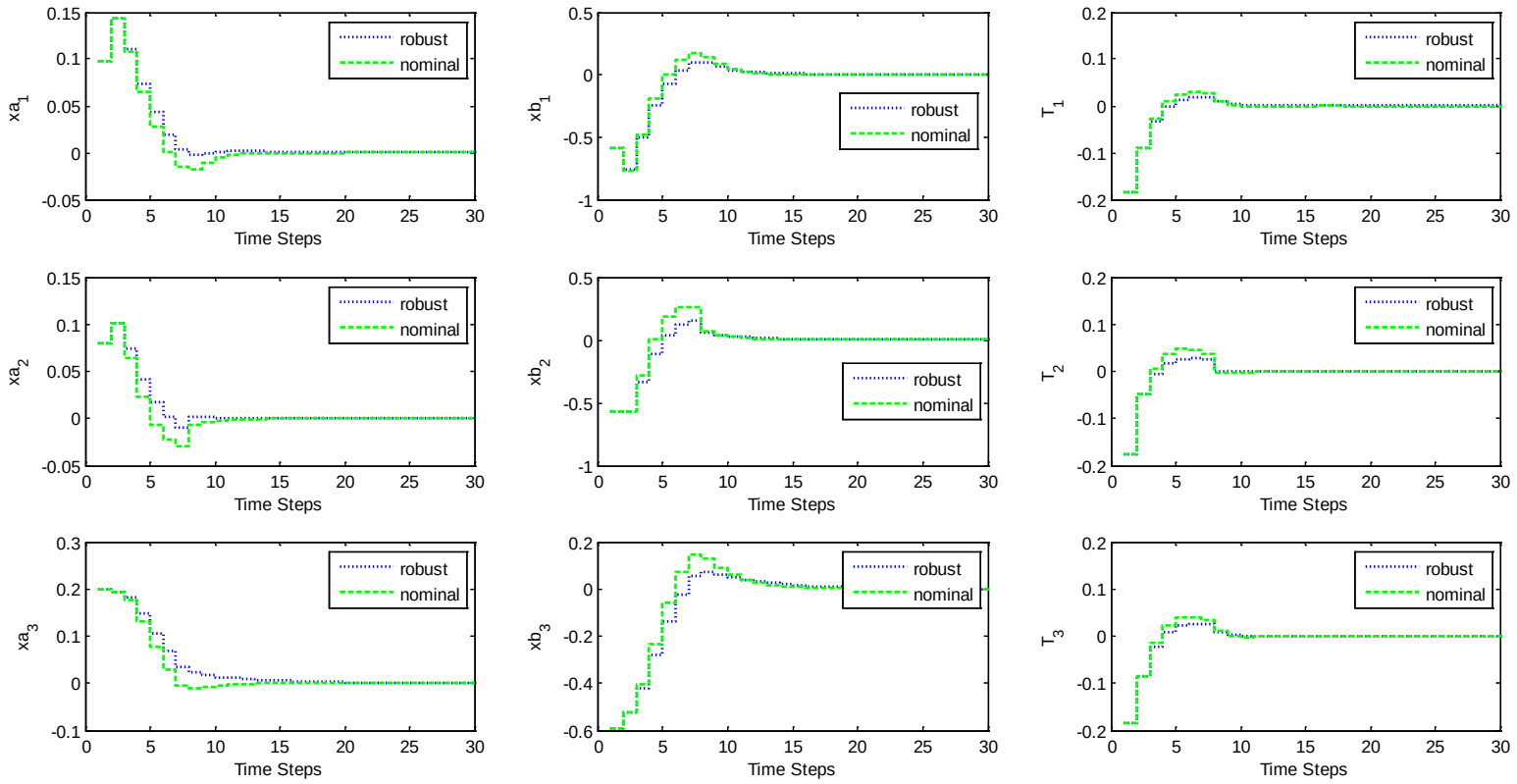


Figure 5.6: System Response to the control inputs for Plant $\boldsymbol{\Delta_1, \Delta_2 = [-0.4, -0.25]}$ and loss period = 5

Chapter 6
Conclusions and Future Work

6.1 PCE-based Robust NMPC

In Chapter 3, a robust controller is formulated using a PCE based model and a min-max optimization. The motivation for this work comes from the conservative nature of previously proposed worst-case type robust controller and from their computational complexity. The use of PCE's and nonlinear empirical models based on Volterra series facilitated the propagation of parametric uncertainty and permitted quick calculation of the $\mathcal{L}2$-norm of the uncertain closed-loop model predictions, thus removing the need for developing a worst-case controller for intermediate predictions. Controller stability is still formulated using a worst-case approach based on SSV test. The reduction in conservatism and computational burden of the robust controller are key contributions of this work. A case study is presented comparing the results of PCE-based RNMPC with an SSV-based RNMPC. The proposed controller was also compared to two nonlinear nominal controllers based on first-principles model and an empirical model using Volterra series that do not consider model error. The PCE-based RNMPC was shown to be clearly superior in all cases when compared in terms of the IAE at different operating conditions. The reduction in computational load of the proposed PCE based RNMPC with the previously SSV-RNMPC was especially significant with executions times of the order of 1 hour for the former versus more than 5 days for the latter. This improvement makes the PCE based RNMPC controller suitable for implementation to processes of larger complexity. The reliance of the proposed algorithm on an empirical nonlinear model is of industrial interest since suitable mechanistic models are often not available or are very difficult to develop.

6.2 PCE Applications in Robust Control and Optimization of batch bioreactor

In Chapter 4 two novel algorithms using PCE are presented for robust optimization and control of a fed-batch reactor modeled using DFBM. The choice of control versus optimization for bioreactor operation mainly depends on the availability of on-line measurements during a batch and on the flexibility of standard operating procedures with

respect to closed-loop control implementation. Assuming that online measurements are available, an online robust control algorithm based on PCE is developed using an economic objective function. In the case study the objective function was chosen as the end-point biomass amount. Since the DFBM that was used is posed as an LP-based model and since the resulting robust control problem also requires an optimization operation, the overall problem is formulated as a bilevel optimization. The inner level optimization is then replaced using a surrogate model between the fluxes to the uncertainty in model parameters using a non-intrusive PCE approach. On the other hand the uncertainty is propagated onto the model predictions along the control prediction horizon using intrusive methods, i.e. by explicitly using the first principle equations of the system. The controller was applied to the process of growth of *E.Coli* based on glucose and acetate and the control actions were calculated based on feedback corrections of glucose and biomass. The robust controller showed better final productivity (biomass level) than the nominal controller by an average of 13% that can be significant in bio-manufacturing. The use of PCE to propagate parametric uncertainty was shown to be computationally very efficient as compared to a Monte Carlo approach. This computational efficiency is expected to be instrumental for the online implementation of the proposed robust algorithm into an actual process.

The offline-robust optimization approach presented in Chapter 4 is based on the minimization of the level of violation of a lower bound on the end point biomass level. This type of objective is important in manufacturing of pharmaceuticals since batches are costly and batch reactor time is extremely critical to supply high demand. A surrogate model of the end-point biomass is developed using a non-intrusive PCE approach which allowed for quick calculation of the variance by analytical formula. The nominal optimization failed to meet the minimum end-point biomass lower limit for 18% of the cases as opposed to 4% violation for robust optimization.

As mentioned above the choice between robust control and robust optimization will depend on the quality of available measurements and on process certification constraints. In the past, pharmaceutical companies have generally avoided on line control in manufacturing operations due to safety concerns and inflexible operating procedures. In these cases, a robust

optimization approach would be preferred. However, recent changes in FDA guidelines have increased the acceptability of on-line control strategies for manufacturing of pharmaceuticals such as the one proposed in this work.

6.3 Robust Observer for Distributed MPC

In previous chapters the topic of computational complexity was tackled by using PCE as a quick mean to propagate uncertainty onto the objective functions and constraints. In Chapter 5 computational complexity is tackled by distributed control. In general, most of the industrial implementations of MPC in the process industries rely on distributed control due to computational and maintenance issues of one single centralized controller for a plant with a large number of inputs and outputs. In Chapter 5 the concept of a robust observer is introduced for linear models which can be used to address the issues of communication loss between subsystems in a distributed control. The proposed robust observer determines upper and lower bounds of all the plant states from the solution of a bilinear optimization problem. The robust observer is then integrated with a robust DMPC algorithm to handle communication loss. It is shown that the robust stability analysis of a robust DMPC algorithm combined with the robust observer can be posed by a system of LMI constraints. The key idea is that each subsystem is equipped with a robust observer to determine bounds for all the states, and since the starting point of the state estimates used by robust observer for each subsystem after communication loss is same, it computes identical bounds for all subsystems' states during periods of communication loss, thus resulting in the same controller being calculated for each subsystem. To account for the limited information available to each subsystem during communication loss the control action for each subsystem is based on the local measured states and on the mean values of the bounds of the unmeasured states. The performance of the robust controller combined with the robust observer was shown to be significantly better than that of a robust controller which uses a nominal model based observer for two different case studies: a distillation column and a reaction-separation system. The impact of the use of the robust observer on closed loop performance was particularly evident for the distillation column due to the ill conditioning of the process and its resulting sensitivity to model error. The implementation of the robust

controller and robust observer algorithm in the reactor system illustrated the computational feasibility of the algorithm for a system with several states.

6.4 Future Work

The findings of this work have helped identify several new topics for further research as follows:

1. Based on the results presented in Chapter 3 and Chapter 5, a distributed NMPC scheme based on PCE can be explored, by combining the results of the two chapters. This can be done by representing each local controller with a PCE-based Robust NMPC and including the model interactions (critical for overall performance of the distributed controller) as disturbances. These disturbances could be estimated by using the robust observer concept developed for linear models in Chapter 5. The robust observer developed in Chapter 5 can be computationally intensive for a large system, and hence there might be a need to develop robust estimators for each subsystem separately.
2. Pharmaceutical industries work under tight constraints on the amount of substrate feeding that can occur during a batch. The robust control algorithm did not consider this constraint because this limits the performance of the batch reactor drastically. One way to develop robust controller would be to define the constraints on controller actions using an offline robust optimization technique for the entire batch length. This type of design on control actions is commonly considered as “Quality by Design” where the control actions are developed by considering all the risks (uncertainty) in the processes.
3. Investigate experimental implementation of the robust control algorithm in Chapter 4 first in a bench scale reactor and eventually in a production unit. Since the algorithm takes into account constraints on input variables it could be implemented in an actual certified process while the bounds on inputs could be set as specified by the certification limits. Although tight constraints on inputs will limit the closed loop

performance they may still provide a better outcome at the end of the batch since the approach is based on the maximization of an end point quality.

Bibliography

Al-Gherwi, W., Budman, H. & Elkamel, A. 2011. A robust distributed model predictive control algorithm. *Journal of Process Control,* 21**,** 1127-1137.

Allgower, F., Findeisen, R. & Nagy, Z.K. 2004. Nonlinear model predictive control: From theory to application. *Journal of the Chinese Institute of Chemical Engineers,* 35**,** 299-315.

Banga, J.R., Alonso, A.A. & Singh, R.P. 1997. Stochastic dynamic optimization of batch and semicontinuous bioprocesses. *Biotechnology Progress,* 13**,** 326-335.

Boyd, S.P. 1994. *Linear matrix inequalities in system and control theory,* Philadelphia, Society for Industrial and Applied Mathematics.

Braatz, R.P., Young, P.M., Doyle, J.C. & Morari, M. 1994. Computational-complexity of mu-calculation. *IEEE Transactions on Automatic Control,* 39**,** 1000-1002.

Cannon, M., Buerger, J., Kouvaritakis, B. & Rakovic, S. 2011. Robust tubes in nonlinear model predictive control. *IEEE Transactions on Automatic Control,* 56**,** 1942-1947.

Chen, H. & Allgower, F. 1998. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica,* 34**,** 1205-1217.

Chen, L.B., Bastin, G. & V, V. 1995. A case-study of adaptive nonlinear regulation of fed-batch biological reactors. *Automatica,* 31**,** 55-65.

De La Pena, D.M. & Christofides, P.D. 2008. Lyapunov-based model predictive control of nonlinear systems subject to data losses. *IEEE Transactions on Automatic Control,* 53**,** 2076-2089.

Dewasme, L., Srinivasan, B., Perrier, M. & Vande Wouwer, A. 2011. Extremum-seeking algorithm design for fed-batch cultures of microorganisms with overflow metabolism. *Journal of Process Control,* 21**,** 1092-1104.

Diaz-Mendoza, R. & Budman, H. 2010a. Design of a robust nonlinear model predictive controller based on a hybrid model and comparison to other approaches. *Industrial & Engineering Chemistry Research,* 49**,** 11482-11490.

Diaz-Mendoza, R. & Budman, H. 2010b. Structured singular valued based robust nonlinear model predictive controller using volterra series models. *Journal of Process Control,* 20**,** 653-663.

Diehl, M., Gerhard, J., Marquardt, W. & Monigmann, M. 2008a. Numerical solution approaches for robust nonlinear optimal control problems. *Computers & Chemical Engineering,* 32**,** 1279-1292.

Diehl, M., Gerhard, J., Marquardt, W. & Monnigmann, M. 2008b. Numerical solution approaches for robust nonlinear optimal control problems. *Computers & Chemical Engineering,* 32**,** 1279-1292.

Doyle, F.J., Ogunnaike, B.A. & Pearson, R.K. 1995. Nonlinear model-based control using 2nd-order volterra models. *Automatica,* 31**,** 697-714.

Doyle, F.J., Packard, A.K. & Morari, M. 1989. Robust controller-design for a nonlinear CSTR. *Chemical Engineering Science,* 44**,** 1929-1947.

Findeisen, R., Imsland, L., Allgower, F. & Foss, B.A. 2003. State and output feedback nonlinear model predictive control: An overview. *European Journal of Control,* 9**,** 190-206.

Frahm, B., Lane, P., Atzert, H., Munack, A., Hoffmann, M., Hass, V.C. & Portner, R. 2002. Adaptive, model-based control by the open-loop-feedback-optimal (olfo) controller for the effective fed-batch cultivation of hybridoma cells. *Biotechnology Progress,* 18**,** 1095-1103.

Fruzzetti, K.P., Palazoglu, A. & Mcdonald, K.A. 1997. Nonlinear model predictive control using hammerstein models. *Journal of Process Control,* 7**,** 31-41.

Ghanem, R. & Spanos, P.D. 1990. Polynomial chaos in stochastic finite-elements. *Journal of Applied Mechanics-Transactions of the ASME,* 57**,** 197-202.

Ghanem, R.G. & Spanos, P.D. 1997. Spectral techniques for stochastic finite elements. *Archives of Computational Methods in Engineering,* 4**,** 63-100.

Heidarinejad, M., Liu, J.F., De La Pena, D.M., Davis, J.F. & Christofides, P.D. 2011. Handling communication disruptions in distributed model predictive control. *Journal of Process Control,* 21**,** 173-181.

Henson, M.A. 1998. Nonlinear model predictive control: Current status and future directions. *Computers & Chemical Engineering,* 23**,** 187-202.

Henson, M.A. 2010. Model-based control of biochemical reactors. *In:* Levine, W. S. (ed.) *The control handbook.* Second ed.: CRC Press.

Hjersted, J.L. & Henson, M.A. 2006. Optimization of fed-batch saccharomyces cerevisiae fermentation using dynamic flux balance models. *Biotechnology Progress,* 22**,** 1239-1248.

Hover, F.S. & Triantafyllou, M.S. 2006. Application of polynomial chaos in stability and control. *Automatica,* 42**,** 789-795.

Kawohl, M., Heine, T. & King, R. 2007. A new approach for robust model predictive control of biological production processes. *Chemical Engineering Science,* 62**,** 5212-5215.

Kim, K.K.K., Braatz, R.D. & Ieee 2012. Probabilistic analysis and control of uncertain dynamic systems: Generalized polynomial chaos expansion approaches. *2012 American Control Conference (Acc)***,** 44-49.

Konno, H. 1976. Maximization of a convex quadratic function under linear constraints. *Mathematical Programming,* 11**,** 117-127.

Kothare, M.V., Balakrishnan, V. & Morari, M. 1996. Robust constrained model predictive control using linear matrix inequalities. *Automatica,* 32**,** 1361-1379.

Kuhlmann, C., Bogle, I.D.L. & Chalabi, Z.S. 1998. Robust operation of fed batch fermenters. *Bioprocess Engineering,* 19**,** 53-59.

Kumar, D., Al-Gherwi, W. & Budman, H. 2012. Robust-distributed mpc with robust observer to handle communication loss. *IFAC 2012.* Singapore.

Kumar, D. & Budman, H. 2014. Robust nonlinear mpc based on volterra series and polynomial chaos expansions. *Journal of Process Control,* 24**,** 304-317.

Kumar, D. & Budman, H. 2015. Robust nonlinear predictive control for a bioreactor based on a dynamic metabolic flux balance model. *IFAC 2015.* Vancouver, Canada.

Kwakernaak, H. 1993. Robust control and hinfinity-optimization - tutorial paper. *Automatica,* 29**,** 255-273.

Lee, J., Lee, S.Y., Park, S. & Middelberg, A.P.J. 1999. Control of fed-batch fermentations. *Biotechnology Advances,* 17**,** 29-48.

Liu, J., De La Pena, D.M. & Christofides, P.D. 2009. Distributed model predictive control of nonlinear process systems. *AICHE Journal,* 55**,** 1171-1184.

Logist, F., Houska, B., Diehl, M. & Van Impe, J.F. 2011. Robust multi-objective optimal control of uncertain (bio)chemical processes. *Chemical Engineering Science,* 66**,** 4670-4682.

Lubbert, A. & Jorgensen, S.B. 2001. Bioreactor performance: A more scientific approach for practice. *Journal of Biotechnology,* 85**,** 187-212.

Löfberg, J. Year. Yalmip : A toolbox for modeling and optimization in matlab. *In:* CACSD Conference, 2004 Taipei, Taiwan.

Ma, D.L. & Braatz, R.D. 2001. Worst-case analysis of finite-time control policies. *IEEE Transactions on Control Systems Technology,* 9**,** 766-774.

Ma, D.L., Chung, S.H. & Braatz, R.D. 1999. Worst-case performance analysis of optimal batch control trajectories. *AICHE Journal,* 45**,** 1469-1476.

Maestre, J.M., De La Pena, D.M. & Camacho, E.F. 2009. A distributed mpc scheme with low communication requirements. *2009 American Control Conference, Vols 1-9***,** 2797-2802.

Magni, L., Raimondo, D.M. & Scattolini, R. 2006. Regional input-to-state stability for nonlinear model predictive control. *IEEE Transactions on Automatic Control,* 51**,** 1548-1553.

Magni, L. & Scattolini, R. 2010. An overview of nonlinear model predictive control. *Automotive Model Predictive Control: Models, Methods and Applications,* 402**,** 107-117.

Mahadevan, R., Edwards, J.S. & Doyle, F.J. 2002. Dynamic flux balance analysis of diauxic growth in escherichia coli. *Biophysical Journal,* 83**,** 1331-1340.

Mandur, J.S. & Budman, H.M. 2015. Simultaneous model identification and optimization in presence of model-plant mismatch. *Chemical Engineering Science,* 129**,** 106-115.

Maner, B.R. & Doyle, F.J. 1997. Polymerization reactor control using autoregressive-plus volterra-based mpc. *AICHE Journal,* 43**,** 1763-1784.

Mayne, D.Q., Kerrigan, E.C., Van Wyk, E.J. & Falugi, P. 2011. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control,* 21**,** 1341-1353.

Mayne, D.Q., Kerrigan, E. C., Falugi, P. Year. Robust model predictive control: Advantages and disadvantages of tube-based methods. *In:* IFAC World Congress, Aug 28-Sep 2, 2011 2011 Milano (Italy). 191-196.

Mayne, D.Q., Rakovic, S.V., Findeisen, R. & Allgower, F. 2006. Robust output feedback model predictive control of constrained linear systems. *Automatica,* 42**,** 1217-1222.

Mayne, D.Q., Seron, M.M. & Rakovic, S.V. 2005. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica,* 41**,** 219-224.

Mcfarlane, D. & Glover, K. 1992. A loop shaping design procedure using h-infinity-synthesis. *Ieee Transactions on Automatic Control,* 37**,** 759-769.

Michalska, H. & Mayne, D.Q. 1993. Robust receding horizon control of constrained nonlinear-systems. *IEEE Transactions on Automatic Control,* 38**,** 1623-1633.

Nagy, Z.K. & Braatz, R.D. 2003. Robust nonlinear model predictive control of batch processes. *AICHE Journal,* 49**,** 1776-1786.

Nagy, Z.K. & Braatz, R.D. 2004. Open-loop and closed-loop robust optimal control of batch processes using distributional and worst-case analysis. *Journal of Process Control,* 14**,** 411-422.

Nagy, Z.K. & Braatz, R.D. 2007. Distributional uncertainty analysis using power series and polynomial chaos expansions. *Journal of Process Control,* 17**,** 229-240.

Najm, H.N. 2009. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual Review of Fluid Mechanics,* 41**,** 35-52.

Norquay, S.J., Palazoglu, A. & Romagnoli, J.A. 1999. Application of wiener model predictive control (wmpc) to a ph neutralization experiment. *IEEE Transactions on Control Systems Technology,* 7**,** 437-445.

Nowak, R.D. & Vanveen, B.D. 1994. Random and pseudorandom inputs for volterra filter identification. *IEEE Transactions on Signal Processing,* 42**,** 2124-2135.

Parker, R.S., Heemstra, D., Doyle, F.J., Pearson, R.K. & Ogunnaike, B.A. 2001. The identification of nonlinear models for process control using tailored "plant-friendly" input sequences. *Journal of Process Control,* 11**,** 237-250.

Qin, S.J. & Badgwell, T.A. 2003. A survey of industrial model predictive control technology. *Control Engineering Practice,* 11**,** 733-764.

Rani, K.Y. & Rao, V.S.R. 1999. Control of fermenters - a review. *Bioprocess Engineering,* 21**,** 77-88.

Rawlings, J.B. & Stewart, B.T. 2008. Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control,* 18**,** 839-845.

Renard, F., Vande Wouwer, A., Valentinotti, S. & Dumur, D. 2006. A practical robust control scheme for yeast fed-batch cultures - an experimental validation. *Journal of Process Control,* 16**,** 855-864.

Scattolini, R. 2009. Architectures for distributed and hierarchical model predictive control - a review. *Journal of Process Control,* 19**,** 723-731.

Scheu, H. & Marquardt, W. 2011. Sensitivity-based coordination in distributed model predictive control. *Journal of Process Control,* 21**,** 715-728.

Skogestad, S., Morari, M. & Doyle, J.C. 1988. Robust-control of ill-conditioned plants - high-purity distillation. *IEEE Transactions on Automatic Control,* 33**,** 1092-1105.

Smets, I.Y., Claes, J.E., November, E.J., Bastin, G.P. & Van Impe, J.F. 2004. Optimal adaptive control of (bio)chemical reactors: Past, present and future. *Journal of Process Control,* 14**,** 795-805.

Smith, A.H.C., Monti, A. & Ponci, F. 2009. Uncertainty and worst-case analysis in electrical measurements using polynomial chaos theory. *IEEE Transactions on Instrumentation and Measurement,* 58**,** 58-67.

Srinivasan, B. & Bonvin, D. 2007. Real-time optimization of batch processes by tracking the necessary conditions of optimality. *Industrial & Engineering Chemistry Research,* 46**,** 492-504.

Srinivasan, B., Bonvin, D., Visser, E. & Palanki, S. 2003. Dynamic optimization of batch processes: Ii. Role of measurements in handling uncertainty. *Computers & Chemical Engineering,* 27**,** 27-44.

Sun, Y.L. & El-Farra, N.H. 2008. Quasi-decentralized model-based networked control of process systems. *Computers & Chemical Engineering,* 32**,** 2016-2029.

Terwiesch, P., Ravemark, D., Schenker, B. & Rippin, D.W.T. 1998. Semi-batch process optimization under uncertainty: Theory and experiments. *Computers & Chemical Engineering,* 22**,** 201-213.

Trodden, P. & Richards, A. 2006. Robust distributed model predictive control using tubes. *2006 American Control Conference, Vols 1-12,* 1-12**,** 2034-2039.

Vanantwerp, J.G. & Braatz, R.D. 2000. A tutorial on linear and bilinear matrix inequalities. *Journal of Process Control,* 10**,** 363-385.

Varma, A. & Palsson, B.O. 1994. Metabolic flux balancing - basic concepts, scientific and practical use. *Bio-Technology,* 12**,** 994-998.

Venkat, A.N., Rawlings, J.B. & Wright, S.J. 2005. Stability and optimality of distributed model predictive control. *2005 44th IEEE Conference on Decision and Control & European Control Conference, Vols 1-8***,** 6680-6685.

Xiu, D. & Tartakovsky, D.M. 2004. Uncertainty quantification for flow in highly heterogeneous porous media. *Computational Methods in Water Resources, Vols 1 and 2,* 55**,** 695-703.

Xiu, D.B. 2010. Generalized polynomial chaos. *Numerical Methods for Stochastic Computations: A Spectral Method Approach***,** 57-67.

Xiu, D.B. & Karniadakis, G.E. 2002. The wiener-askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing,* 24**,** 619-644.

Zavala, V.M. & Biegler, L.T. 2009. The advanced-step nmpc controller: Optimality, stability and robustness. *Automatica,* 45**,** 86-93.

Zhang, Y. & Li, S.Y. 2007. Networked model predictive control based on neighbourhood optimization for serially connected large-scale processes. *Journal of Process Control,* 17**,** 37-50.

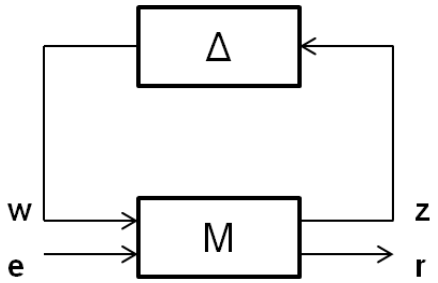
Appendix A
Interconnection Matrix

The robust controller presented in chapter 3 requires SSV calculation for terminal and input constraints. The equations to be solved have to be transformed to an appropriate interconnection matrix and uncertainty description.

A general form of the uncertain Volterra series can be represented as A. 1

$$\hat{y} = (h_{11} + \delta h_{11})u^2 + (h_1 + \delta h_1)u + d \qquad \textbf{A. 1}$$

Problem at hand is to formulate A. 1 into corresponding $\boldsymbol{M}, \boldsymbol{\Delta}$ so that the relationship between r, e is shown by RHS of A. 1.



To this end following structure of $\boldsymbol{M}, \boldsymbol{\Delta}$ is proposed

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \vdots & k_{SSV} \\ 0 & 0 & 0 & 0 & 0 & \vdots & k_{SSV}u \\ 0 & 0 & 0 & 0 & 0 & \vdots & k_{SSV}u^2 \\ 0 & k_{SSV} & 0 & 0 & 0 & \vdots & 0 \\ 0 & 0 & k_{SSV} & 0 & 0 & \vdots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ d & h_1 & h_{11} & \delta h_1 & \delta h_{11} & \vdots & 0 \end{bmatrix} \qquad \textbf{A. 2}$$

$$\boldsymbol{\Delta} = \delta_1 \boldsymbol{I_{5x5}}$$

$$\begin{bmatrix} \mathbf{z} \\ r \end{bmatrix} = \begin{bmatrix} \boldsymbol{M_{11}} & \boldsymbol{M_{12}} \\ \boldsymbol{M_{21}} & \boldsymbol{M_{22}} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ e \end{bmatrix}$$

$$\begin{bmatrix} w(1) \\ w(2) \\ w(3) \\ w(4) \\ w(5) \end{bmatrix} = \boldsymbol{\Delta} \begin{bmatrix} z(1) \\ z(2) \\ z(3) \\ z(4) \\ z(5) \end{bmatrix} = \delta_1 \begin{bmatrix} z(1) \\ z(2) \\ z(3) \\ z(4) \\ z(5) \end{bmatrix} \qquad \textbf{A. 3}$$

Going element by element of $\boldsymbol{z}$ vector and finally deriving the relationship between r and e.

$$\begin{aligned} z(1) &= k_{SSV} e \\ z(2) &= k_{SSV} u\, e \\ z(3) &= k_{SSV} u^2\, e \\ z(4) &= k_{SSV} w(2) = k_{SSV} \delta_1 z(2) = k_{SSV}^2 \delta_1 u\, e \\ z(5) &= k_{SSV} w(3) = k_{SSV} \delta_1 z(3) = k_{SSV}^2 \delta_1 u^2\, e \end{aligned} \qquad \textbf{A. 4}$$

Substitute for $\boldsymbol{w}$ and $\boldsymbol{z}$ in the equation for r

$$\begin{aligned} r &= \boldsymbol{M_{21}} \boldsymbol{w} + \boldsymbol{M_{22}} e \\ r &= d\, w(1) + h_1 w(2) + h_{11} w(3) + \delta h_1 w(4) + \delta h_{11} w(5) \\ r &= \delta_1 [d\, z(1) + h_1 z(2) + h_{11} z(3) + \delta h_1 z(4) + \delta h_{11} z(5)] \\ r &= k_{SSV} \delta_1 [d + (h_1 + k_{SSV} \delta_1 \delta h_1)\, u + (h_{11} + k_{SSV} \delta_1 \delta h_{11}) u^2] e \end{aligned} \qquad \textbf{A. 5}$$

Equation A.5 represents skew-μ formulation of A.1.

Appendix B

Model Parameters for Reactor-Separator Case Study

Following are the parameters used to define the robust model used in Section 5.3.2 for Reactor-Separator process.

$A_{nominal} =$

0.3932	-0.0017	-0.4626	0.0770	-0.0022	-0.0693	0.1783	-0.0107	-0.1493
0.1917	0.3963	3.1374	0.0154	0.0418	0.3775	0.0424	0.2300	1.0364
0.0038	0.0000	0.5061	0.0005	0.0000	0.1123	0.0008	0.0000	0.2819
0.3689	-0.0007	-0.4459	0.1136	-0.0011	-0.1319	0.1035	-0.0069	-0.1195
0.1879	0.4055	3.0585	0.0439	0.1025	0.8037	0.0266	0.1434	0.8219
0.0035	0.0000	0.4725	0.0009	0.0000	0.1511	0.0006	0.0000	0.1862
0.4046	0.0034	-0.2713	0.3918	0.0047	-0.1508	0.6620	0.0259	-0.0520
0.0919	0.1749	0.7593	0.0902	0.1532	0.3735	0.1660	0.5942	0.1458
0.0015	0.0000	0.2991	0.0007	0.0000	0.2282	0.0002	0.0000	0.3630

$A^{[1]} =$

0.1966	-0.0009	-0.2313	0.0385	-0.0011	-0.0346	0.1782	-0.0107	-0.1493
0.0958	0.1981	1.5687	0.0077	0.0209	0.1887	0.0424	0.2300	1.0363
0.0019	0.0000	0.2531	0.0002	0.0000	0.0562	0.0008	0.0000	0.2819
0.1845	-0.0003	-0.2230	0.0568	-0.0006	-0.0660	0.1035	-0.0069	-0.1195
0.0940	0.2027	1.5293	0.0220	0.0512	0.4019	0.0266	0.1434	0.8218
0.0017	0.0000	0.2362	0.0004	0.0000	0.0756	0.0006	0.0000	0.1861
0.2023	0.0017	-0.1357	0.1959	0.0023	-0.0754	0.6620	0.0259	-0.0520
0.0459	0.0874	0.3796	0.0451	0.0766	0.1868	0.1660	0.5941	0.1458
0.0007	0.0000	0.1495	0.0003	0.0000	0.1141	0.0002	0.0000	0.3629

$A^{[2]} =$

0.1966	-0.0009	-0.2313	0.1001	-0.0028	-0.0900	0.1782	-0.0107	-0.1493
0.0958	0.1981	1.5687	0.0201	0.0543	0.4907	0.0424	0.2300	1.0363
0.0019	0.0000	0.2531	0.0006	0.0000	0.1460	0.0008	0.0000	0.2819
0.1845	-0.0003	-0.2230	0.1477	-0.0014	-0.1715	0.1035	-0.0069	-0.1195
0.0940	0.2027	1.5293	0.0571	0.1332	1.0448	0.0266	0.1434	0.8218
0.0017	0.0000	0.2362	0.0012	0.0000	0.1965	0.0006	0.0000	0.1861
0.2023	0.0017	-0.1357	0.5093	0.0061	-0.1961	0.6620	0.0259	-0.0520
0.0459	0.0874	0.3796	0.1172	0.1991	0.4856	0.1660	0.5941	0.1458
0.0007	0.0000	0.1495	0.0009	0.0000	0.2967	0.0002	0.0000	0.3629

$A^{[3]} =$

0.4522	-0.0020	-0.5320	0.0385	-0.0011	-0.0346	0.1782	-0.0107	-0.1493
0.2204	0.4557	3.6080	0.0077	0.0209	0.1887	0.0424	0.2300	1.0363
0.0043	0.0000	0.5820	0.0002	0.0000	0.0562	0.0008	0.0000	0.2819
0.4243	-0.0008	-0.5128	0.0568	-0.0006	-0.0660	0.1035	-0.0069	-0.1195
0.2161	0.4663	3.5173	0.0220	0.0512	0.4019	0.0266	0.1434	0.8218
0.0040	0.0000	0.5434	0.0004	0.0000	0.0756	0.0006	0.0000	0.1861
0.4653	0.0039	-0.3120	0.1959	0.0023	-0.0754	0.6620	0.0259	-0.0520
0.1057	0.2011	0.8732	0.0451	0.0766	0.1868	0.1660	0.5941	0.1458
0.0017	0.0000	0.3439	0.0003	0.0000	0.1141	0.0002	0.0000	0.3629

$A^{[4]} =$

0.4522	-0.0020	-0.5320	0.1001	-0.0028	-0.0900	0.1782	-0.0107	-0.1493
0.2204	0.4557	3.6080	0.0201	0.0543	0.4907	0.0424	0.2300	1.0363
0.0043	0.0000	0.5820	0.0006	0.0000	0.1460	0.0008	0.0000	0.2819
0.4243	-0.0008	-0.5128	0.1477	-0.0014	-0.1715	0.1035	-0.0069	-0.1195
0.2161	0.4663	3.5173	0.0571	0.1332	1.0448	0.0266	0.1434	0.8218
0.0040	0.0000	0.5434	0.0012	0.0000	0.1965	0.0006	0.0000	0.1861
0.4653	0.0039	-0.3120	0.5093	0.0061	-0.1961	0.6620	0.0259	-0.0520
0.1057	0.2011	0.8732	0.1172	0.1991	0.4856	0.1660	0.5941	0.1458
0.0017	0.0000	0.3439	0.0009	0.0000	0.2967	0.0002	0.0000	0.3629

$$\boldsymbol{B}_{nominal} = \boldsymbol{B}^{(1)} = \boldsymbol{B}^{(2)} = \boldsymbol{B}^{(3)} = \boldsymbol{B}^{(4)}$$

$$\begin{vmatrix} -0.0102 & 0.0054 & -0.0017 & -0.0020 \\ 0.0711 & -0.0226 & 0.0092 & 0.0139 \\ 0.0236 & -0.0056 & 0.0040 & 0.0061 \\ -0.0076 & 0.0397 & -0.0098 & -0.0013 \\ 0.0523 & -0.2705 & 0.0608 & 0.0088 \\ 0.0133 & -0.0408 & 0.0290 & 0.0028 \\ -0.0030 & 0.0279 & -0.0056 & -0.0004 \\ 0.0085 & -0.0791 & 0.0139 & 0.0011 \\ 0.0049 & -0.0214 & 0.0151 & 0.0184 \end{vmatrix}$$

Appendix C

MATLAB Codes

PCE-based RNMPC for Chapter 3

```
%-----------------------------------------------------------------
% Main program to implement NMPC
%-----------------------------------------------------------------

function main_program_chapter6_case_study_3(w1, w2, disturbance, ph_sp,
ph, ...
    nit, q2, Cv, n_valve, in_co, ss_condn, filename)
% w1                % weighting factor for manipultd var, 1.
% w2                % weighting factor for manipultd var, 2.
% q2                % process parameter, q2
% Cv                % process parameter, Cv
% n                 % process parameter, n
% in_co             % initial conditions
% ss_condn          % Steady State Conditions
% filename          % filename for storing the results
%% Enter Disturbance and Set-point Data, change these if nit is changed
weight      = [w1 w2];                  % manipulated variable movement
weight
% load('chapter6_disturbance4');              % Process disturbance
% disturbance = zeros(100,1);
% load('ph_setpoint_profile_2');              % set-point changes in pH

%% SS and other information for ODE
t0 = 0;  % Initial time for ODE simulation
tf = 25; % Final time for ODE simulation

%% Parameter information************************************************
d_ss_nom = [ q2 ; Cv; n_valve ];

%% info for y conversions ******************************************

pr = zeros(2,1);
pr(1) = 14;      % tank height
pr(2) = 07.0016; % pH

dv = zeros(2,1);
dv(1) = 5.0;
dv(2) = 3.0;

%% infor for u conversions *****************************************

prm = zeros(2,1);
prm(1) = ss_condn(1,1);
prm(2) = ss_condn(1,2);
```

```
pv1mv = 0.2;
pv2mv = 0.225;

dvm = zeros(2,1);
dvm(1) = pv1mv*prm(1,1);
dvm(2) = pv2mv*prm(2,1);
load('corrida2_ny1')
load('corrida2_ny2')

[hy1,h11_ll,h11_cp,h12_ll,h12_cp] = dah(pny1);
[vhy1, vh11_ll,vh11_cp,vh12_ll,vh12_cp] = dah(vny1);

[hy2,h21_ll,h21_cp,h22_ll,h22_cp] = dah(pny2);
[vhy2,vh21_ll,vh21_cp,vh22_ll,vh22_cp] = dah(vny2);
%% Defining Uncertain parts of Volterra Parameters
hy1_phi_2    = 0;
h11_ll_phi_2 = zeros(1,3);
h21_ll_phi_2 = zeros(1,3);
h11_cp_phi_2 = zeros(3,3);
h21_cp_phi_2 = zeros(3,3);

hy2_phi_2    = 0;
h22_ll_phi_2 = zeros(1,3);
h12_ll_phi_2 = zeros(1,3);
h22_cp_phi_2 = zeros(3,3);
h12_cp_phi_2 = zeros(3,3);

hy1_phi_1    = vhy1;        %1-D PCE
h11_ll_phi_1 = vh11_ll;
h21_ll_phi_1 = vh21_ll;
h11_cp_phi_1 = vh11_cp;
h21_cp_phi_1 = vh21_cp;

hy2_phi_1    = vhy2;        % 1-D PCE
h22_ll_phi_1 = vh22_ll;
h12_ll_phi_1 = vh12_ll;
h22_cp_phi_1 = vh22_cp;
h12_cp_phi_1 = vh12_cp;


%% Robust Design Parameters
opmu = 'dU';        % Parameters for calling mussv
                    % d = display any warnings
                    % U = calculate only the upper bound
blk = [...
    -40 0 ; ...
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 1
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 2
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 3
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 4
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 5
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 6
```

```
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 7
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 8
    -01*ones(4,1) , 0*ones(4,1) ; ... % Block 9
    2 2];
esi = 1E-5;     % used for entering in Interconnection matrix (M22)

%% Optimisation Options

% op --- optimisation parameters for Robust Formulation
op = optimset('fminsearch');
op.Display = 'Off';
op.TolFun = 1E-4;
op.TolX   = 1E-3;

% opco --- optimisation parameters for non-Robust formulation
opco = optimset('fminsearch');
opco.Display = 'Off';
opco.TolFun = 1E-4;
opco.TolX   = 1E-3;
% opco.TolX = value;

%% MPC Parameters and initialisation for 1st iteration

m_ch = 2;                       % control horizon
ed   = zeros(20,1);             % disturbance vector
ic   = zeros(20,1);             % initial conditions vector
du1  = weight(1,1)*ones(1,2);   % weight movement u1 (1x2)
% du1 = [weight(1,1) weight(1,1)/10];
u1r  = Inf*ones(2,1);           % restrictions u1
du2  = weight(1,2)*ones(1,2);   % weight movement u2
% du2 = [weight(1,2) weight(1,2)/10];
u2r  = Inf*ones(2,1);           % restrictions u2
vtc = [0.02;0.02];                % terminal conditions (2x1)
e_s = 1E-6;                     % value of e_s for zone change

ei1 =  [-0.05; 0.05] ;           % value for the initial estimates, it is
related
ei2 = [-0.07; 0.07] ;            % value for the initial estimates, it is
related
ei3 = [-0.08; 0.08] ;             % value for the initial estimates, it is
related
ei4 = [-0.2; 0.2] ;             % value for the initial estimates, it is
related
ei5 = [-0.35; 0.25] ;             % value for the initial estimates, it is
related
ei6 = [-0.55; 0.45] ;             % value for the initial estimates, it is
related
ei7 = [-0.8; 0.7] ;             % value for the initial estimates, it is
related
                                % to the value of the disturbance
% nit = 80;                      % number of sampling instants to consider
```

```
ypast1       = [0;0];           % y(k-1) reinitialised at every time step
(2x1)
uatk_minus_1 = [0;0];           % u1(k-1)reinitialised at every time step
(2x1)
uatk_minus_2 = [0;0];           % u1(k-2)reinitialised at every time step
(2x1)
kuig = zeros(4,1);              % initial estimates for manipulated
variable
                                % (4x1)= [u1(2x1); u2(2x1)]
% variables to be stored at every iteration
ycd   = zeros(nit,2);           % y with disturbance
ycd_sp = zeros(nit,2);          % y with disturbance - y_sp
mv    = zeros(nit,4);           % manipuated variable (2 cols for m_ch)
ymcd  = zeros(nit,2);           % y with matrix
d_i   = zeros(nit,2);           % disturbances
tcito = zeros(nit,1);           % time vector
y10   = zeros(nit,2);           % y at p=10
wc    = zeros(nit-1,4);         % condition for changing the controller
ynode = zeros(nit,2);           % y from ODE
h_n   = zeros(nit,1);           % boolean with or without variation
determined from cal_nom
n_nit = zeros(nit,1);           % stores the index for ymax in f_robust
and f_nonrobust
y_robust_state = zeros(nit,6);  % stores the y returned by f_robust_state
at each iteration
y_PCE_state = zeros(20,3,nit);
cal_nom = 0;                    % --> caso without parameter variation
iae     = zeros(1,1);
d_ss = d_ss_nom;
%-------------------------------------------------------------------------
-
%% Iterations and Clock Start

tit = clock;           % start the clock
for i=1:nit

    tstart = clock;

    fprintf('\nIteration %2.0f',i)
    % Calculate Initial Conditions for the ouptput

    ic(1:10,1) = 0;
    ic(11:20,1) = 0;



    % search for good initial guess
    % search for the initial guess of manipulated variable without
    % uncertain volterra series parameters, i.e w/o SSV

    if(cal_nom==1)

        [umu, fv] = fminsearch(@(ku)f_nonrobust(ed,ku,...
```

```
        h11_ll,h11_cp, h12_ll, h12_cp,...
        hy1,...
        h21_ll,h21_cp, h22_ll, h22_cp,...
        hy2,...
        ic, ...
        uatk_minus_1,uatk_minus_2,...
        ypast1, ...
        du1, du2,...
        u1r, u2r,...
        vtc,...
        m_ch),...
        kuig,opco);
    [fv, n] = f_nonrobust(ed,umu,...
        h11_ll,h11_cp, h12_ll, h12_cp,...
        hy1,...
        h21_ll,h21_cp, h22_ll, h22_cp,...
        hy2,...
        ic, ...
        uatk_minus_1, uatk_minus_2,...
        ypast1, ...
        du1, du2,...
        u1r, u2r,...
        vtc,...
        m_ch) ;
    fprintf('\nControlling factor %1.5f',n)
elseif(cal_nom==0)

    [kuig, fv] = fminsearch(@(ku)f_initial_guess(ed,ku,...
        h11_ll,h11_cp, h12_ll, h12_cp,...
        hy1,...
        h21_ll,h21_cp, h22_ll, h22_cp,...
        hy2,...
        ic, ...
        uatk_minus_1, uatk_minus_2,...
        ypast1, ...
        du1, du2,...
        u1r, u2r,...
        vtc,...
        m_ch), kuig, ...
        optimset('Display', 'Off'));

    kuig = 0.99*kuig;

    [umu,fv] = fminsearch(@(ku)f_robust(blk,esi,ed,ku,...
        h11_ll, h11_cp, h12_ll, h12_cp,...
        hy1,...
        h21_ll,h21_cp, h22_ll, h22_cp,...
        hy2,...
        h11_ll_phi_1, h11_cp_phi_1, h12_ll_phi_1, h12_cp_phi_1,...
        hy1_phi_1,...
        h21_ll_phi_1,h21_cp_phi_1, h22_ll_phi_1, h22_cp_phi_1,...
        hy2_phi_1,...
        h11_ll_phi_2, h11_cp_phi_2, h12_ll_phi_2, h12_cp_phi_2,...
```

```
    hy1_phi_2,...
    h21_ll_phi_2,h21_cp_phi_2, h22_ll_phi_2, h22_cp_phi_2,...
    hy2_phi_2,...
    vh11_ll,vh11_cp,vh12_ll, vh12_cp,...
    vh21_ll, vh21_cp, vh22_ll, vh22_cp,...
    ic, ...
    uatk_minus_1, uatk_minus_2,...
    ypast1, ...
    du1, du2,...
    vtc,...
    m_ch,...
    opmu),...
    kuig,op);

[fv, n] = f_robust(blk,esi,ed,umu,...
    h11_ll, h11_cp, h12_ll, h12_cp,...
    hy1,...
    h21_ll,h21_cp, h22_ll, h22_cp,...
    hy2,...
    h11_ll_phi_1, h11_cp_phi_1, h12_ll_phi_1, h12_cp_phi_1,...
    hy1_phi_1,...
    h21_ll_phi_1,h21_cp_phi_1, h22_ll_phi_1, h22_cp_phi_1,...
    hy2_phi_1,...
    h11_ll_phi_2, h11_cp_phi_2, h12_ll_phi_2, h12_cp_phi_2,...
    hy1_phi_2,...
    h21_ll_phi_2,h21_cp_phi_2, h22_ll_phi_2, h22_cp_phi_2,...
    hy2_phi_2,...
    vh11_ll,vh11_cp,vh12_ll, vh12_cp,...
    vh21_ll, vh21_cp, vh22_ll, vh22_cp,...
    ic, ...
    uatk_minus_1, uatk_minus_2,...
    ypast1, ...
    du1, du2,...
    vtc,...
    m_ch,...
    opmu);
y_robust_state(i,:) = f_robust_state(blk,esi,ed,umu,...
    h11_ll, h11_cp, h12_ll, h12_cp,...
    hy1,...
    h21_ll,h21_cp, h22_ll, h22_cp,...
    hy2,...
    h11_ll_phi_1, h11_cp_phi_1, h12_ll_phi_1, h12_cp_phi_1,...
    hy1_phi_1,...
    h21_ll_phi_1,h21_cp_phi_1, h22_ll_phi_1, h22_cp_phi_1,...
    hy2_phi_1,...
    h11_ll_phi_2, h11_cp_phi_2, h12_ll_phi_2, h12_cp_phi_2,...
    hy1_phi_2,...
    h21_ll_phi_2,h21_cp_phi_2, h22_ll_phi_2, h22_cp_phi_2,...
    hy2_phi_2,...
    vh11_ll,vh11_cp,vh12_ll, vh12_cp,...
    vh21_ll, vh21_cp, vh22_ll, vh22_cp,...
    ic, ...
    uatk_minus_1, uatk_minus_2,...
```

```
        ypast1, ...
        du1, du2,...
        vtc,...
        m_ch,...
        opmu);

    fprintf('\nControlling factor %1.5f',n)
end
u = zeros(20,1);
u(1:m_ch,1) = umu(1:m_ch,1);
u(m_ch + 1:10,1) = umu(m_ch);
u(11:10+m_ch , 1)= umu(3:2*m_ch,1);
u(11+m_ch:20,1) = umu(2*m_ch);
%-------------------------------------------------
y_PCE_state(:,:,i) = PCE_parameter_Volterra_state(u, ed, ic, ...
    uatk_minus_1, uatk_minus_2,...
    ypast1, ...
    h11_ll, h11_ll_phi_1, h11_ll_phi_2, ...
    h11_cp, h11_cp_phi_1, h11_cp_phi_2, ...
    h12_ll, h12_ll_phi_1, h12_ll_phi_2, ...
    h12_cp, h12_cp_phi_1, h12_cp_phi_2, ...
    hy1, hy1_phi_1, hy1_phi_2, ...
    hy2, hy2_phi_1, hy2_phi_2, ...
    h22_ll, h22_ll_phi_1, h22_ll_phi_2, ...
    h21_ll, h21_ll_phi_1, h21_ll_phi_2, ...
    h22_cp, h22_cp_phi_1, h22_cp_phi_2, ...
    h21_cp, h21_cp_phi_1, h21_cp_phi_2);

% Nominal part of Volterra series using the new manipulated variable.
ypast1(1,1) = h11_ll(1,1)*u(1,1) + ...
    h11_ll(1,2)*uatk_minus_1(1,1) + ...
    h11_ll(1,3)*uatk_minus_2(1,1) + ...
    h11_cp(1,1)*u(1,1)^2 + ...
    h11_cp(1,2)*u(1,1)*uatk_minus_1(1,1) + ...
    h11_cp(1,3)*u(1,1)*uatk_minus_2(1,1) + ...
    h11_cp(2,2)*uatk_minus_1(1,1)^2 + ...
    h11_cp(2,3)*uatk_minus_1(1,1) * uatk_minus_2(1,1) + ...
    h11_cp(3,3)*uatk_minus_2(1,1)^2 + ...
    hy1*ypast1(1,1)+ ...
    h12_ll(1,1)*u(11,1) + ...
    h12_ll(1,2)*uatk_minus_1(2,1) + ...
    h12_ll(1,3)*uatk_minus_2(2,1) + ...
    h12_cp(1,1)*u(11,1)^2 + ...
    h12_cp(1,2)*u(11,1)*uatk_minus_1(2,1) + ...
    h12_cp(1,3)*u(11,1)*uatk_minus_2(2,1) + ...
    h12_cp(2,2)*uatk_minus_1(2,1)^2 + ...
    h12_cp(2,3)*uatk_minus_1(2,1) * uatk_minus_2(2,1) + ...
    h12_cp(3,3)*uatk_minus_2(2,1)^2 + ...
    ic(1,1);

ypast1(2,1) = h21_ll(1,1)*u(1,1) + ...
    h21_ll(1,2)*uatk_minus_1(1,1) + ...
    h21_ll(1,3)*uatk_minus_2(1,1) + ...
```

```
        h21_cp(1,1)*u(1,1)^2 + ...
        h21_cp(1,2)*u(1,1)*uatk_minus_1(1,1) + ...
        h21_cp(1,3)*u(1,1)*uatk_minus_2(1,1) + ...
        h21_cp(2,2)*uatk_minus_1(1,1)^2 + ...
        h21_cp(2,3)*uatk_minus_1(1,1) * uatk_minus_2(1,1) + ...
        h21_cp(3,3)*uatk_minus_2(1,1)^2 + ...
        hy2*ypast1(2,1)+ ...
        h22_ll(1,1)*u(11,1) + ...
        h22_ll(1,2)*uatk_minus_1(2,1) + ...
        h22_ll(1,3)*uatk_minus_2(2,1) + ...
        h22_cp(1,1)*u(11,1)^2 + ...
        h22_cp(1,2)*u(11,1)*uatk_minus_1(2,1) + ...
        h22_cp(1,3)*u(11,1)*uatk_minus_2(2,1) + ...
        h22_cp(2,2)*uatk_minus_1(2,1)^2 + ...
        h22_cp(2,3)*uatk_minus_1(2,1) * uatk_minus_2(2,1) + ...
        h22_cp(3,3)*uatk_minus_2(2,1)^2 + ...
        ic(11,1);
    %-------------------------------------------------------------------
    %--ODE Calculation for yplant-------------------------------------
    uab = i_y_srev([u(01),u(11)],prm,dvm);
    [t yode] = ode45(@phneu2,[t0 tf],in_co,[],...
        uab',d_ss);
    d_ph = vcalph(yode(end,:)); % Solve for pH using ODE solution
    in_co = yode(end,:);
    yreal = n_y_srev([yode(end,1),d_ph],pr,dv);% yplant at time t = i
    %-------------------------------------------------------------------
    % Calcualte disturbance (ed) and reinitialise other parameters
    ed = [(yreal(1) - ypast1(1,1)) * ones(10,1); ...
        (yreal(2) - ypast1(2,1) - ph_sp(i)) * ones(10,1)]; % Next
unmeasured disturbance
    uatk_minus_2 = uatk_minus_1;            % next u1(k-2)
    uatk_minus_1 = [u(01); u(11)];          % next u1(k-1)

    % Next disturbance to the plant
    if (i > 1)
        d_ss(1) = d_ss_nom(1) + disturbance(i)*d_ss_nom(1);
        d_ss(2:3) = d_ss_nom(2:3);
    end
    %-------------------------------------------------------------------
    % Saving all the data for each time step like unmeasured disturbance,
    % manipulated variable ...
    % and yplant from ODE

    ynode(i,:) = [yode(end,1) , d_ph];      % Absolute value of output
    d_i(i,:)   = [ed(1,1) ed(11,1)];        % Unmeasured disturbance
    mv(i,:)    = [u(01) u(02) u(11) u(12)]; % manipulated variable
    ycd(i,:)   = yreal;
    ymcd(i,:)  = ypast1';
    n_nit(i,:) = n;

    if(i==1)
        ycd_sp(i,:) = ycd(i,:);
    else
```

```
        ycd_sp(i,:) = [ycd(i,1) ycd(i,2)-ph_sp(i-1)];
    end
    % Normalised value of output
    % from the plant
    clear tode yode
    %-------------------------------------------------------------------
    %-----Dual Mode Decision Criterion and initialisation of kuig------
    if (i == 3 )
        kuig = [ei1(1,1)*ones(m_ch,1);ei1(2,1)*ones(m_ch,1)];
    elseif(i==4)
        kuig = [ei2(1,1)*ones(m_ch,1);ei2(2,1)*ones(m_ch,1)];
    elseif(i==5)
        kuig = [ei3(1,1)*ones(m_ch,1);ei3(2,1)*ones(m_ch,1)];
    else
        kuig = 0.99*umu;
    end

    % abs(ycd(i,1)-ycd(i-1,1))<e_s && abs(ycd(i,2)-ycd(i-1,2))...
    if(i>1)
        wc(i-1,:) = [abs(ycd(i,1)-ycd(i-1,1)) ...
            abs(ycd(i,2)-ycd(i-1,2)-(ph_sp(i)-ph_sp(i-1))) ...
            abs(ycd(i,1)) abs(ycd(i,2)- ph_sp(i))];
        if(cal_nom==0)
            if (wc(i-1,1)<e_s && wc(i-1,2)<e_s && wc(i-1,3)<e_s && ...
                    wc(i-1,4)<e_s)
                cal_nom = 1;        % nominal case
                h_n(i,1) = 1;
%                 kuig = [ei(1,1)*ones(m_ch,1); ei(2,1)*ones(m_ch,1)];
            else
                h_n(i,1)= 0;        %Uncertain case
            end
        end
        if(cal_nom ==1)
            if(abs(yreal(1))>e_s || abs(yreal(2))>e_s)
                cal_nom = 0;        % Uncertain case
                h_n(i,1) = 0;
            else
                h_n(i,1) = 1;   % Nominal Case
            end
        end
    end

    tcito(i,1) = etime(clock,tstart);   %Saving the calculation time
    % needed at every time step i
    fprintf('\nabs(yreal) = %1.5f',abs(yreal))
    fprintf('\n')
    fprintf('manipultd var = %1.5f', umu)
    fprintf('\n*****************************************************\n')
end
%-------------------------------------------------------------------
%% Print the final results-------------------------------------------
fprintf('total time = %1.5f',etime(clock,tit))  %total time
fprintf('\n')
```

```
iae = sum(sum(abs(ycd_sp),1));
end
%-------------------------------------------------------------------------
```

PCE-based Robust Control for Chapter 4

```
%-------------------------------------------------------------------
```

% Main function for model, control parameters and initial conditions

```
%-------------------------------------------------------------------

clear all
clc
% Initial Conditions
z0_model = [0.4 0.21 .2 0.001]';
V0 = 0.3;   % L, Initial Volume of the reactor, guess
Vmax = 0.4; % L, Final maximum batch volume, guess
Vmin = 0.2; % L, Final minimum batch volume, guess
Fmax = 0.1;   % L/h
Zgl_feed = 5;
%% Model parameters
A = [0 9.46 9.84 19.23; 35 12.92 12.73 0; -39.43 0 1.24 12.12];
c = ones(4,1);
kla = 4.0;  % hr^-1, Mahadevan paper
Km = 0.015; % mM, Mahadevan paper
GUR_max = 6.5;   % mM/g-dw/hr, Mahadevan paper
OUR_max = 12.0;   % mM/g-dw/hr, Mahadevan paper
Ki = 1.0;
% Uncertainty Information
GUR_sig = 0.2;  % +/- 20%, guess
OUR_sig = 0.2;  % +/- 20%, guess
kla_sig = 0.2;
Km_sig = 0.01;
Ki_sig = 0.2;

% # of time steps and step size
nit = 22;
tend = 11;   % h, total time of cell culture growth, guess
dt = tend/nit;   % h, time, guess

% frequency for disturbance
% t = [0:1:nit]'; n = 12; % frequency for disturbance
% disturbance = sin(2*pi/n*t);
% n2 = 8; % frequency for changing plant definition
load('perf_disturbance.mat');
load('perf_disturbance2.mat');
beta = 0; % cell death parameter
%% PCE parameters
n_dim = 1;      % PCE dimensions (1 or 2), max is 2
n_order = 2;     % order of PCE, max is 3
l = 4;
```

```
n_PCE = factorial(n_dim+n_order)/factorial(n_dim)/factorial(n_order);   %
total # of PCE terms
%% controller parameters
p = nit;
% u = 0.001*ones(p,1);
load('XV_reference_nit_22_tend_11.mat');
uig = u;
load('XV_reference_nit_22_tend_11_aug_2_2014.mat');
XVref_lb = XV_lb_bound;
XVref_ub = XV_ub_bound;
% uig = [0.002*ones(nit,1); 0.001*ones(nit,1)];


% load the biomass reference profile and uig
Vref = zeros(nit,1);
for i=1:nit
    Vref(i,1) = (V0 + (ones(1,i)*(uig(1:i)-uig(nit+1:nit+i)))*dt);
end

% Optimisation parameters
op = optimset('fmincon');
op.Display = 'On';
op.TolFun = 1E-6;
op.TolX   = 1E-5;
op.MaxIter = 10000;
op.MaxFunEvals = 100000;
op.Algorithm = 'interior-point';

% objective function weightsw = [10 0.05 0.5];   % e_x-xref, var(x)
%% Plant parameters
alpha_p_list =  [0 0; ...
                 -2 0; ...
                 -1 0; ...
                 -0.5 0; ...
                 0.5 0; ...
                 0 -2; ...
                 -2 -2; ...
                 -1 -2; ...
                 -0.5 -2; ...
                 0.5 -2;...
                 ];
w_list = [10 20; 3 30; 3 10; 20 20; 20 30; ];
% GUR_plant = GUR_max;
%% the control and plant loop
% initialisation
z0_plant = [0.4 0.21 .2 0.001]';
fb_k = 0;                           % initialisation of fb error
alpha_p = alpha_p_list(6,:);
matlabpool(2)
parfor i = 1:2
    w = w_list(i,:);
    filename =
strcat('robust_kla_dist_econ_obj7_robust_model_bds5_nit_22_', ...
```

```
        num2str(alpha_p(1)),'_', num2str(alpha_p(2)), '_',
num2str(w(1)),...
         '_', num2str(w(2)),'_P_F.mat');
    main_controller3(n_PCE, n_dim, n_order, l, nit, dt, tend, ...
        A, c, kla, kla_sig, Km, GUR_max, GUR_sig, OUR_max, Ki, Ki_sig, ...
        alpha_p, uig, Fmax, fb_k, disturbance, disturbance2,  ...
        p, op, z0_model, z0_plant, V0, Vmax, Vmin, Zgl_feed, ...
        OUR_sig, beta, XVref_lb, XVref_ub,w, filename)
end
matlabpool close %
end

%-------------------------------------------------------------------------------------------------------
% Main Controller
%-------------------------------------------------------------------
function main_controller3(n_PCE, n_dim, n_order, l, nit, dt, tend, ...
                    A, c, kla, kla_sig, Km, GUR_max, GUR_sig, OUR_max, Ki,
Ki_sig, ...
                    alpha_p, uig, Fmax, fb_k, disturbance, disturbance2,
...
                    p, op, z0_model, z0_plant, V0, Vmax, Vmin, Zgl_feed,
...
                    OUR_sig, beta, XVref_lb, XVref_ub, w, filename)
% variables to store
z0_model = [z0_model; zeros(4*(n_PCE-1),1)];    % robust model predictions
at every time step for current concentrations

z_plant = zeros(nit,3);
y_plant = zeros(nit,1);
V_plant = zeros(nit,1);
u_plant = zeros(2*nit,p);
nu_plant = zeros(nit,4);
F_plant = zeros(nit,1); P_plant = zeros(nit,1);

z_model = zeros(nit,4*n_PCE);       % stores model predictions
nu_model = zeros(4, n_PCE, nit);
fb_plant = zeros(nit,1);            % feedback error in nominal model
prediction

time_store = zeros(nit,1);
ef_plant = zeros(nit,1);
fval_plant = zeros(nit,1);
obj_plant = zeros(nit,5);

% initialise the control inputs
uig_k = [uig(1:p,1); uig(nit+1:nit+p,1)];
u = zeros(2*p,1);
umax = Fmax*ones(2*p,1);
du_max = Fmax;
description = 'Robust controller without cell death, kla disturbance,
bounded reference trajectory';
```

```
%% Plant parameters
% dist2 = [0*ones(nit/2,1);0*ones(nit/2,1)];
GUR_plant = (1 + alpha_p(1)*GUR_sig)*GUR_max*(1+0*disturbance2);
kla_plant = kla*(1+alpha_p(2)*kla_sig)*(1-disturbance);
OUR_plant = OUR_max*(1+0*OUR_sig*disturbance);
Ki_plant = Ki*(1 + 0*kla_sig)*(1-0*disturbance2);

for  k = 1:nit
    % check for prediction horizon
    if(k>1)
        if(p>nit+1-k)
            % receding horizon case
            p = nit+1-k;
            uig_k = [u(2:p+1,1); u(p+3:end)];% initial guess comes from
previous computed solution
            umax = Fmax*ones(2*p,1);
        else
            uig_k = [uig(k:k+p-1,1);uig(nit+k:nit+k+p-1,1)];
%             uig_k = u;
        end
    end
%     Xref_k = Xref(k:k+p-1,1).*Vref(k:k+p-1,1);      % Total Biomass
trajectory
%     Xactual_ref = Xref(k:k+p-1,1);                  % Biomass
concentration trajectory
%     uref_k = [uig(k:k+p-1,1); uig(nit+k:nit+k+p-1,1)];
% uref =  [Fref, Pref];

    XVref_lb_k = XVref_lb(k:k+p-1,1);
    XVref_ub_k = XVref_ub(k:k+p-1,1);

    A_cons = [tril(ones(p,p)) tril(-ones(p,p)); ... % Linear Constraint:
V(k+i)<=Vmax
            -tril(ones(p,p)) tril(ones(p,p))]; % Linear Constraint:
V(k+i)>=Vmin
    b_cons = [(Vmax - V0)/dt*ones(p,1); ...       % Linear Constraint:
V(k+i)<=Vmax
            (V0 - Vmin)/dt*ones(p,1)];            % Linear Constraint:
V(k+i)>=Vmin

    tstart = clock;
%     if (Vmax-V0<1e-6)
%         u = zeros(2*p,1);
%     else
%     end
    [u, fval, exitflag] = fmincon(@(ku)objfun7(ku, z0_model, V0, dt, p,
...
        A, c, kla, Km, GUR_max, OUR_max, GUR_sig, kla_sig, Ki, Ki_sig, ...
        Zgl_feed, n_dim, n_order, l, XVref_lb_k, XVref_ub_k, w, fb_k) ,
uig_k, ...
        A_cons, b_cons, [],[], zeros(2*p,1), umax,[],op);
%         u = ones(2*p,1); fval = 0; exitflag = -0.02;
```

```
       time_store(k,1) = etime(clock,tstart);
    % decide on the next control action
%     if(k>1)
%         [u, fval] = input_validation_2(u, fval, [u_plant(k-1,:)';
u_plant(nit+k-1,:)'], ...
%                 z0_model, V0, dt, p, A, c, kla, Km, GUR_max, OUR_max,
...
%                 GUR_sig, kla_sig, Ki, Ki_sig, Zgl_feed, n_dim,
n_order,...
%                 l, XVref_lb_k, XVref_ub_k, w, fb_k, nit);
%     end
    % objective function vector
    obj_plant(k,:) = objfun7_vec(u, z0_model, V0, dt, p, ...
        A, c, kla, Km, GUR_max, OUR_max, GUR_sig, kla_sig, Ki, Ki_sig, ...
        Zgl_feed, n_dim, n_order, l, XVref_lb_k, XVref_ub_k, w, fb_k);

    % plant dynamics
    % plant parameters
        GUR_p =  GUR_plant(k); kla_p = kla_plant(k);
        OUR_p = OUR_plant(k); Ki_p = Ki_plant(k);
    [y, nu] = plant_dynamics(z0_plant, V0, [u(1);u(p+1)], dt, A, c, ...
                kla_p, Km, GUR_p, OUR_p, Ki_p, Zgl_feed, 1);
%     y(4) = (1-beta*disturbance(k))*y(4); % loss of cell due to
Perfusion
    % run robust model dynamics for one time step
    [z0_model, nu_m] = robust_dynamics([u(1);u(p+1)], z0_model, V0, dt, 1,
...
        A, c, kla, Km, GUR_max, OUR_max, GUR_sig, kla_sig, Ki, Ki_sig,
Zgl_feed, ...
        n_dim, n_order, l);
    % reinitialise
    z0_plant = y(1:4)';
    V0 = y(5);
    fb_k = y(4) - z0_model(4);

    % save the results
    z_plant(k,:) = y(1:3);
    u_plant(k,1:p) = u(1:p)';
    u_plant(nit+k,1:p) = u(p+1:end)';
    F_plant(k,1:p) = u(1:p)';
    P_plant(k,1:p) = u(p+1:end)';
    V_plant(k,1) = y(5);
    y_plant(k,1) = y(4);
    nu_plant(k,:) = nu;
    ef_plant(k,1) = exitflag;
    fval_plant(k,1) = fval;

    z_model(k,:) = z0_model';
    nu_model(:,:,k) = nu_m;
    fb_plant(k,:) = fb_k;
```

```
    if(n_dim==1)
        if(n_order==2)
            z0_model(6)= 0 ; z0_model(10) = 0;
        else
            z0_model(6)= 0 ;
        end
    elseif(n_dim==2)
        if(n_order==1)
            z0_model(6)= 0 ; z0_model(10) = 0;
        end
    end

    save(filename);
end
%-------------------------------------------------------------------------------------------------------
```

% Robust Dynamics

```
%-----------------------------------------------------------------
function [z0_model_new, nu_m] = robust_dynamics(u, z0_model, V0, dt, p,
...
    A, c, kla, Km, GUR_max, OUR_max, Ki, GUR_sig, kla_sig, Ki_sig,
Zgl_feed, ...
    n_dim, n_order, l)
% robust model dynamics for one time step

nu_total = length(c);      % no. of fluxes
n_PCE = factorial(n_order + n_dim)/factorial(n_dim)/factorial(n_order);
%% make PCE for the nu and mu at every prediction horizon

nu = zeros(p,n_PCE,nu_total);
mu = zeros(p,n_PCE);
[nu, mu] = make_PCE(n_dim, n_order, l, z0_model, V0, u, dt, A, c, kla,...
            Km, GUR_max, OUR_max, Ki, Zgl_feed, p, GUR_sig, kla_sig,
Ki_sig);
%% propagate uncertainty in metabolite concentrations
% at the end determine PCE for biomass at every time step in the horizon
tspan = [0 dt];
z0 = [z0_model(1:4); V0; z0_model(5:end)];
% start the loop for dynamics
for k =1
    F0 = u(1); P0 = u(2);
    % Determine cell growth rate, mu
    mu_k = mu(k,:);

    % ODE solver parameters
    % options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
    Anu_gl = A(1,:)*squeeze(nu(k,:,:))';
    Anu_o2 = A(2,:)*squeeze(nu(k,:,:))';
    Anu_Ac = A(3,:)*squeeze(nu(k,:,:))';

    if (n_dim == 1)
        if(n_order ==1)
```

```
%             cs = 1;
            [t,z] = ode45(@(t,z) model_dynamics_1(t,z, kla, ...
                Anu_gl, Anu_o2, Anu_Ac, mu_k, Zgl_feed, F0, P0), tspan,
z0);
        else
%             cs = 2;
            [t,z] = ode45(@(t,z) model_dynamics_2(t,z, kla, ...
                Anu_gl, Anu_o2, Anu_Ac, mu_k, Zgl_feed, F0, P0), tspan,
z0);
        end
    elseif(n_order ==1)
%         cs = 3;
        [t,z] = ode45(@(t,z) model_dynamics_3(t,z, kla, kla_sig, ...
            Anu_gl, Anu_o2, Anu_Ac, mu_k, Zgl_feed, F0, P0), tspan, z0);
    else
%         cs = 4;
        [t,z] = ode45(@(t,z) model_dynamics_4(t,z, kla, ...
            Anu_gl, Anu_o2, Anu_Ac, mu_k, Zgl_feed, F0, P0), tspan, z0);
    end


%     z0 = z(end,:)';     % reinitialise

    if (z(end,1) <= 1e-6)     %check for O2 concentration
        z(end,1) = 1e-6;
    end
    if (z(end,2) <= 1e-6)     %check for O2 concentration
        z(end,2) = 1e-6;
    end
    if (z(end,3) <= 1e-6)     %check for O2 concentration
        z(end,3) = 1e-6;
    end
    % saving the results
    z0_model_new = [z(end,1:4) z(end,6:end)]';
end
nu_m = squeeze(nu(1,:,:))';
end
%-----------------------------------------------------------------
% Model Dynamics
%-----------------------------------------------------------------

function ydot = model_dynamics_1(t,z, kla, ...
            Anu_gl, Anu_o2, Anu_Ac, mu, Zgl_feed, F, P)

V = z(5); X0 = z(4); X1 = z(9);

ydot(1,1) = F/V*(Zgl_feed - z(1)) - Anu_gl(1)*X0 - Anu_gl(2)*X1;
ydot(6,1) = -F/V*z(6) - Anu_gl(2)*X0 - Anu_gl(1)*X1;

ydot(2,1) = kla*(0.21 - z(2)) - Anu_o2(1)*X0 - Anu_o2(2)*X1 - F/V*z(2);
ydot(7,1) = -kla*z(7) - Anu_o2(2)*X0 - Anu_o2(1)*X1 - F/V*z(7);
```

```
ydot(3,1) = -F/V*z(3) + Anu_Ac(1)*X0 + Anu_Ac(2)*X1;
ydot(8,1) = -F/V*z(8) + Anu_Ac(2)*X0 + Anu_Ac(1)*X1;

ydot(4,1) = mu(1)*X0 + mu(2)*X1 - (F-P)/V*X0;
ydot(9,1) = mu(2)*X0 + mu(1)*X1 - (F-P)/V*X1;

ydot(5,1) = F-P;
end
%-----------------------------------------------------------------------
% PCE model for the ODE's
%-----------------------------------------------------------------------

function [nu, mu] = make_PCE(n_dim, n_order, l, z0_model, V0, u, dt, A, c,
kla,...
            Km, GUR_max, OUR_max, Ki, Zgl_feed, p, GUR_sig, kla_sig,
Ki_sig)

%% Determine the Gauss Quadrature points
% l = level of accuracy
% n_dim = # of uncertainties, dimensions in PCE
% max n_dim = 2, max n_order = 2
[xi,w] = gausspoints(l, n_dim);
%% Run the plant for nit steps
num_points = length(w);
nu_total = length(c);
y_plant = zeros(p,nu_total);
nu_plant = zeros(p,nu_total,num_points);
if (n_dim<2)
    for i = 1:num_points
        GUR_p = GUR_max*(1+GUR_sig*xi(i,1));
        if(n_order == 1)
            z0(1,1) = z0_model(1) + z0_model(5)*xi(i,1);
            z0(2,1) = z0_model(2) + z0_model(6)*xi(i,1);
            z0(3,1) = z0_model(3) + z0_model(7)*xi(i,1);
            z0(4,1) = z0_model(4) + z0_model(8)*xi(i,1);
        else
            z0(1,1) = z0_model(1) + z0_model(5)*xi(i,1) + ...
                                        z0_model(9)*(xi(i,1)^2-1);
            z0(2,1) = z0_model(2) + z0_model(6)*xi(i,1) + ...
                                        z0_model(10)*(xi(i,1)^2-1);
            z0(3,1) = z0_model(3) + z0_model(7)*xi(i,1) + ...
                                        z0_model(11)*(xi(i,1)^2-1);
            z0(4,1) = z0_model(4) + z0_model(8)*xi(i,1) + ...
                                        z0_model(12)*(xi(i,1)^2-1);
        end
%         kla_p = OUR_max;
        [y_plant, nu_plant(:,:,i)] = plant_dynamics(z0, V0, u, dt, A, c,
...
            kla, Km, GUR_p, OUR_max, Ki, Zgl_feed, p);
    end
else
```

```
    for i = 1:num_points
        GUR_p = GUR_max*(1+GUR_sig*xi(i,1));
        kla_p = kla*(1+kla_sig*xi(i,2));
        if(n_order == 1)
            z0(1,1) = z0_model(1) + z0_model(5)*xi(i,1) +
z0_model(9)*xi(i,2);
            z0(2,1) = z0_model(2) + z0_model(6)*xi(i,1) +
z0_model(10)*xi(i,2);
            z0(3,1) = z0_model(3) + z0_model(7)*xi(i,1) +
z0_model(11)*xi(i,2);
            z0(4,1) = z0_model(4) + z0_model(8)*xi(i,1) +
z0_model(12)*xi(i,2);
        else
            z0(1,1) = z0_model(1) + z0_model(5)*xi(i,1) + ...
                                        z0_model(9)*(xi(i,1)^2-1);
            z0(2,1) = z0_model(2) + z0_model(6)*xi(i,1) + ...
                                        z0_model(10)*(xi(i,1)^2-1);
            z0(3,1) = z0_model(3) + z0_model(7)*xi(i,1) + ...
                                        z0_model(11)*(xi(i,1)^2-1);
            z0(4,1) = z0_model(4) + z0_model(8)*xi(i,1) + ...
                                        z0_model(12)*(xi(i,1)^2-1);
                                    % change this section when n_order=2,
                                    % n_dim = 2
        end
        [y_plant, nu_plant(:,:,i)] = plant_dynamics(z0, V0, u, dt, A, c,
...
            kla_p, Km, GUR_p, OUR_max, Ki, Zgl_feed, p);
    end
end
%% Determine PCE points for every nu at all the prediction horizon
n_PCE = factorial(n_order + n_dim)/factorial(n_dim)/factorial(n_order);
nu = zeros(p,n_PCE,nu_total); % structure reference
mu = zeros(p,n_PCE);
for i = 1:p
    for j = 1:nu_total
        nu(i,:,j) = a_PCE(n_order, n_dim, xi, w,
squeeze(nu_plant(i,j,:)));
    end
    mu(i,:) = c'*squeeze(nu(i,:,:))';
end
```

%----------

PCE-based Robust Optimization for Chapter 4

%----------

% Main function for model, optimization parameters

%----------

```
clear all
clc
% Initial Conditions
z0_model = [0.4 0.21 .2 0.001]';
```

```
V0 = 0.3;   % L, Initial Volume of the reactor, guess
Vmax = 0.4; % L, Final maximum batch volume, guess
Vmin = 0.20; % L, Final minimum batch volume, guess
% Fig = .1;  % L/h
Fmax = 0.3;   % L/h
Zgl_feed = 5;

% Model parameters
A = [0 9.46 9.84 19.23; 35 12.92 12.73 0; -39.43 0 1.24 12.12];
c = ones(4,1);
kla = 4;  % hr^-1, Mahadevan paper
Km = 0.015; % mM, Mahadevan paper
GUR_max = 6.5;   % mM/g-dw/hr, Mahadevan paper
OUR_max = 12;   % mM/g-dw/hr, Mahadevan paper
Ki = 1.0;

% Uncertainty Information
GUR_sig = 0.2;  % +/- 20%, guess
OUR_sig = 0.2;  % +/- 20%, guess
kla_sig = 0.2;
Km_sig = 0.2;
Ki_sig = 0.2;
% # of time steps and step size
nit = 110;
tend = 11;   % h, total time of cell culture growth, guess
dt = tend/nit;   % h, time, guess
number_of_inputs = tend;
% Initial guess for Feed rate
load('feed_rate_ig_nit_220_robust.mat');
uig = uig_discrete;
% uig = [(Vmax-V0)/dt/nit(1)*ones(nit(1),1); 1e-3*ones(nit,1)];
% uig = 0.02*ones(2*number_of_inputs,1);
% clear u
% umax = Fmax*ones(2*length_of_u,1);
umax = Fmax*ones(2*number_of_inputs,1);
%% PCE parameters
n_dim = 2;      % PCE dimensions (1 or 2), max is 2
n_order = 3;     % order of PCE, max is 3

%% Define the Gauss Quadratures parameters
% num_point, abscissae(xi), weights (w)
l = 5;  % level of accuracy, related to order of PCE l>n_order

%% Optimisation Parameters
op = optimset('fmincon');
op.Display = 'On';
op.TolFun = 1E-7;
op.TolX   = 1E-6;
op.MaxIter = 10000;
op.MaxFunEvals = 100000;
op.Algorithm = 'interior-point';
% op.LargeScale = 'on';
%% Linear Optimisation Constraints
```

```
% A_cons = [tril(ones(nit,nit)) tril(-ones(nit,nit)); ... % Linear
Constraint: V(tend)<=Vmax
%             -eye(nit,nit) eye(nit,nit) ];   % P<0.5*F at all time steps
% b_cons = [(Vmax - V0)/dt*ones(nit,1); ...       % Linear Constraint:
V(tend)<=Vmax
%             -1e-5*ones(nit,1)];                     % P<0.5*F at all
time steps
tol = 1e-5;
[A_cons, b_cons] = make_lin_cons(number_of_inputs, nit, Vmax, Vmin, V0,
dt, tol);
%% Call the optimisation
alpha_list = [0.03 0.034 .038];
u_list = zeros(2*nit,1);

matlabpool(2)

parfor i = 1:3
    alpha = alpha_list(i);
    filename = strcat('robust_zgl_5_time_11_x0_0p001_2d_HOT_alpha_',...
                num2str(alpha),'changed_Vol_constraints.mat');
    [y_plant, nu_plant] = call_robust_optmzn(z0_model, V0, dt, nit, ...
                A, c, kla, Km, GUR_max, OUR_max, Ki, GUR_sig, kla_sig,
Ki_sig, Zgl_feed, ...
                n_dim, n_order, l, alpha, uig, A_cons, b_cons,
number_of_inputs, ...
                umax, op, filename);
end
matlabpool close
end

%-------------------------------------------------------------------------
% Optimization function
%-------------------------------------------------------------


function [y_plant, nu_plant] = call_robust_optmzn(z0_model, V0, dt, nit,
...
        A, c, kla, Km, GUR_max, OUR_max, Ki, GUR_sig, kla_sig, Ki_sig,
Zgl_feed, ...
        n_dim, n_order, l, alpha, uig, A_cons, b_cons, number_of_inputs,
...
        umax, op, filename)

    tstart = clock;
    [u_discrete, fval, exitflag] = fmincon(@(ku) prob_cost(ku, z0_model,
V0,dt, nit, ...
        A, c, kla, Km, GUR_max, OUR_max, Ki, GUR_sig, kla_sig, Ki_sig,
Zgl_feed, ...
        n_dim, n_order, l, alpha), ...
        uig, A_cons, b_cons, [], [], zeros(2*number_of_inputs,1), ...
        umax,[], op);
```

```
    time_store = etime(clock,tstart);
    u = make_input(u_discrete, nit);
    save(filename);
    % check the pce coefficients for determined Gl feeding and perfusion
    % profile
    a = make_PCE(n_dim, n_order, l, z0_model, V0, u, dt, A, c, kla, Km,
GUR_max,...
        OUR_max, Ki, Zgl_feed, nit, GUR_sig, kla_sig, Ki_sig);

    % run nominal plant dynamics
    [y_plant, nu_plant] = plant_dynamics(z0_model, V0, u, dt, A, c, kla,
Km, GUR_max,...
        OUR_max, Ki, Zgl_feed, nit);
%     time_plant = linspace(dt,tend,nit)';
    description = 'Gl inhibition, robust optmzn obj is prob cost,
Perfusion can be more than feeding';
    % save the file
    save(filename);
%-----------------------------------------------------------------------
% Model Dynamics
%-----------------------------------------------------------------------

function [y_plant, nu_plant] = plant_dynamics(z0_model, V0, u, dt, A, c,
kla, Km, GUR_p,...
            OUR_p, Ki, Zgl_feed, nit)
% structure of z0_model = [zgl, zo2, zac, x]
% LP Model solution
n = length(c);
y_plant = zeros(nit,5);
nu_plant = zeros(nit,4);
for k = 1:nit
    F0 = u(k,1); P0 = u(nit+k,1);
    X_k = z0_model(4);
    if(length(kla)==nit)
        kla_p = kla(k);
    else
        kla_p = kla;
    end
    % Determine cell growth rate, mu
    % modified to reflect constrained dynamics
    Anew = [A; A(1:2,:)*X_k; -A(3,:)*X_k];
    b = [GUR_p*(z0_model(1)/(Km + z0_model(1) + (z0_model(1)^2)/Ki ));...
        OUR_p; ...
        100; ...
        F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt; ...
        kla_p*(0.21 - z0_model(2)) - F0/V0*z0_model(2) +
z0_model(2)/dt;...
        - F0/V0*z0_model(3) + z0_model(3)/dt; ];
    [nu, mu, ef0] = linprog(-c,Anew,b,[],[],zeros(n,1),[]);

    % integrate the metabolite concentrations

    % ODE solver parameters
```

```
    % options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
    if(ef0 == 1)
        Anu_gl = A(1,:)*nu; Anu_o2 = A(2,:)*nu; Anu_Ac = A(3,:)*nu;
    else
        mu = 0; Anu_gl = 0; Anu_o2 = 0; Anu_Ac = 0; nu = zeros(n,1);
    end
%     Anu_gl = A(1,:)*nu; Anu_o2 = A(2,:)*nu; Anu_Ac = A(3,:)*nu;

    tspan = [0 dt];
    z0 = [z0_model; V0];

    if (z0(1)<=0 && z0(2)<=0 && z0(3)<=0)
        z = [0 0 0 z0(4) V0];
    else
        [t,z] = ode45(@(t,z) model_dynamics(t,z, kla_p, Anu_gl, Anu_o2,
Anu_Ac, ...
            -mu, Zgl_feed, F0, P0), tspan, z0);
    end
    y = z(end,:)';

    if (z(end,1) <= 1e-6)     %check for O2 concentration
        y(1,1) = 1e-6;
    end
    if (z(end,2) <= 1e-6)     %check for O2 concentration
        y(2,1) = 1e-6;
    end
    if (z(end,3) <= 1e-6)     %check for O2 concentration
        y(3,1) = 1e-6;
    end
    % Reinitialise z0_model and V0 for next time step
    z0_model = y(1:4,1);
    V0 = y(5);

    % store the plant dynamics
    nu_plant(k,:) = nu';
    y_plant(k,:) = y';
end
%-------------------------------------------------------------
%----- make PCE coefficients----------------------------------------------------------------------
%-------------------------------------------------------------
function a = make_PCE(n_dim, n_order, l, z0_model, V0, u, dt, A, c, kla,
Km, GUR_max,...
            OUR_max, Ki, Zgl_feed, nit, GUR_sig, kla_sig, Ki_sig)

%% Determine the Gaus Quadrature points
% l = level of accuracy
% n_dim = # of uncertainties, dimensions in PCE
[xi,w] = gausspoints(l, n_dim);
%% Run the plant for nit steps
num_points = length(w);
X = zeros(num_points,1);
```

```
Ki_p = Ki;
if (n_dim<2)
    for i = 1:num_points
        GUR_p = GUR_max*(1+GUR_sig*xi(i,1));
%         kla_p = OUR_max;
        X(i) = plant(z0_model, V0, u, dt, A, c, kla, Km, GUR_p,...
            OUR_max, Ki_p, Zgl_feed, nit);
    end
else
    for i = 1:num_points
        GUR_p = GUR_max*(1+GUR_sig*xi(i,1));
        kla_p = kla*(1+kla_sig*xi(i,2));
        X(i) = plant(z0_model, V0, u, dt, A, c, kla_p, Km, GUR_p,...
            OUR_max, Ki_p, Zgl_feed, nit);
    end
end
%% Determine PCE points
a = a_PCE(n_order, n_dim, xi, w, X);
end
%-------------------------------------------------------------------------
% Robust Objective Function
```

```
function objfun = prob_cost(u_discrete, z0_model, V0,dt, nit, ...

    A, c, kla, Km, GUR_max, OUR_max, Ki, GUR_sig, kla_sig, Ki_sig,
Zgl_feed, ...
    n_dim, n_order, l, alpha)
%% Explaing all the parameters
%% Develop PCE coefficients
% u = zeros(nit,1);
u = make_input(u_discrete,nit);
a = make_PCE(n_dim, n_order, l, z0_model, V0, u, dt, A, c, kla, Km,
GUR_max,...
            OUR_max, Ki, Zgl_feed, nit, GUR_sig, kla_sig, Ki_sig);
a_nom = a(1);
a_sig = a(2:end,1)'*a(2:end,1);
F = u(1:nit,1);
P = u(nit+1:end,1);
V = (sum(F-P))*dt + V0;
% V = ones(1,nit)*u*dt + V0;
% alpha = 0.3;
objfun = a_sig*V^2/(V*a_nom - alpha)^2;
end
%-----------------------------------------------------------------
```

Distributed Robust MPC with communication loss for Chapter 5

```
%-----------------------------------------------------------------
Main Algorithm for Robust DMPC
```

```
clc
clear all
%% Models
Ts=3;   %min
[Ap, Bp, Cp, Dp] = reactor_separator(Ts);
```

```
% [Ap,Bp,Cp,Dp]= ssdata(plant_p); % Nominal state-space matrices for
discrete
%% Define vertices and uncertainty
% uncertainty in 9 states, plant divided into 3 subsystem
% each subsystem has uncertainty delta

dela_max = [0.15 0.3 0]';     % uncertainty in continuous systems 9states,
4 inputs
dela_min = [-0.5 -0.5 -1e-4]';

[Ap1, Ap2 Ap3 Ap4 Ap5 Ap6 Ap7 Ap8] = model_uncertainty(Ap, dela_max, ...
                        dela_min);
% Ap1 = 10*Ap; Ap2 = Ap; Ap3 = Ap; Ap4 = Ap;
% Ap5 = Ap; Ap6 = Ap; Ap7 = Ap; Ap8 = Ap;

Bp1 = Bp; Bp2 = Bp; Bp3 = Bp; Bp4 = Bp;
Bp5 = Bp; Bp6 = Bp; Bp7 = Bp; Bp8 = Bp;
% delta for the estimator
flag_robust_state = 1;    % robust estimator = 1 and non-Robust estimator
= 0;
delp_list = [ -0.1, -0.4, -0.1, -0.4, -0.4, -0.1, 0.3, -0.3; ...
              -0.4, -0.25, 0.25, 0.25, -0.4, -0.25, 0.3, 0.3; ...
             zeros(1,8)];
%              0.3, -0.3,-0.3, 0.3, 0.3,-0.3, -0.3, 0.3];     % choose
between delb_c_max and min

%% state weights
n_x = size(Ap,1); n_u = size(Bp,2);
Qs1 = 0.1*eye(n_x); Qs2 = eye(n_x); Qs3 = eye(n_x);
% input weights
R1=0.2; R2=0.5; R3 = 0.5;
% input constraints
% um1=1e6/12.6e5; um2=3/5.04; um3 = 1e6/13.32; um4 = 1e6/11.88;
um1 = 0.9; um2 = 0.9; um3 =0.9; um4 =0.9;
Xm1=um1^2; Xm2=diag([um2^2, um3^2]); Xm3 = um4^2;
Xm= {um1^2; diag([um2^2, um3^2]); um4^2};
%% Operating Conditions
%no. of sampling time
m=30;
%range of communication loss
l1 = 1; l2 = 5;
l3 = 7; l4 = 11;
l5 = 13; l6 = 17;
l7 = 19; l8 = 23;
l9 = 25; l10 = 29;
l11 = 31; l12 = 35;
l13 = 37; l14 = 54;
l15 = 57; l16 = 62;
l17 = 65; l18 = 514;
l19 = 561; l20 = 160;
l21 = 142; l22 = 145;
l23 = 146; l24 = 149;
l25 = 150; l26 = 153;
```

```
l27 = 154; l28 = 157;
l29 = 158; l30 = 161;

J_u = zeros(length(delp_list),1);
J_y = zeros(length(delp_list),1);

% for i = 1: 3
%% Initial Conditions

%-------------Values to be stored at each iteration
% cooperative cost function
%-------------------------------------------------
%--------Solver Parameters------------------------
ErrTol=5e-2;
MaxIteration=25;
tcito = zeros(m,1);
loss_flag = 0;  % no commn loss to start with
matlabpool(3)
% dmpc_func={@mpc11_ymip; @mpc22_ymip; @mpc33_ymip};
F_cell = cell(3,1);
xs = [0.865 0.1213 380.22 0.88 0.117 376.07 0.7505 0.2437 385.55]';
xk0_actual = [0.5 0.5 450 0.5 0.5 450 0.3 0.7 460]';
xk0 = xko_normalised(xk0_actual, xs);
%-----------------------------------------
%----Initialisation-----------------------------------
for i=2
%     xk0 = [-0.01; 0.01; -0.05; 0.01; -0.01 ;-0.05; 0.01; -0.01; -0.05];
% Initial Point, starting point for set-point change.
%     xk0 = -.3*ones(9,1);
    xk = xk0;
    gdata = zeros(1,m);
    tdata = zeros(1,m+1); tdata(1) = 0;
    xdata = zeros(n_x,m+1); xdata(:,1)=xk0;
    FFdata=zeros(4,n_x,m);
    udata=zeros(n_u,m);
    onclock = zeros(1,m);
    J= 0; %
    DD1=[];
    DD2=[];
    DD3=[];
    F1_old=zeros(1,n_x); F2_old=zeros(2,n_x); F3_old = zeros(1,n_x);
    F= zeros(4,9); gamma_old = zeros(3,1); gamma = zeros(3,1);
    x1k_old=xk; x2k_old=xk; x3k_old = xk;     %--Both the controllers
start from same point

    x1l=x1k_old(1,1); x1h=x1k_old(1,1); % Define the bounds on both the
states
    x2l=x1k_old(2,1); x2h=x1k_old(2,1); % states are x(1,1) and x(2,1);
    x3l=x1k_old(3,1); x3h=x1k_old(3,1);
    x4l=x1k_old(4,1); x4h=x1k_old(4,1);
    x5l=x1k_old(5,1); x5h=x1k_old(5,1);
    x6l=x1k_old(6,1); x6h=x1k_old(6,1);
    x7l=x1k_old(7,1); x7h=x1k_old(7,1);
```

```
    x8l=x1k_old(8,1); x8h=x1k_old(8,1);
    x9l=x1k_old(9,1); x9h=x1k_old(9,1);

    xbd=[x1h;x1l;x2h;x2l;x3h;x3l;x4h;x4l;x5h;x5l;x6h;x6l;x7h;x7l;...
                                    x8h;x8l;x9h;x9l];  % robust estimator
    xk_n = xk;      % nominal estimator
    %-------------------------------------------------
    D=[1;0;0;0];        % Step Input to the controller for x(1,1) state
    dt=5;
    FF=[];
    % Plant Definition
    delp = delp_list(:,i)    ;
    Apd = Ap*(eye(n_x,n_x) + blkdiag(delp(1)*eye(3,3), ...
                            delp(2)*eye(3,3), delp(3)*eye(3,3))) ;
    Bpd = Bp;
    for k=1:1:m
        k
        tstart = clock;
        % Bpd = Bn*[1+delp(1,k) 0; 0 1+delp(2,k)];
        % xbd
        if (((k>l1) && (k<l2)) || ((k>l3) && (k<l4)) || ((k>l5) &&
(k<l6))...
                || ((k>l7) && (k<l8)) || ((k>l9) && (k<l10)) ...
                || ((k>l11) && (k<l12)) || ((k>l13) && (k<l14)) ...
                || ((k>l15) && (k<l16)) || ((k>l17) && (k<l18)) ...
                || ((k>l19) && (k<l20)) || ((k>l21) && (k<l22)) ...
                || ((k>l23) && (k<l24)) || ((k>l25) && (k<l26)) ...
                || ((k>l27) && (k<l28)) || ((k>l29) && (k<l30)))
            loss_flag =1;
            %         xbdrhs=[xbd(1);-xbd(2);xbd(3);-xbd(4)];
            if (flag_robust_state ==1)
                xbd=bdmpc3(Ap, Bp, dela_max, dela_min, xbd, F);
%                   xbd = [0.5; 0.1;0.5;0.1;0.5; 0.1;0.5;0.1;0.5;0.11;...
%                       0.5;0.12;0.51;0.09; 0.48;0.13; 0.52;.08];
            else
                xk_n = Ap*xk_n + Bp*u;
                xbd = [xk_n(1,1) xk_n(1,1) xk_n(2,1) xk_n(2,1) ...
                    xk_n(3,1) xk_n(3,1) xk_n(4,1) xk_n(4,1) ...
                    xk_n(5,1) xk_n(5,1) xk_n(6,1) xk_n(6,1) ...
                    xk_n(7,1) xk_n(7,1) xk_n(8,1) xk_n(8,1) ...
                    xk_n(9,1) xk_n(9,1)]';
            end
        else
            xbd=[xk(1,1) xk(1,1) xk(2,1) xk(2,1) xk(3,1) xk(3,1) xk(4,1)
xk(4,1) ...
                xk(5,1) xk(5,1) xk(6,1) xk(6,1) xk(7,1) xk(7,1) xk(8,1)
xk(8,1) ...
                xk(9,1) xk(9,1)]';
            xk_n = xk;
            loss_flag = 0;
        end
        xbd;
        for iterations=1:MaxIteration
```

```
            iterations
            gamma = zeros(3,1);
            parfor n_sub = 1:3
%                 dmpc = dmpc_func(n_sub);
%                 Xm_temp = getfield(Xm, dmpc_list(n_sub));
%                 Xm_temp = cell2mat(Xm(n_sub));
                [F_temp, g_temp, QQ, YY]=dmpc(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6,
Ap7, Ap8, ...

Bp,R1,Qs1,x1k_old,F1_old,F2_old,F3_old,Xm,xbd,loss_flag, n_sub);
%                 [F2,g2,QQ2,YY2]=mpc22_ymip(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6,
Ap7, Ap8, ...
%
Bp,R1,Qs1,x2k_old,F1_old,F2_old,F3_old,Xm2,xbd,loss_flag);
%                 [F3,g3,QQ4,YY3]=mpc33_ymip(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6,
Ap7, Ap8, ...
%
Bp,R1,Qs1,x3k_old,F1_old,F2_old,F3_old,Xm3,xbd,loss_flag);
                F_cell(n_sub,1) = {F_temp};
                gamma(n_sub,1) = g_temp;
            end
            F1 = cell2mat(F_cell(1));
            F2 = cell2mat(F_cell(2));
            F3 = cell2mat(F_cell(3));
%             abs(norm([F1;F2;F3])-norm([F1_old;F2_old;F3_old]))
            abs(norm(gamma) - norm(gamma_old))
            if abs(norm(gamma) - norm(gamma_old))<= ErrTol
                %  [F1;F2];
                %  k;iterations; value=norm([F1;F2]-[F1_old;F2_old]);
                break;
            end
            F1_old = F1;
            F2_old = F2;
            F3_old = F3;
            gamma_old = gamma;
            F = [F1; F2; F3];
            DD1=[DD1 gamma(1)];
            DD2=[DD2 gamma(2)];
            DD3 = [DD3 gamma(3)];
        end
        %g1, g2
        %QQ1,QQ2
        %    iterations

        %if iterations == MaxIteration
        %    break;
        %end
%         F1_old = 0*F1;
%         F2_old = 0*F2;
%         F3_old = 0*F3;
%         F=[F1;F2;F3];
        FFdata(:,:,k)=F;
```

```
        if (((k>l1) && (k<l2)) || ((k>l3) && (k<l4)) || ((k>l5) &&
(k<l6))...
                || ((k>l7) && (k<l8)) || ((k>l9) && (k<l10)) ...
                || ((k>l11) && (k<l12)) || ((k>l13) && (k<l14)) ...
                || ((k>l15) && (k<l16)) || ((k>l17) && (k<l18)) ...
                || ((k>l19) && (k<l20)) || ((k>l21) && (k<l22)) ...
                || ((k>l23) && (k<l24)) || ((k>l25) && (k<l26)) ...
                || ((k>l27) && (k<l28)) || ((k>l29) && (k<l30)))
            %    xbdrhs=[xbd(1);-xbd(2);xbd(3);-xbd(4)];
            %    xbd=bdmpc(An, Bn, delb, xbd, F);
            %       xbd(3)-xbd(4);
            %    xbd(1)-xbd(2);
            u1=F(1,1:3)*xk(1:3,1) + ...
                F(1,4)*mean([xbd(7) xbd(8)]) + F(1,5)*mean([xbd(9)
xbd(10)])+ ...
                F(1,6)*mean([xbd(11) xbd(12)]) + F(1,7)*mean([xbd(13)
xbd(14)])+ ...
                F(1,8)*mean([xbd(15) xbd(16)]) + F(1,9)*mean([xbd(17)
xbd(18)]);

            u2=F(2,4:6)*xk(4:6,1) + ...
                F(2,1)*mean([xbd(1) xbd(2)]) + F(2,2)*mean([xbd(3)
xbd(4)])+ ...
                F(2,3)*mean([xbd(5) xbd(6)]) + F(2,7)*mean([xbd(13)
xbd(14)])+ ...
                F(2,8)*mean([xbd(15) xbd(16)]) + F(2,9)*mean([xbd(17)
xbd(18)]);
            u3=F(3,4:6)*xk(4:6,1) + ...
                F(3,1)*mean([xbd(1) xbd(2)]) + F(3,2)*mean([xbd(3)
xbd(4)])+ ...
                F(3,3)*mean([xbd(5) xbd(6)]) + F(3,7)*mean([xbd(13)
xbd(14)])+ ...
                F(3,8)*mean([xbd(15) xbd(16)]) + F(3,9)*mean([xbd(17)
xbd(18)]);
            u4=F(4,7:9)*xk(7:9,1) + ...
                F(4,1)*mean([xbd(1) xbd(2)]) + F(4,2)*mean([xbd(3)
xbd(4)])+ ...
                F(4,3)*mean([xbd(5) xbd(6)]) + F(4,4)*mean([xbd(7)
xbd(8)])+ ...
                F(4,5)*mean([xbd(9) xbd(10)]) + F(4,6)*mean([xbd(11)
xbd(12)]);
        else
            %        xbd=[xk(1,1) xk(1,1) xk(2,1) xk(2,1)]';
            u1=F(1,:)*xk;
            u2=F(2,:)*xk;
            u3=F(3,:)*xk;
            u4=F(4,:)*xk;
        end
        %u=F*xk;
        u=[u1;u2;u3;u4]
        udata(1:4,k)=u;
        if(k ==15)
```

```
            xk = xk0;
        else
            xk = Apd*xk+Bpd*u;
        end
        xdata(:,k+1)=xk;
        x1k_old=xk; x2k_old=xk ;  x3k_old = xk;   % reset the initial
condition of both controller

        %we must define xbdrhs for the purpose of defining lower bds for
%fmincon
        %in bdmpc

        J=J + xk'*Qs1*xk + u'*diag(R1*ones(4,1))*u;
        tdata(k+1)=k;
        tcito(k,1) = etime(clock,tstart);
        filename = strcat('w1_',num2str(Qs1(1,1)),'_w2_',num2str(R1),...
            '_robust_estimator_plant_new_', num2str(i),'_lp_3.mat');
        save (filename);
    end
    J_y(i) = sum(diag(diag(diag(xdata*xdata'))*Qs1));
    J_u(i) = J - J_y(i);
    save(filename);
end

matlabpool close
end
%-----------------------------------------------------------------
```

% Nominal Process Model for Reactor-Separator

```
%-----------------------------------------------------------------------
function [An,Bn,Cp,Dp] = reactor_separator(Ts)

% Two series reactors and separator
% process is linearised to determine State space model
%% Operating conditions
% States
xa1 = 0.865;
xb1 = 0.1213;
T1  = 380.22;    %K
xa2 = 0.88;
xb2 = 0.117;
T2  = 376.07;    %K
xa3 = 0.7505;
xb3 = 0.2437;
T3  = 385.55;    %K

% xa1 = y(1); xb1 = y(2); T1 = y(3);
% xa2 = y(4); xb2 = y(5); T2 = y(6);
% xa3 = y(7); xb3 = y(8); T3 = y(9);

% Inputs
Q1  = 10.8e5/3600;      %KJ/s
```

```
F20 = 5.04/3600 ;       %m3/s
Q2  = 11.4e5/3600;    %KJ/s
Q3  = 10.0e5/3600;     %KJ/s
%% Process parameters
F10 = 5.04/3600;    %m3/s
Fr  = 15.04/3600;   %m3/s
F1  = Fr+F10;       %m3/s
k1  = 2.77e3;       % s^-1
E1  = 5e4;      %KJ/kmol
T10 = 300;      % K
F2  = F20+F1;      % m3/s
k2  = 2.5e3;    %s^-1
E2  = 6e4;      %KJ/kmol
T20 = 300;      % K
Fp  = 5.04/3600;    %m3/s
V1  = 1.0;      %m3
V2  = 0.5;
V3  = 1.0;
rho = 1000;     %kg/m3
Cp  = 4.2;      %KJ/kg/K
dH1 = 6e4;      %KJ/kmol
dH2 = 7e4;      %KJ/kmol
Mw  = 250;
R   = 8.314;    %KJ/kmol/K
xa20    = 1;
xb20    = 0;
alpha_a = 3.5;
alpha_b = 1.0;
alpha_c = 0.5;
%% A nominal
% 9 states
% xa1, xb1, T1, xa2, xb2, T2, xa3, xb3, T3
% concentration of a, b & Temperature in each process unit
A = zeros(9,9);
alpha_D = (alpha_c + (alpha_a-alpha_c)*xa3 + (alpha_b - alpha_c)*xb3)^2;

%% Subsystem 1

A(1,1) = -F10/V1 - Fr/V1 - k1*exp(-E1/R/T1);
A(1,3) = -k1*E1/R/(T1^2)*xa1*exp(-E1/R/T1)*(T1/xa1);
A(1,7) = Fr/V1*alpha_a*(alpha_c + (alpha_b -
alpha_c*xb3))/alpha_D*(xa3/xa1);
A(1,8) = -Fr/V1*alpha_a*xa3*(alpha_b - alpha_c)/alpha_D*(xb3/xa1);

A(2,1) = k1*exp(-E1/R/T1)*(xa1/xb1);
A(2,2) = -F10/V1 - Fr/V1 - k2*exp(-E2/R/T1);
A(2,3) = (k1*xa1*E1/R/(T1^2)*exp(-E1/R/T1) - k2*xb1*E2/R/(T1^2)*exp(-
E2/R/T1))*(T1/xb1);
A(2,7) = -Fr/V1 * alpha_b*xb3*(alpha_c)/alpha_D*(xa3/xb1);
A(2,8) = Fr/V1*alpha_b*(alpha_c + (alpha_a -
alpha_c)*xa3)/alpha_D*(xb3/xb1);

A(3,1) = dH1/(Mw*Cp)*k1*exp(-E1/R/T1)*(xa1/T1);
```

```
A(3,2) = dH2/(Mw*Cp)*k2*exp(-E2/R/T1)*(xb1/T1);
A(3,3) = -F10/V1 - Fr/V1 + dH1*k1*xa1*E1/(Mw*Cp)/R/(T1^2)*exp(-E1/R/T1) +
...
           dH2*k2*xb1*E2/(Mw*Cp)/R/(T1^2)*exp(-E2/R/T1);
A(3,9) = Fr/V1*(T3/T1);
%-----------------------------------------------------------------
%% Subsystem 2
A(4,1) = F1/V2*(xa1/xa2);
A(4,4) = -F1/V2 -F20/V2-k1*exp(-E1/R/T2);
A(4,6) = -k1*xa2*E2/R/(T2^2)*exp(-E1/R/T2)*(T2/xa2);

A(5,2) = F1/V2*(xb1/xb2);
A(5,4) = k1*exp(-E1/R/T2)*(xa2/xb2);
A(5,5) = -F1/V2 - F20/V2 - k2*exp(-E2/R/T2);
A(5,6) = (k1*xa2*E1/R/(T2^2)*exp(-E1/R/T2) - k2*xb2*E2/R/(T2^2)*exp(-
E2/R/T2))*(T2/xb2);

A(6,3) = F1/V2*(T1/T2);
A(6,4) = dH1/(Mw*Cp)*k1*exp(-E1/R/T2)*(xa2/T2);
A(6,6) = -F1/V2 -F20/V2 + dH1/(Mw*Cp)*k1*xa2*E1/R/(T2^2)*exp(-E1/R/T2) +
...
                       dH2/(Mw*Cp)*k2*xb2*E2/R/(T2^2)*exp(-E2/R/T2);
%-----------------------------------------------------------------------
%% Subsystem 3
A(7,4) = F2/V3*(xa2/xa3);
A(7,7) = -F2/V3 + (Fr + Fp)/V3 - (Fr + Fp)/V3*alpha_a*(alpha_c + (alpha_b
- alpha_c)*xb3)/alpha_D;
A(7,8) = (Fr + Fp)/V3*alpha_a*xa3*(alpha_b - alpha_c)/alpha_D*(xb3/xa3);

A(8,5) = F2/V3*(xb2/xb3);
A(8,7) = (Fr + Fp)/V3*alpha_b*xb3*(alpha_a - alpha_c)/alpha_D*(xa3/xb3);
A(8,8) = -F2/V3 + (Fr+Fp)/V3 -(Fr + Fp)/V3*alpha_b*...
                   (alpha_c + (alpha_a - alpha_c)*xa3)/alpha_D;

A(9,6) = F2/V3*(T2/T3);
A(9,9) = -F2/V3;
%-----------------------------------------------------------------------
%% B matrix
B =  zeros(9,4);
B(3,1) = 1/rho/Cp/V1*(Q1/T1);
B(4,2) = (xa20 - xa2)/V2*(F20/xa2);
B(5,2) = (xb20 - xb2)/V2*(F20/xb2);
B(6,2) = (T20 - T2)/V2*(F20/T2);
B(6,3) = 1/rho/Cp/V2*(Q2/T2);
B(9,4) = 1/rho/Cp/V3*(Q3/T3);
%-----------------------------------------------------------------------
Acn = 60*A; Bcn = 60*B; C = eye(9,9); D = zeros(9,4);
%% Continuous to discrete
% plant_c = ss(Acn, Bcn, C,D);
plant_d      = c2d(ss(Acn,Bcn,C,D),Ts);
[An,Bn,Cp,Dp]= ssdata(plant_d);
end
```

```
%-------------------------------------------------------------
% Algorithm for Robust Observer
function [bd] = bdmpc3(An, Bn, dela_max, dela_min, xx, F)

%UNTITLED6 Summary of this function goes here
%   xx      : includes bounds on states x at previous time instant.
%   An, Bn  : Nominal model
%   dela_max, dela_min are upper and lower limit on uncertainty in An
%   Optimisation problem solved is as follows:
%   {An*[I+del] + Bn*F}* [x(1); x(2); ... ; x(9) ]
%   x = x(1); x(2); ... x(9);
%   st, x(1)l< x(1) < x(1)u
%   st, x(2)l< x(2) < x(2)u

%% Formulate the problem as bilinear problem
% Minimise Bilinear problem: a'x + y'Qx + b'y
%                            s.t. Ax =< c
%                            s.t. By =< d
% x represents state bounds, y represents B matrix uncertainties
del_l =
diag([dela_min(1)*ones(3,1);dela_min(2)*ones(3,1);dela_min(3)*ones(3,1)]);
% Lower bounds on dela for matrix An
% del_u = diag(delb_max);         % Upper bounds on delb for matrix B
x_l = [xx(2) xx(4) xx(6) xx(8) xx(10) xx(12) xx(14) xx(16) xx(18)]';   %
lower bound vector for states
x_u = [xx(1) xx(3) xx(5) xx(7) xx(9) xx(11) xx(13) xx(15) xx(17)]';   %
upper bound vector for states
%% define the new constraints
A = [eye(9,9); -eye(9,9)];
del_u_new = dela_max - dela_min;
if(norm(x_u - x_l)>1e-6)
    xu_new = x_u - x_l;
else
    xu_new = 1e-6*ones(9,1);
end
c = [xu_new; zeros(9,1)];
B = [eye(3,3); -eye(3,3)];
d = [del_u_new; zeros(3,1)];
%% Objective function parameters
constant = An*del_l*x_l + Bn*F*x_l;
a_temp  = An + Bn*F + An*del_l;
% b = zeros(3,1);
bd = zeros(18,1);
for i = 1:9
    a = a_temp(i,:)';
    b = [An(i,1:3)*x_l(1:3,1); ...
            An(i,4:6)*x_l(4:6,1); ...
            An(i,7:9)*x_l(7:9,1)];
    Q = zeros(9,3);
    Q(1:3,1) = An(i,1:3)'; Q(4:6,2) = An(i,4:6)'; Q(7:9,3) = An(i,7:9)';

    [x_temp, del_temp, xl_temp] = bilinear_yalmip(a, Q, b, A, B, c,d);
```

```
    delta_p = diag([del_temp(1)*ones(3,1); del_temp(2)*ones(3,1);
del_temp(3)*ones(3,1);]);
    xl_temp = (An*(eye(9,9) + del_l + delta_p)+Bn*F)*(x_temp + x_l);

    [x_temp, del_temp, xh_temp] = bilinear_yalmip(-a, -Q, -b, A, B, c, d);
    delta_p = diag([del_temp(1)*ones(3,1); del_temp(2)*ones(3,1);
del_temp(3)*ones(3,1);]);
    xh_temp = (An*(eye(9,9) + del_l + delta_p)+Bn*F)*(x_temp + x_l);
%     xh = -xh_temp - constant(i); xl = xl_temp + constant(i);
    bd(2*i-1:2*i,1) =  [xh_temp(i); xl_temp(i)];
end
%-------------------------------------------------------

% Bilinear Solver
function [x, y, z] = bilinear_yalmip(a, Q, b, A, B, c, d)
% Solves Bilinear problem of the form( min: a'*x + x'*Q*y + b'*y)
% s.t. Ax<= c, By<=d
% using Yalmip bmibnb solver
%% Yalmip variables
x = sdpvar(length(a),1);
y = sdpvar(length(b),1);

ineq = [A*x - c<=0, B*y - d <= 0];
obj = a'*x + x'*Q*y + b'*y;
options = sdpsettings('verbose',0,'solver','bmibnb');
solvesdp(ineq,obj,options);
x = double(x); y = double(y); z = double(obj);
end
%-------------------------------------------------------------
% Distributed Controller with LMI's
%-------------------------------------------------------------

function [F_temp, g_temp, QQ, YY]=dmpc(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6, Ap7,
Ap8, ...

Bp,R1,Qs1,x1k_old,F1_old,F2_old,F3_old,Xm,xbd,loss_flag, n_sub)

switch n_sub
    case 1
        Xm_temp = cell2mat(Xm(1));
        [F_temp, g_temp, QQ, YY] = mpc11_ymip(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6,
Ap7, Ap8, ...

Bp,R1,Qs1,x1k_old,F1_old,F2_old,F3_old,Xm_temp,xbd,loss_flag);
    case 2
        Xm_temp = cell2mat(Xm(2));
        [F_temp, g_temp, QQ, YY] = mpc22_ymip(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6,
Ap7, Ap8, ...

Bp,R1,Qs1,x1k_old,F1_old,F2_old,F3_old,Xm_temp,xbd,loss_flag);
```

```
    case 3
        Xm_temp = cell2mat(Xm(3));
        [F_temp, g_temp, QQ, YY] = mpc33_ymip(Ap1,Ap2, Ap3, Ap4, Ap5, Ap6,
Ap7, Ap8, ...

Bp,R1,Qs1,x1k_old,F1_old,F2_old,F3_old,Xm_temp,xbd,loss_flag);
end
%-------------------------------------------------------------
% DMPC for sub-system 1
function [F1,g1,QQ,YY]=mpc11_ymip(Am1, Am2, Am3, Am4, Am5, Am6, Am7,
Am8,...
                Bm1,R,Q1,xk,F1_old,F2,F3,Xm,xbd, loss_flag)
A1 = Am1+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A2 = Am2+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A3 = Am3+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A4 = Am4+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A5 = Am5+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A6 = Am6+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A7 = Am7+Bm1(:,2:3)*F2+Bm1(:,4)*F3;
A8 = Am8+Bm1(:,2:3)*F2+Bm1(:,4)*F3;

B1=Bm1(:,1); B2=Bm1(:,1); B3=Bm1(:,1); B4=Bm1(:,1);
B5=Bm1(:,1); B6=Bm1(:,1); B7=Bm1(:,1); B8=Bm1(:,1);

Q1=Q1+F2'*R*F2 + F3'*R*F3;

%Define LMIs
   gamma=sdpvar(1,1);
   Q = sdpvar(9,9);
   Y=sdpvar(1,9,'full');
%    ineq = [[1 xk'; xk Q] >= 0];
   ineq = [];
   ff = fullfact([2,2,2,2,2,2,2,2,2]);
   if(loss_flag == 1)
       for  i = 1:512
           xk_temp = [xbd(ff(i,1));xbd(2+ff(i,2)); xbd(4+ff(i,2)); ...
               xbd(6+ff(i,3)); xbd(8+ff(i,4)); xbd(10+ff(i,2));  ...
               xbd(12+ff(i,3)); xbd(14+ff(i,4)); xbd(16+ff(i,2))];
           ineq = [ineq, [1 xk_temp'; xk_temp Q]>=0];
       end
   else
       for  i = 1
       xk_temp = [xbd(ff(i,1));xbd(2+ff(i,2)); xbd(4+ff(i,2)); ...
                           xbd(6+ff(i,3)); xbd(8+ff(i,4));
xbd(10+ff(i,2));  ...
                           xbd(12+ff(i,3)); xbd(14+ff(i,4));
xbd(16+ff(i,2))];
       ineq = [ineq, [1 xk_temp'; xk_temp Q]>=0];
       end
   end
   ineq = [ineq, [Q Q*A1'+Y'*B1' Q*Q1^0.5 Y'*R^0.5; ...
                  A1*Q'+B1*Y  Q zeros(9,9)   zeros(9,1);...
                  (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,1); ...
```

```
                    (Y'*R^0.5)' zeros(1,9) zeros(1,9) gamma] >=0, ...
                    [Q Q*A3'+Y'*B3' Q*Q1^0.5 Y'*R^0.5; ...
                    A3*Q'+B3*Y  Q zeros(9,9)   zeros(9,1);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,1); ...
                    (Y'*R^0.5)' zeros(1,9) zeros(1,9) gamma] >=0, ...
                    [Q Q*A5'+Y'*B5' Q*Q1^0.5 Y'*R^0.5; ...
                    A5*Q'+B5*Y  Q zeros(9,9)   zeros(9,1);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,1); ...
                    (Y'*R^0.5)' zeros(1,9) zeros(1,9) gamma] >=0, ...
                    [Q Q*A6'+Y'*B6' Q*Q1^0.5 Y'*R^0.5; ...
                    A6*Q'+B6*Y  Q zeros(9,9)   zeros(9,1);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,1); ...
                    (Y'*R^0.5)' zeros(1,9) zeros(1,9) gamma] >=0, ...
                    [Xm -Y; -Y' Q]>=0];
   obj = gamma;
   ops = sdpsettings('solver','sedumi','sedumi.eps',1e-5,'verbose',0);
   solvesdp(ineq, obj, ops);
   g1 = double(gamma);  YY = double(Y); QQ = double(Q);
   F1=YY*QQ^(-1);
end

%-------------------------------------------------------------------------------------------------
```

% DMPC for sub-system 2

```
%-----------------------------------------------------------------
function [F2,g2,QQ,YY]=mpc22_ymip(Am1, Am2, Am3, Am4, Am5, Am6, Am7,
Am8,...
                Bm1,R,Q1,xk,F1,F2_old,F3,Xm,xbd,loss_flag)

A1 = Am1+Bm1(:,1)*F1+Bm1(:,4)*F3;
A2 = Am2+Bm1(:,1)*F1+Bm1(:,4)*F3;
A3 = Am3+Bm1(:,1)*F1+Bm1(:,4)*F3;
A4 = Am4+Bm1(:,1)*F1+Bm1(:,4)*F3;
A5 = Am5+Bm1(:,1)*F1+Bm1(:,4)*F3;
A6 = Am6+Bm1(:,1)*F1+Bm1(:,4)*F3;
A7 = Am7+Bm1(:,1)*F1+Bm1(:,4)*F3;
A8 = Am8+Bm1(:,1)*F1+Bm1(:,4)*F3;


B1=Bm1(:,2:3); B2=Bm1(:,2:3); B3 = Bm1(:,2:3); B4=Bm1(:,2:3);
B5 = Bm1(:,2:3); B6 = Bm1(:,2:3); B7 = Bm1(:,2:3); B8 = Bm1(:,2:3);

Q1=Q1+F1'*R*F1+F3'*R*F3;
%Define LMIs
   gamma=sdpvar(1,1);
   Q = sdpvar(9,9);
   Y=sdpvar(2,9,'full');
%    ineq = [[1 xk'; xk Q] >= 0];
   ineq = [];
   ff = fullfact([2,2,2,2,2,2,2,2,2]);
   if(loss_flag == 1)
       for  i = 1:512
```

```
            xk_temp = [xbd(ff(i,1));xbd(2+ff(i,2)); xbd(4+ff(i,2)); ...
                xbd(6+ff(i,3)); xbd(8+ff(i,4)); xbd(10+ff(i,2));  ...
                xbd(12+ff(i,3)); xbd(14+ff(i,4)); xbd(16+ff(i,2))];
            ineq = [ineq, [1 xk_temp'; xk_temp Q]>=0];
        end
    else
        for  i = 1
        xk_temp = [xbd(ff(i,1));xbd(2+ff(i,2)); xbd(4+ff(i,2)); ...
                            xbd(6+ff(i,3)); xbd(8+ff(i,4));
xbd(10+ff(i,2));  ...
                            xbd(12+ff(i,3)); xbd(14+ff(i,4));
xbd(16+ff(i,2))];
        ineq = [ineq, [1 xk_temp'; xk_temp Q]>=0];
        end
    end
    ineq = [ineq, [Q Q*A1'+Y'*B1' Q*Q1^0.5 Y'*R^0.5; ...
                    A1*Q'+B1*Y  Q zeros(9,9)   zeros(9,2);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,2); ...
                    (Y'*R^0.5)' zeros(2,9) zeros(2,9) gamma*eye(2,2)] >=0,
...
                    [Q Q*A3'+Y'*B3' Q*Q1^0.5 Y'*R^0.5; ...
                    A3*Q'+B3*Y  Q zeros(9,9)   zeros(9,2);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,2); ...
                    (Y'*R^0.5)' zeros(2,9) zeros(2,9) gamma*eye(2,2)] >=0,
...
                    [Q Q*A5'+Y'*B5' Q*Q1^0.5 Y'*R^0.5; ...
                    A5*Q'+B5*Y  Q zeros(9,9)   zeros(9,2);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,2); ...
                    (Y'*R^0.5)' zeros(2,9) zeros(2,9) gamma*eye(2,2)] >=0,
...
                    [Q Q*A6'+Y'*B6' Q*Q1^0.5 Y'*R^0.5; ...
                    A6*Q'+B6*Y  Q zeros(9,9)   zeros(9,2);...
                    (Q*Q1^0.5)' zeros(9,9) gamma*eye(9,9) zeros(9,2); ...
                    (Y'*R^0.5)' zeros(2,9) zeros(2,9) gamma*eye(2,2)] >=0,
...
                    [Xm -Y; -Y' Q]>=0];

    obj = gamma;
    ops = sdpsettings('solver','sedumi','sedumi.eps',1e-5,'verbose',0);
    solvesdp(ineq, obj, ops);
    g2 = double(gamma);  YY = double(Y); QQ = double(Q);
    F2=YY*QQ^(-1);
end
%-------------------------------------------------------------
```