Modeling and State Estimation of Bio-processes Using Dynamic Flux Balances

by

Xin Shen

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, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Radhakrishnan Mahadevan Professor, Dept. of Chem. Eng. & App. Chem., University of Toronto
Supervisor(s):	Hector Budman Professor, Dept. of Chemical Engineering, University of Waterloo
Internal Member:	William A. Anderson Professor, Dept. of Chemical Engineering, University of Waterloo
Internal-External Member:	Brian Ingalls Professor, Dept. of Applied Mathematics, University of Waterloo
Internal Member:	Luis Ricardez-Sandoval A. Professor, Dept. of Chemical Engineering, University of Waterloo

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 Contributions

Chapter 3 has been published as a paper in the journal computer & chemical engineering. The author developed and implemented the documented methodologies, obtained the numerical results, and completed the writing of the article. Hector Budman was involved in the conceptualization, supervision, and writing of the work. All authors read and approved the final manuscript.

Chapter 4 has been published as a special issue of bioreactor control in the journal Processes. The author developed and implemented the documented methodologies, obtained the numerical results, and completed the writing of the paper. Hector Budman was involved in the conceptualization, supervision, and writing of the work. All authors read and approved the final manuscript.

Chapter 5 has been published as a paper in the journal Computer & Chemical engineering. The author implemented the documented methodologies, obtained the numerical results, and completed the writing of the article. Hector Budman was involved in the supervision, conceptualization, and writing of the work. All authors read and approved the final manuscript.

Chapter 6 is based on a paper that has been submitted and is currently under review. The author developed and implemented the documented methodologies, obtained the numerical results, and completed the writing of the paper. Hector Budman was involved in the supervision, conceptualization, and writing of the work.

Chapter 7 is a summary of experiments and models developed by the author. The author developed the experimental setup and collected the data. Hector Budman was involved in the supervision and conceptualization of the work.

Abstract

Due to the increasing demand for bio-pharmaceuticals, optimization of bio-processes' productivity and reduction of process variability have become critical goals for manufacturers. Mathematical models of the fermentation processes are instrumental in achieving these goals.

Dynamic flux balance analysis (DFBA), sometimes also referred to as dynamic flux balance modeling (DFBM), is a type of mechanistic modeling approach that can describe the dynamic evolution of key metabolites based on the structure of metabolic networks. DFBA predicts the dynamic evolution of metabolites based on the assumption that resources are optimally allocated so as to maximize/minimize a biological objective function, e.g. maximization of cell growth. Accordingly, DFBA is formulated by a linear programming (LP) problem to compute the metabolic fluxes at each time interval. Then, the evolution of concentrations of different metabolites over time is obtained from the integration of mass balances that are based on the calculated fluxes.

Generally, the LP used to solve a DFBM for a particular microorganism may have multiple solutions. Mathematically, the multiplicity of solutions arises due to the underdeterminancy of the LP. On the other hand, from the biological point of view, the occurrence of multiple solutions may correctly describe the behavior of different strains of the same microorganism or alternatively the occurrence of metabolism switches under different operating conditions. The choice of one solution in the presence of multiplicity is further complicated by the fact that different commercial solvers may lead to different solutions of identical LPs. However, a good DFBA model should be solver-independent while it should be able to correctly describe available data for a specific microorganism strain.

Following the above a good LP solver should choose the specific solution based on the strain instead of choosing the solution "randomly" as most commercial solvers do. Hence, the first contribution of this research is to construct a solver that can select a specific solution among all possible optima that is compatible with experimental data. The weighted primal-dual method (WPDM) presented in Chapter 3, is a modified version of the interior point method (IPM) which uses interior weights to solve the LP. By manipulating these weights, the specific optimal solution can be obtained when multiple optimal solutions occur. The interior weights can be found by fitting experimental data obtained for a specific strain of a microorganism.

Although WPDM was able to select optimal solutions to fit the data, it was found to be computationally expensive and thus less suitable for large networks. To address this, an alternative fast and low-code algorithm called the ellipsoidal reflection method (ERM) was

developed as described in Chapter 6. This algorithm is able to select particular solutions among all possible solutions based on the combination of quadratic programming (QP) and LP problems. ERM plays the same role in DFBM but it can greatly reduce the computations thus making it suitable for future real-time applications.

An important application of mechanistic models such as DFBM in bioreactors is for the purpose of estimation of states that cannot be measured directly from available measurements. The ability of estimate variables such as growth rate, productivity or key nutrients are crucial for controlling and optimizing the process. State estimation for biochemical systems is particularly difficult due to the lack of online measurements in industrial bio-processes. While variables such as dissolved oxygen, temperature and pH are regularly measured and controlled, most metabolites' concentrations cannot be measured online. Thus, lack of observability of unmeasured states from measured ones are a known challenge in bio-processes.

To address the lack of observability, set membership estimation (SME) is proposed whereby the upper and lower bounds of each state are estimated based on limited measurements. This approach is motivated by the fact that the cell culture media recipe is generally fixed and the variations of the initial concentrations with respect to the nominal recipe are within small ranges. The SME treats the variation of initial concentrations as a set and propagates the initial bounds of the set onto the bounds of each metabolite at each time step. In this research, two methods of SME are proposed to estimate the bounds of metabolites.

The first state estimation method, described in chapter 4, is based on the identification of active constraints and assumes that the solution is always unique in DFBA. Since the concentration is varying with time, the LP problem in DFBA can be formulated as an LP with varying parameters. Then, Multiparametric linear programming (mpLP) can be used to convert the DFBA system into a variable structure system (VSS). VSS describes the system as composed of multiple subsystems where each subsystem describes a different region of the state space. For each subsystem, an extended Kalman filter (EKF) is constructed to estimate the key states, and the remaining states are estimated by SME. Moreover, the states crossing in or out of each region of the state space are monitored by a special algorithm and switches between different EKFs are determined accordingly. In the *E. coli* model, it was assumed that only biomass and culture volume are measured and are used to estimate the bounds of the other states.

The second state estimation method presented in chapter 5 is an extension of the first method but it explicitly considers the existence of multiple solutions. In this second method, WPDM is used to replace the LP solver in DFBA and multiparametric nonlinear

programming (mpNLP) is employed to solve the WPDM interior point-based algorithm. To propagate the uncertain sets by nonlinear mapping, the sets are split into smaller sets and are propagated separately by a linear mapping approximation. This is followed by an assembly operation of all these mapped sets together into one set for each state. Again, for the *E. coli* model, only biomass and culture volume are assumed to be measured and are used to estimate bounds on the other states. This method is shown to generate bounds of all states much faster than a Monte Carlo algorithm.

To test these methods proposed a platform of culturing *B. pertussis* has been set up. In chapter 7, a batch culture of *B. pertussis* and modeling by DFBM are presented. The protocols of shake flask, batch culture, and measurements of concentrations of amino acids in the culture by HPLC are set up. To solve the multiplicity issue, ERM is used in the modeling by DFBM. Based on the experimental data, DFBM adapted from the previous model is used to fit. The DFBM model can roughly capture the dynamics of key amino acids but not of all of them.

Acknowledgements

First of all, I would like to thank my thesis supervisor Professor Hector Budman. It is him that encourages me to challenge my knowledge and ability boundaries and surpass myself. Each time when I felt stuck at some difficult point in my research, he devoted lots of time to help me find a way out and to explore novel methods and innovative ideas. Each time, when we had different opinions about methods, we hotly debated them but he was forgiving about my occasional intense style of discussion. His passion and perseverance influenced me a lot in life and work.

Receiving sponsorship from Sanofi was very fortunate for my career and helped to achieve a dream since I was 20 years old. As a student majoring in pharmaceutical engineering for my undergraduate study, the cooperative research with Sanofi presented me with a precious opportunity to explore industrial problems. It is the first time that I feel that my knowledge and skills could serve to improve processes that will directly affect people's health and happiness. I would also like to thank Dr. Melih Tamer who provided me with very good guidance for my experiments so that I could set up a bioreactor experiment from scratch and build a fully operational system.

I would like to express my gratitude to Mariana Carvalho, Piyush Agarwal, Michael Vitelli, Ali Nikdel, Abhishek Mishra, and Charles Dal Castel. Mariana, Vitelli, Ali, and Abhishek helped me a lot with experiments, from HPLC to bioreactor, from shake flask experimentation and fluorescence spectroscopy. I also would like to thank Piyush for giving me different advice about modeling and data analysis. I also would like to thank Professor Valerie Ward and Marc G. Aucoin for letting me use their equipment.

During my four-year graduate studies and life, I received lots of useful suggestions, friendship, love, and help from different colleagues. Here I can only name a few. I would like to thank Yue Yuan, Tharun Subramanian, Honghao Zheng, Shuji Chang, Meghana Chepuru, Ittisak Promma, Mohammad Aghaee, Alex Vasile, Ali Ghodba, Zahra Negahban, and Arshia Fazeli.

I also would like to thank my friend Jiwu Huang and Yuxuan Zhou that providing important advice about HPLC. Cherishing the affection and warmth I received, I sincerely express my gratitude to Demin Yin, Haoyu Wu, Shuchen Liu, Yiming Zhu, Qingmin Zeng for your support.

Finally, I would like to thank my parents, I really miss my family!

Dedication

I dedicate this thesis to my grandparents and parents for their support and love. I love you all dearly.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing demand for bio-pharmaceuticals has motivated pharmaceutical companies to monitor and optimize their manufacturing processes. Mathematical models are crucial for implementing effective monitoring and optimization approaches. A key limitation for effective monitoring of bio-processes is a lack of analytical techniques for measuring key process variables in real time. Thus, model based estimation of key variables from available measurements is an attractive option for effective process monitoring. Despite recent breakthroughs in artificial intelligence (AI) based data-driven models, mechanistic modelling approaches are generally preferred due to their superior extrapolation ability as compared to AI models since the latter are mostly accurate only within a neighborhood of the training data. In the case of bio-processes, mechanistic modelling approaches use prior knowledge about the process such as the balance of energy and mass, the occurrence of particular metabolic reactions, kinetic rates of certain reactions etc.

Modeling approaches for bio-processes can be classified into 4 categories, segregated and structured models, unsegregated and structured models, segregated and unstructured models, and unsegregated and unstructured models [94, 61]. A model is segregated if the cells in the system have different states. For example, computational fluid mechanics models are segregated because these models assume cells are in a non-homogeneous environment so cells at different locations have different states (metabolites' concentrations) [75]. A model is structured if it captures the conversion among different metabolites based on biochemical or genomic information, or among different parts of the cells based on biological knowledge. For instance, a model describing the conversion of metabolites from glucose to intermediate metabolites and then to ATP is considered as structured because it describes known metabolic reactions among species. The most commonly used biochemical models for bioreactor is unsegregated and unstructured. These models assume all cells are

at the same nominal state and approximate the behavior of the bio-system based on the correlation between biomass and products with the substrates. The applications of these models for optimization, monitoring, and control of bioreactor operations are limited.

Dynamic flux balances models (DFBM) are based on the dynamic extension of flux balance analysis [64]. The resulting DFBM models are unsegregated and structured. From the knowledge of biochemistry and genome, key metabolic networks are modeled so that the conversion among metabolites in the models is structured. These models assume that cells optimally distribute resources of metabolites so as to maximize/minimize a biological objective such as the growth rate at each time step. Linear programming (LP) is commonly used to model the resulting optimization problem. The objective of the LP is the maximization of a biological objective, such as biomass growth rate, ATP or other [33]. The decision variables are the metabolic reaction rates referred to as metabolic fluxes. Since the stoichiometry of the metabolic networks, bounds of each reaction, and resources available to the cell at each time interval pose restrictions on the fluxes, they are modeled as constraints in the LP. By solving the LP at each time interval the fluxes can be obtained. Then, the metabolites' concentrations can be calculated at each time step based on mass balances as functions of the calculated fluxes. In this way, DFBM can capture the consumption or production of metabolites' concentrations over time.

In metabolic networks, a metabolite can be converted into another metabolite through one or more pathways, thus providing robustness and ability of cells to adapt and grow in different environments. Different strains of the same species may distribute resources differently across the metabolic network but still exhibit similar or same growth rate. Correspondingly, the metabolic models describing different strains of the same microorganism will exhibit multiple optimal solutions of their corresponding LP's. The occurrence of multiple optima of an LP is referred to as multiplicity which is widely observed in DFBM [59, 63, 76, 77, 96]. Geometrically, the set of multiple solutions is given by an optimal hyperplane where any solution on that hyperplane is an optimal solution. The choice of the solution obtained for the LP depends on the choice of the LP solver. Different versions of simplex and interior-point solvers can be used. Simplex methods will select solutions at verteces of the feasible region of decision variables whereas interior-point methods will select interior points on the optimal hyperplane. Usually, vertex points correspond to sparse solutions because more decision variables are zero. The trajectories of the metabolites' concentrations with respect to time depend on the solutions chosen by the solver at each time interval and thus they will be distinctively different for different solvers. Thus, the model becomes solver-dependent due to multiplicity.

When multiplicity exists, all optimal solutions are mathematically correct but not necessarily compatible with data. Thus, some of the optimal solutions do not correctly describe

the biological behaviour of the system. An ideal LP solver should find a unique solution compatible with experimental data and find different solutions for different metabolism. On the other hand, the solution is based on the key assumption that metabolism is efficient owing to natural evolution [83]. While this could be a sensible assumption for wild strains it may not be sufficient for describing engineered strains. A crucial novelty of this research is to use a novel interior-point based algorithm referred to as weighted primal-dual method (WPDM)[1] to select optimal solutions that are compatible with experimental data.

The primal-dual method is a type of interior-point method. By introducing interior-point weights into the primal-dual method, the weighted primal-dual method becomes a strictly convex form of the original LP. The optimal solution of proposed WPDM is unique and it leads to the same objective function value as the original LP. The minimum point of the strictly convex surface of the WPDM can be changed along the optimal hyperplane by varying the interior-point weights. Therefore, any optimal solution of the original LP can be obtained by using different interior-point weights. In practice, the interior-point weights can be found by fitting experimental data. Based on WPDM, the trajectories of metabolites can be forced to fit experimental data better than commonly used LP solvers.

While WPDM was used for most of the studies conducted in this research, it was found to be computationally expensive method thus making it unsuitable for very large metabolic networks or for potential online applications. The computational expense is related to the fact that the number of interior-point weights is the number of constraints. In most DFBMs, each flux has an upper bound and a lower bound. Besides these bounds, some extra constraints may be required. In total, the number of constraints is generally much larger than the number of decision variables (fluxes). For large metabolic problems with a large number of constraints, many corresponding interior-point weights are required which makes the computation heavy.

To address the computational expense of WPDM, an alternative method referred to as the ellipsoidal reflection method (ERM)[2] is proposed. ERM involves the combined solution of LP and QP problems to find the fluxes that fit experimental data. ERM first uses the LP solver to identify the optimal hyperplane and then it constructs a QP to select the best solution on that hyperplane. The QP problem defines the ellipsoidal contours of the convex quadratic objective. Then, these ellipsoidal contours are rotated by a reflection operation until they point and intersect the optimal hyperplane of multiple solutions at a solution that best fits the experimental data. The fitting performance is evaluated by the sum of

[1] Paper has been published. Shen, X., & Budman, H. (2020). A method for tackling primal multiplicity of solutions of dynamic flux balance models. Computers & Chemical Engineering, 143, 107070.

[2] A paper has been submitted to Computers & Chemical Engineering, which is currently under review.

square errors between model predictions and the data. A key advantage of this technique as compared to WPDM is that the number of decision variables is equal to the number of fluxes while WPDM is related to the number of constraints. Thus, the computation of ERM does not increase with the number of constraints as in WPDM. Also, since both the QP and LP are solved with commercial solvers, this method is low-code, fast, and accurate.

Bioprocess model-based monitoring is another key focus of the current research. Since bio-processes are typically lacking in terms of available online measurements, state estimation based on limited measurements is crucial for implementing online monitoring and optimization algorithms. Although process monitoring using unstructured biochemical models have been widely investigated in the literature, research on state estimation based on DFBM is limited. State estimation algorithms are motivated by the premise that if a feedback control can be designed based on the available measurements, the state estimation error can be controlled to zero. Therefore, a necessary condition to design a state observer is that enough measurements are available for observing (estimating) the unmeasured states. Different observability conditions has been proposed for different type of dynamic systems [110]. Observability tests for nonlinear systems can be classified as local observability and global observability tests[15]. Local observability assesses whether a point in state space can be estimated based on available measurements. Global observability assesses whether state can be estimated in the entire space. Since in industrial practice bioreactors need to be operated within a very narrow range of operating conditions, it is sufficient to test observability within that range.

Due to lack of online measurements it is very difficult or impossible to satisfy observability of all the states predicted by a DFBM model. The variables that are generally monitored online in bio-processes include optical density for biomass, pH, dissolved oxygen, aeration rate, agitation rate, and temperature. Most states of DFBM or other biochemical models are concentrations of metabolites which are very difficult to be measured online [4]. For instance, derivatizing agents are required to react with amino acids so that amino acids can be detected by fluorescence sensors when using high-performance liquid chromatography. This requirement generally rules out online monitoring of amino acids concentrations.

To overcome the observability problems, two types of set membership estimation methods (SME) are proposed in this research to estimate states based on DFBM with limited measurements. The proposed estimation approaches are based on the premise that an industrial bioprocess always uses the same media composition but the concentrations of media components may vary due to inaccuracy in media preparation or unknown variability in raw materials. On the other hand, based on experience and experiments, the variations in initial media components' levels can be generally quantified based on experience and a priori knowledge. Based on this knowledge, the problem can be formulated as a state

estimation problem in the presence of uncertainty in initial media composition and limited measurements where the latter can be used as feedback to reduce the uncertainty at each time step. Thus, these SME methods are designed to propagate the initial uncertainty over time onto the states such that the upper bounds and lower bounds of each state can be estimated.

Propagating the initial uncertainty directly by the DFBM model is very difficult because of its optimization-based nature. In DFBM the fluxes are solved at each time interval by an LP solver or WPDM and then are substituted into the state equations to calculate the states in the next time step. Multiparametric programming is introduced to address this problem that arises from the fact that at each time interval, a different set of constraints may become active [2]. If the varying states are regarded as parameters, the LP problem is converted into a multiparametric linear programming (mpLP) problem at each time step. MpLP can be solved offline so that explicit equations of optimal fluxes can be obtained and then substituted into the state equations.

To apply mpLP, the feasible parameter space is partitioned into different critical regions and then, explicit equations for calculating the fluxes can be obtained for each critical region. The biological interpretation of these critical regions is that cells have different metabolic behaviors that are represented by different regions of state space. As different regions have different equations of optimal fluxes and finally, a series of different state equations, the system comprised of sub-systems is referred to as a variable structure system (VSS). When the LP solution is unique, the application of mpLP is straightforward. However, when the LP solution is not unique, a WPDM with varying parameters must be solved to calculate fluxes. WPDM is especially modified for nonlinear programming so that multiparametric nonlinear programming (mpNLP) is used to deal with varying parameters instead of mpLP that is used for cases with unique solutions. The mpNLP proposed in this research divides the feasible parameter space into different zones and quadratic programming is then used to approximate the WPDM within the given zone. If the accuracy of the approximation is not sufficient, the mpNLP finds the best direction to divide the zones until the accuracy is satisfied or the zone is too small to be accounted for. In summary, by the use of either mpLP or mpNLP, the DFBM can be converted into VSS whether the solution is unique or not.

The first SME method[3] proposed in Chapter 4 is only applicable when the LP solution is unique so that mpLP is used to simplify DFBM into VSS. Once the states determining the fluxes can be estimated, the dynamic of DFBM can be properly described. It is shown

[3]Paper has been published in Shen, X., & Budman, H. (2021). Set Membership Estimation with Dynamic Flux Balance Models. Processes, 9(10), 1762.

that despite the existence of many critical regions only few of them are required to be considered in practical cell culture applications. Then, an approach is presented to find the minimum number of states so that fluxes for all these related critical regions can be estimated. Furthermore, by using extended Kalman filters (EKFs) for each critical region to estimate the observable states, the total number of states that need to be measured can be further reduced. Hence, SME is used to estimate unobservable states, and EKF is used to estimate observable states. As the fermentation continues, the state may enter from one critical region to another region. Since the observable states are different for different critical regions, an algorithm is deployed to detect switching states between regions.

The second SME method[4] proposed in Chapter 5 is applied when the solution is not unique. As mentioned above, the mpNLP can be used to solve off-line the WPDM with varying parameters so that DFBM can be simplified into a Variable Structure System (VSS). Inspired by the puzzle concept, the critical regions are disassembled into zones, bounds on states are propagated within each zone, and then the zones and corresponding bounds are assembled together and general bounds on states are calculated. The main differences between SME in chapter 5 from chapter 4 are the occurrence of a unique versus multiple solutions and the use of EKF for estimation of some states. The SME proposed in chapter 5 is a general method and has fewer requirements compared with SME in chapter 4. It considers the influence of multiplicity issue by WPDM with mpNLP. It does not require some states to be measured to construct the EKF. Therefore, this SME algorithm has greater applicability.

To test the algorithms proposed in this thesis, an experimental platform was developed. Since the research focuses on the metabolism of *B. pertussis*, a batch culture of *B. pertussis* was chosen for the experiments. The current platform can be also used for other similar cell lines or strains. The platform includes shake flasks for the seed, culture in a 2L bioreactor, and HPLC for measuring the amino acids' concentrations in the culture. The current protocol for batch culture of *B. pertussis* in 2L bioreactor is a scale-down model of Sanofi's fermentation step of the whooping cough vaccine manufacturing process. Thus, the results are expected to mimic their process for future process optimization and control. In this research, a successful batch culture of *B. pertussis* was conducted and its purity was verified by plating. The amino acids' concentrations at different times were measured by HPLC and used to calibrate a DFBM previously developed. While part of the amino acids were fitted well to the data, some could not be fitted. The main possible reason for the lack of fit is that the culture response was significantly different from the earlier Sanofi's experiments for which the DFBM was originally developed. The calibration of parameters

[4]Paper has been published. Shen, X., & Budman, H. (2022). Online estimation using dynamic flux balance model and multiparametric programming. Computers & Chemical Engineering, 164, 107872.

cannot compensate for the possible differences in model structure errors.

Following the above, this work presents the following novel contributions:

i. A novel WPDM method is introduced to solve the multiplicity issue of linear programming for dynamic flux balance analysis. Since WPDM is a data-driven method, it can be applied for engineered strains with more flexibility than a typical LP solver. The uniqueness and continuity of the solution by WPDM are proven.

ii. The mpLP and mpNLP methods are employed to simplify the LP and WPDM inside the DFBM so that the state equations can be converted into VSS and used with equations that are solved a priori off-line. Since each subsystem in VSS corresponds to different metabolic patterns, the method is useful for the identification of the metabolism bottleneck.

iii. Two types of SME tailored for DFBM are utilized to tackle the lack of observability and online measurements. Through the SME for DFBM, estimating the dynamic of metabolites with limited measurements is available so that model-based process control and model-based online optimization can be constructed.

iv. To tackle the multiplicity issue for large metabolic networks, the ERM method is proposed. This method overcomes the expensive computation of WPDM but still preserve the features of WPDM since it is data-drivenwhile ensuring uniqueness and continuity of the optimal solution. On the other hand, the WPDM method remains relevant because it served as the basis of the set-based model estimation method proposed in this thesis for the case that multiple solutions exist.

v. The development of a platform for culturing *B. pertussis* and quantification of amino acid concentrations in the culture is an important step to the application of these state estimation methods proposed. It can be used for future research and development of optimization and process control algorithms.

The thesis is organized as follows. Chapter 1 introduces the research and briefly describes each topic. Chapter 2 presents the background and a literature review. Chapter 3 presents the WPDM method for tackling the multiplicity issue of LP in DFBM. Chapter 4 introduces the SME with EKF to estimate concentrations based on the assumption of a unique solution in DFBM. Chapter 5 presents the SME approach for the case that the DFBM has multiple solutions. Chapter 6 introduces the ERM for model fitting when multiplicity occurs and the comparison of this method with WPDM. Chapter 7 presents the development of the experimental platform for the batch culture of *B. pertussis*.

Chapter 2

Theoretical Background and Literature Review

2.1 Dynamic Flux Balance Model

Dynamic modeling of the metabolism of a microorganism is challenging due to the complexity of biological systems and very limited experimental data. Assumptions regarding model structure and model segregation are often made based on the level of detail required for the model and the data available for model calibration. Accordingly, four types of models have been reported for describing cell cultures: segregated and structural, non-segregated and structural, segregated and unstructured, and non-segregated and unstructured [94, 61]. Segregation means that cells are treated individually within the cell population and the metabolic status of each cell is considered based on its environment. Structure means that the metabolic reaction networks, as determined from genome information, is used to define the interaction among different metabolites. For instance, the conversion of a specific substrate to a particular product is modeled as a series of reactions based on the genome in a structured model but it is modeled by only one macroscopic reaction in an unstructured model.

Sets of ordinary differential equations are commonly used to model non-segregated and unstructured models. The Monod equation is the most common type of equation to model kinetic rates in such models [94]. Non-segregated and unstructured models oversimplify the relationship between substrates, cells, and products so that these models are not suitable to explain the metabolism and to optimize the process. DFBM is a type of non-segregated and structured model that takes the structure of the metabolic network into consideration. It

assumes that cells can manipulate their metabolic fluxes to boost growth or other biological objectives. DFBM is always comprised of two parts: governing equations and optimization. There are two forms of DFBM [64] based on the use of a static optimization approach or a dynamic optimization approach. The latter involves a dynamic optimization over the entire time horizon of interest, e.g. the duration of a cell culture batch. Because of the higher computation expense of the dynamic optimization approach, the static optimization approach is more popular. In this approach the time domain of interest is divided into several time steps and a static optimization is conducted at each step. Hence, the fluxes solved by the DFBM are assumed to be piecewise constant along each sampling interval.

The basic DFBM combines a state space model with an LP as follows. The state space model as the function of metabolic fluxes $\boldsymbol{v_k}$ is defined in Eq. (2.1).

$$\boldsymbol{\psi_{k+1}} = \boldsymbol{\psi_k} + \Delta t \psi_{bio,k} \boldsymbol{S} \boldsymbol{v_k} \tag{2.1}$$

where subscript k indicates time step from 0, 1, $2 \cdots$ and Δt is the time step size. $\boldsymbol{\psi_k}$ is a state vector of n_ψ state variables at time step k, including biomass concentration $\psi_{bio,k}$. $\boldsymbol{S} \in \mathbb{R}^{n_\psi} \times \mathbb{R}^{n_r}$ is a matrix containing stoichiometric coefficients of all reactions involved in the metabolic network, where n_r is the number of reactions considered in the metabolic network.

The metabolic flux vector $\boldsymbol{v_k} \in \mathbb{R}^{n_r}$ is determined by a linear programming (LP) problem according to Eq. (2.2). At each time step, $\boldsymbol{v_k}$ is solved by the LP solver and substituted into Eq. (2.1) to obtain the state vector values at the next time step.

$$\min_{\boldsymbol{v_k}} \quad \boldsymbol{f}^T \boldsymbol{v_k} \tag{2.2a}$$

$$\text{subject to} \quad \boldsymbol{G} \boldsymbol{v_k} \leq \boldsymbol{g}(\boldsymbol{\psi_k}) \tag{2.2b}$$

$$\boldsymbol{F} \boldsymbol{v_k} = \boldsymbol{h}(\boldsymbol{\psi_k}) \tag{2.2c}$$

where the constant vector $\boldsymbol{f} \in \mathbb{R}^{n_r}$, the constant matrix $\boldsymbol{G} \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_r}$, the constant matrix $\boldsymbol{F} \in \mathbb{R}^{n_F} \times \mathbb{R}^{n_r}$, vector-valued function $\boldsymbol{g} \in \mathbb{R}^{n_G}$ of states $\boldsymbol{\psi_k}$, vector-valued function $\boldsymbol{h} \in \mathbb{R}^{n_G}$ of states $\boldsymbol{\psi_k}$. n_G is the number of inequality constraints and n_F is the number of equality constraints. Eq. (2.2a) denotes the objective function of the LP. The most commonly used objective is the maximization of biomass growth rate, or equivalently the minimization of its negative value, as shown in Eq. (2.2a) but other objectives have been also considered. Eqs. (2.2b) and (2.2c) describe balance equations and metabolic constraints such as charge balance, reaction rate bounds, and available nutrient bounds. Eq. (2.2) can be interpreted as the ability of the cells to regulate the metabolic fluxes $\boldsymbol{v_k}$ to boost growth at each time step.

2.2 Multiplicity in Dynamic Flux Balance Model

Due to a large number of reactions relative to the number of available constraints, the optimal solution to the linear programming problem defined in Eq. (2.2) is often not unique [59, 63, 76, 77, 96]. This phenomenon is referred to as multiplicity (or primal multiplicity). The multiplicity particularly hampers the potential application of DFBA models for estimation, optimization, and process control for three major reasons: i- the selection of different optima at each time interval generates different time trajectories for metabolite concentrations. Although these alternative trajectories satisfy the constraints of DFBA, only a few of them fit the experiments. ii- commercial LP algorithms generally do not consider the existence of multiple optima thus converging to an optimal solution according to chosen initial guesses or the hyper-parameters of the optimizer. The solution selected may not match the experimental data over long time horizons. iii- optimal solutions found by commonly used simplex methods are not continuous when the multiplicity exists. Thus, if the model is used for model-based control, disturbances can cause discontinuous jumps among solutions thus resulting in non-smooth trajectories.

A key question is whether specific flux distributions are more plausible than others. Schuetz and his colleagues found that in different bacterial wild types *E. coli* the fluxes are distributed in the near-optimal region of the Pareto surface of three objectives (maximum ATP yield, maximum biomass yield, and minimum sum of absolute fluxes) so that cells tend to achieve an optimal trade-off between metabolism and minimal flux adjustment to different conditions [93]. However, this approach cannot explain why specific flux distributions for different conditions occur at a specific region of the Pareto surface. More recently, the idea of minimal flux adjustment was further extended to consider a minimal enzyme cost by [83]. Based on this idea, a solution that satisfies a parsimonious enzyme usage FBA (pFBA) can be found from a bi-level optimization problem that searches fluxes with the lowest overall sum of fluxes [60, 47]. This method is particularly suitable for describing wild-type strains' behavior following the assumption that those strains must adapt to various environments with limited nutrients. On the other hand, engineered strains are selected or genetically modified for special operating conditions and productivity and thus they may have an inefficient metabolism so that the pFBA principle may not correctly capture the distribution of fluxes [98]. Hence, trying to fit data by applying the minimization of enzyme cost in the presence of primal multiplicity may require the use of additional kinetic constraints with many kinetic parameters that are difficult to obtain. Although other enzyme cost functions based on ideal assumptions that need fewer parameters could be used, e.g. the minimal total flux that assumes that all reactions need the same amount of enzyme, the prediction accuracy resulting from such approximations may be poor [83].

2.3 Methods Proposed for the Multiplicity Problem

To solve the primal multiplicity issue of DFBA, different methods were reported that are based on either auxiliary objectives or auxiliary rules. An efficient method for tackling multiplicity should have the following properties: i- the uniqueness of the optimal solution can be assured, ii- continuity of the optimal solution with respect to different possible perturbations should be proven, iii- flexibility to select a unique optimal solution that is consistent with experimental data, and iv- computational efficiency for large metabolic networks.

Auxiliary objectives based methods involve the use of extra objectives for optimization to reduce the feasible space. For example, minimal total (absolute) flux [47, 78, 83] or a minimal number of active reactions [78] were used as auxiliary objectives to exclude inefficient flux distributions, similar to the enzyme cost functions proposed by [83]. However, these objectives may not be suitable for cells that were engineered towards a particular purpose, e.g. for maximizing productivity. In the current study, it will be shown that the application of these auxiliary objectives cannot fit experimental data well in the case study of *Bordetella pertussis*. To further reduce the feasible space, hierarchical optimization, sometimes referred to as lexicographic optimization, was applied using a series of auxiliary biological objectives in priority order [36, 2, 46, 93]. To solve hierarchical optimization efficiently, equivalent weight method [2, 100] and Simplex-based method [42] have been proposed. Hierarchical optimization can theoretically select a specific vertex and non-vertex optimal solutions but finding a series of auxiliary objectives that assure the uniqueness of the optimal solution, that are consistent with experimental data, and that preserved continuity at the optimal solution is complex with this method. Auxiliary objectives used in hierarchical optimization can be chosen based on the user's prior knowledge about the microorganism. Inverse optimization [115] has been proposed as a more systematic way to find objectives. Given a vector of flux, constraints, and initial guesses of objective coefficients, inverse optimization can find the best objective coefficients that make the vector of flux optimal. This inverse optimization idea has been extended to find a quadratic form and non-parametric form objectives for flux balance analysis [115]. However, inverse optimization has not been studied as yet for dynamic flux models.

Auxiliary rules based methods use special rules to select the optimal solution when multiple optima occur. Smallbone and Simeonidis proposed two geometric methods to select the center of the solution hull as the true optimal flux distribution [102]. Although the center of the solution hulls is a unique flux distribution, there is no biological evidence to justify that the resulting solution provides a good fitting of available data. The lexicographic perturbation method is a modified simplex-based method [51] that was applied

to prevent dual degeneracy. However, this method can only find unique optimal solutions located at vertexes.

2.4 Lack of Measurements and Lack of Observability

For most industrial fermentation processes the available online measurements are very scarce. Commonly available online measurements include pH, dissolved oxygen, temperature, pressure, off-gas analysis, glucose, liquid level, foam amount, and weight. However, online measurements of concentrations of metabolites which are most informative about the process are very limited or not available altogether. Lack of measurements prevents the use of DFBMs for model-based monitoring or optimization.

To address the lack of online measurements soft sensors have been proposed. Soft sensors are algorithms that estimate the values of the states based on a few available online measurements. Data-driven soft sensors are currently very popular driven by the recent interest in the artificial intelligence research area. Reported data-driven soft sensors are generally based on artificial neural networks [97], support vector machines [73], partial least squares [85], genetic programming [53], principal component analysis [113] and fuzzy inference [52, 54]. These machine learning and artificial intelligence (AI) methods combined with different spectral-based devices have been used for inferring concentrations in cell cultures. For example, near-infrared spectrometer and Raman spectroscopy, have been applied to fermentation processes to estimate concentrations online [16]. However, despite their popularity, the main drawback of data-driven soft sensors is that their accuracy is limited to the region of data used for model training [24] and their prediction ability deteriorates due to the scarce data available for calibration [41]. Also, the lack of mechanistic information on the black box models based soft-sensors introduces concerns about the safety and reliability of controllers designed based on these sensors [41].

Another category of soft sensors is state observers based on mechanistic models such as the Luenberger observer, Kalman filter, particle filter, and moving horizon estimation [106]. These state observers estimate the values of some states based on the convergence of state prediction errors provided that sufficient measurements are available [4]. A key prerequisite of for designing these state observers is that observability can be satisfied with respect to the estimated states. State observers can only partially solve the lack of measurements because they still require a minimal number of measurements to satisfy observability conditions. It will be shown later in the manuscript that unless enough states of a DFBM model are measured online it is difficult to satisfy full observability for all the states. In contrast with state observers, an interval observer is a special type of

state observer that can estimate the bounds of states but some type of observability or convergence condition is still necessary [38] for their application. Some interval observers exploit the order-preserving properties of cooperative systems to estimate the bounds of states [28].

In the absence of observability of some states, instead of estimating their specific values, it is possible to estimate intervals (ranges) of values based on a priori known range of initial conditions, i.e. range of values at time = 0. This type of problem is referred to in the literature as an initial values problem with parameter uncertainty or set-valued ODE integration. The parameter here refers to either uncertain initial states or some model parameter such as a kinetic constant. To bypass the strict observability requirement, different set theory-based methods for estimation have been proposed, including interval analysis [49], Taylor models [65] and set membership estimation (SME) [95, 17, 3, 66]. All these methods are based on models that are used to propagate uncertainty in model parameters or/and in initial conditions to obtain sets of states' values at different times [9]. The main difference between these methods is in their approach to tightening the resulting bounds, e.g. interval analysis is based on a multidimensional interval. Interval analysis usually cannot capture the nonlinearity and correlation between different states which may result in divergence of estimates fast [49]. SME algorithms [95] are based on the use of convex sets and can capture nonlinearity and correlation between states but at the expense of less tight bounds as compared to Taylor models. SME has been applied to linear systems [17]. The propagation of uncertainty over time is performed by a series of affine mapping operations over sets. Different shapes of sets have been used to contain the uncertainty, including zonotopes [3], parallelotopes [17], and ellipsoids [66]. In contrast, Taylor models are based on Taylor expansions and bounds on remainders to obtain very tight nonconvex bounds [65]. However, because Taylor methods use non-convex bounds, it is difficult to exploit the available measurements although some computationally expensive relaxation methods have been proposed to achieve this goal [91]. Considering the limited measurements' availability and implementation convenience, this research focuses on a set-membership estimation approach.

2.5 Theoretical Background

2.5.1 Convex programming

Convex programming involves optimization problems where the objective function and feasible region are convex [12]. Let assume the objective function $f(\cdot)$ is a mapping from

$\mathbb{R}^n$ into $\mathbb{R}$, and $\theta \in [0,1]$. The objective function $f(\cdot)$ is convex if for any $\boldsymbol{x}$ and $\boldsymbol{y}$ in the domain of the objective function, $f(\theta\boldsymbol{x}+(1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x})+(1-\theta)f(\boldsymbol{y}))$ holds. Similarly, S is a convex set if for any element $x, y \in S$ and $\theta \in [0,1]$, $\theta x + (1-\theta)y \in S$. If the feasible region of the objective function is within the convex set, then the corresponding optimization problem is referred to as convex programming. A general convex programming is defined as Eq. (2.3).

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \tag{2.3a}$$

$$\text{subject to} \quad g_i(\boldsymbol{x}) \leq \boldsymbol{0} \qquad i = 1, ..., m \tag{2.3b}$$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.3c}$$

where the decision variable is $\boldsymbol{x} \in \mathbb{R}^n$; the objective function $f(\cdot)$ is a convex function; the inequality constraints $\boldsymbol{g_i}(\boldsymbol{x}) \leq \boldsymbol{0}, i = 1, ..., m$ are also convex functions; the equality constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ are affine transformation of $\boldsymbol{x}$; $\boldsymbol{A} \in \mathbb{R}^p \times \mathbb{R}^n$ with $rank(A) = p < n$. The inequality and equality constraints define a convex set.

Convex programming has been widely applied in different areas of application such as machine learning, data science, and control engineering [12]. Many problems can be expressed or transformed into convex programming problems. In this research, linear programming and quadratic programming are both convex programming problems that are considered in the studies. Nonlinear programming and multiparametric programming, both applied in this work, are also highly related to convex programming and thus convex programming is a main platform of this research.

Interior-point Method

The interior-point method is a commonly-used general method for solving convex programming problems [12, 29]. The KKT conditions for Eq. (2.3) are given in Eq. (2.4). Although there are different versions of interior-point methods, the core idea of these methods is to

solve the KKT conditions by the Newton's iterative method.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.4a}$$
$$g_i(\boldsymbol{x}) \leq \boldsymbol{0} \qquad i = 1, ..., m \tag{2.4b}$$
$$\boldsymbol{\lambda} \geq \boldsymbol{0} \tag{2.4c}$$
$$\nabla_{\boldsymbol{x}} f + \sum_{1}^{m} \lambda_i \nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) + \boldsymbol{A}^T \boldsymbol{\mu} = \boldsymbol{0} \tag{2.4d}$$
$$\lambda_i g_i(\boldsymbol{x}) = \boldsymbol{0} \qquad i = 1, ..., m \tag{2.4e}$$

where λ_i and μ_i are the Lagrange multipliers of inequality and equality constraints respectively. Eq. (2.4e) is referred to as the complementary slackness condition.

2.5.2 Linear programming

Linear programming problems is a special type of convex programming problems, where both constraints and the objective function are linear [29, 68]. A typical example of linear programming is defined in Eq. (2.5). Linear constraints include equality constraints as in Eq. (2.5c) and inequality constraints as in Eq. (2.5b). The linear constraints define a high-dimensional polyhedron in the space of the decision variables. The linear objective is convex function but not strictly convex and thus the optimal solution is not always unique.

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T \boldsymbol{x} \tag{2.5a}$$
$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{2.5b}$$
$$\boldsymbol{A_e}\boldsymbol{x} = \boldsymbol{b_e} \tag{2.5c}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a matrix with rank m ($m < n$); $\boldsymbol{b} \in \mathbb{R}^n$ and $\boldsymbol{c} \in \mathbb{R}^n$ are vector; $\boldsymbol{x} \in \mathbb{R}^n$ are decision variables including slack variables.

Simplex Method

For brevity, only the geometrical interpretation of the simplex method is presented. The linear constraints in the LP problem define a feasible region described by a convex polyhedron in the space of the decision variables. The direction of the vector $\boldsymbol{c}$ corresponds to the gradient of the objective function with respect to the decision variables. The Simplex

method evaluates the objective function at the vertexes along the edges of the polyhedron and iteratively seeks for the vertex with the smallest objective function value [6]. This search procedure is referred to as "pivoting" and it is repeated until all neighboring vertexes have a larger objective function as compared to the candidate vertex. The optimal vertex is referred to as a basic optimum. Although there are different versions of the simplex methods, the key idea of pivoting through different vertexes remains the same.

2.5.3 Quadratic programming

Quadratic programming (QP) is an optimization problem involving a quadratic objective function and linear constraints as per Eq. (2.6) [29, 74]. Without loss of generality, $\boldsymbol{Q}$ can always be converted into a symmetric matrix by $\boldsymbol{Q} = \frac{1}{2}(\boldsymbol{Q} + \boldsymbol{Q}^T)$. Since $\boldsymbol{Q}$ is the Hessian matrix of the objective it determines the existence of a unique or multiple solutions. If $\boldsymbol{Q}$ is positive definite or positive semidefinite, the objective is a convex function. If $\boldsymbol{Q}$ is positive definite, there is at most one solution. If $\boldsymbol{Q}$ is positive semidefinite, the number of solutions can be more than one. If $\boldsymbol{Q}$ is indefinite, the problem is no longer a convex problem and thus there could be more than one local minimum. Different algorithms can be used for solving quadratic programming problems, including the active set and interior-point methods.

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \tag{2.6a}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{2.6b}$$

$$\boldsymbol{A_e}\boldsymbol{x} = \boldsymbol{b_e} \tag{2.6c}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is a symmetric matrix; $\boldsymbol{c} \in \mathbb{R}^n$ and $\boldsymbol{x} \in \mathbb{R}^n$ are vectors; $\boldsymbol{A}$ and $\boldsymbol{A_e}$ are matrices at proper dimension; $\boldsymbol{b}$ and $\boldsymbol{b_e}$ are vectors.

2.5.4 Nonlinear programming

Nonlinear programming is a type of optimization problem involving a nonlinear objective function $f(\cdot)$ and nonlinear constraints $g_i(\cdot)$ and $h_j(\cdot)$, where $i = 1, \cdots, m$ and $j = 1, \cdots, p$ [12, 111]. It should be noticed that nonlinear programming can be non-convex. Therefore, many optima and local optima can exist. If different initial values are used, different solutions may be obtained. Commonly used algorithms for solving nonlinear programming problems are based on the solution of KKT conditions, differentiability, and constraint qualifications. However, since the nonlinear programming problem is not convex, KKT

conditions are merely necessary but not sufficient conditions for a solution to be optimal. Typical algorithms for nonlinear programming include interior-point method and sequential quadratic programming.

$$\begin{aligned}
&\min_{\boldsymbol{x}} && f(\boldsymbol{x}) && && (2.7a)\\
&\text{subject to} && g_i(\boldsymbol{x}) \leq \mathbf{0} && i = 1, ..., m && (2.7b)\\
&&& h_j(\boldsymbol{x}) = \mathbf{0} && j = 1, ..., p && (2.7c)
\end{aligned}$$

2.5.5 Multiparametric programming

Multiparametric programming is inspired by the concept of sensitivity analysis of the right hand side (RHS) of the constraints in an LP[10, 84]. When the RHS of the constraints involve multiple parameters and these parameters are within some sets, such an optimization problem can be tackled by multiparametric programming. In Eq. (2.8), $\boldsymbol{\theta}$ and $\boldsymbol{\theta_e}$ are parameters. $\boldsymbol{g}$ and $\boldsymbol{h}$ are vector-valued functions. S_1 and S_2 are sets containing parameters.

$$\begin{aligned}
&\min_{\boldsymbol{x}} && f(\boldsymbol{x}) && (2.8a)\\
&\text{subject to} && \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{\theta} && (2.8b)\\
&&& \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{\theta_e} && (2.8c)\\
&&& \boldsymbol{\theta} \in S_1 && (2.8d)\\
&&& \boldsymbol{\theta_e} \in S_2 && (2.8e)
\end{aligned}$$

When the objective and constraints of multiparametric programming are linear, the optimization is referred to as multiparametric linear programming (mpLP). QP or NLP problems can be tackled by multiparametric quadratic programming (mQP) or multiparametric nonlinear programming (mpNLP) respectively. In the multiparametric programming approach the parameters' set is partitioned into piece-wise continuous sets and for each of these sub-sets there are different expressions for calculating the optimal solution as a function of the parameters' values. Hence, multiparametric programming solves a priori the optimization problem and provides a look-up table of analytical expressions for calculating the optima for different sets. After determining the given set and substituting the parameters into expressions from the loop-up table, the optimal solution can be directly calculated. Methods of mpLP and mpQP are developed and have been applied to different

problems. Using mpLP, the DFBM can be simplified from a system containing an inner optimization problems into a variable structure system. However, most of the reported algorithms of mpLP ignore important issues such as the existence of multiple solutions. In contrast with mpLP and mpQP which do not involve any approximation, mpNLP is based on an approximation which makes it computationally expensive.

2.5.6 Observer and Observability

When a dynamic mechanistic model such as DFBM is available a model-based observer can be designed to estimate the states. The observer can be constructed to estimate the concentrations from a limited set of online measurements. Let's assume a general dynamic system described by Eq. (2.9a) and a measurement function given by Eq. (2.9b).

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \tag{2.9a}$$

$$\boldsymbol{y} = h(\boldsymbol{x}) \tag{2.9b}$$

where $\boldsymbol{x}$ is the states and $\boldsymbol{y}$ is the measurements. An observer is a dynamic system defined according to Eq. (2.10), such that, $\lim_{t\to+\infty} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| = 0$. The observer is an adjoint dynamic system of the system given in Eq. (2.9) to estimate $\boldsymbol{x}$ from a limited set of measurements $\boldsymbol{y}$ and the estimated state is $\hat{\boldsymbol{x}}$. $\boldsymbol{\xi}$ is the observer state of the observer. For very long time, the estimated value $\hat{\boldsymbol{x}}$ will converge to the true states $\boldsymbol{x}$.

$$\dot{\boldsymbol{\xi}} = \phi(\boldsymbol{\xi}, \boldsymbol{y}) \tag{2.10a}$$

$$\hat{\boldsymbol{x}} = h(\boldsymbol{\xi}, \boldsymbol{y}) \tag{2.10b}$$

The definition above is for an asymptotic observer for which the estimated value converges to the true state asymptotically. In contrast with the asymptotic observer, the parameters of some observers can be tuned to change the convergence speed. These observers are referred to as tunable observers [5]. If the system contains noise, the observer may give different estimates of states. Some observers can provide the optimal estimate and thus they are referred to as optimal observers. For example, the Kalman filter is an optimal observer that has been widely used in motion control. It considers prior knowledge of the process noise and measurement noise and based on this knowledge it calculates the optimal estimate of states. The nonlinear version of the Kalman filter is the Extended Kalman Filter (EKF).

Some measurements are necessary to construct the observer. Observability is a necessary condition to construct a tunable observer [5]. For a n dimensional nonlinear system

defined in Eq. (2.9), the local observability condition at $\boldsymbol{x_0}$ is Eq. (2.11) [88].

$$\begin{bmatrix} h(\boldsymbol{x_0}) \\ L_f h(\boldsymbol{x_0}) \\ \cdots \\ L_f^{n-1} h(\boldsymbol{x_0}) \end{bmatrix} = n \tag{2.11}$$

where $L_f h$ is first order Lie derivative of h with respect to f.

2.5.7 Set Theory

Set Operations

In topology theory, a connected space is a topological space that cannot be represented as the union of two or more disjoint non-empty open subsets [101]. A subset of a topological space S is a connected set if it is a connected space when viewed as a subspace of S [101]. The discussions of sets in this research are limited to connected sets.

Vector operations, like summation and multiplication, can be extended from vectors to sets. Set operations are operations of sets and all elements in the sets are considered. In contrast with vector operations that focus on individual elements of the vector, set operations focus on populations. Commonly used set operations, include linear mapping, projection, translation, Minkowski addition, intersection, union, lifting, and outer approximation.

Set Membership Estimation

If the initial state $\boldsymbol{x_0}$ of the system in Eq. (2.9a) is a set X, the set propagation describes how the set $X(t)$ evolves with time. To calculate the set propagation, set operations are required. By using these set operations, a larger set varying with time can be constructed to contain the propagated set. Such a larger set can be interpreted as an estimate of the boundary of the state. Instead of estimating the optimal value of states varying with time, methods to estimate sets of states are referred to as set membership estimation [9]. By projecting the sets onto each dimension, the bounds of states can be estimated. The difficulty of set membership estimation is that the set of states of nonlinear systems can diverge. Different set membership methods and different shapes of sets have been proposed [9, 17, 3, 66].

Chapter 3

A Method for Tackling Primal Multiplicity of Solutions of Dynamic Flux Balance Models

3.1 Overview

[1] A method is presented to tackle primal multiplicity of Dynamic flux balance analysis (DFBA) which is a Linear Programming (LP) based modeling approach that assumes that the cell distributes fluxes such as to maximize a specific biological objective. When the LP problem has multiple optima, the LP solvers usually only report the first optimum that it is reached which may not fit well the experimental data. To tackle this primal multiplicity problem, the weighted primal-dual method with auxiliary parameters is used to calculate a unique time trajectory for a given set of initial conditions. Through tuning of these auxiliary parameters, a unique optimal solution can be obtained and calibrated to fit available experimental data. Beyond its capability to tackle multiplicity, the algorithm is shown to significantly improve the prediction of some metabolites in a case study of the fed-batch fermentation of *Bordetella pertussis*.

[1] Adapted from Shen, X., & Budman, H. (2020). A method for tackling primal multiplicity of solutions of dynamic flux balance models. Computers & Chemical Engineering, 143, 107070.

3.2 Introduction

Quantitative methods to analyze metabolic processes occurring in microorganisms are crucial for the improvement and optimization of bio-processes. Flux balance analysis (FBA) refers to a steady-state modeling approach of genome-based metabolism [108]. FBA is formulated as a linear programming problem involving the maximization of a biological objective function with respect to the flux distribution across the metabolic network. However, FBA models are often under-determined thus resulting in primal multiplicity. Researchers have proposed different algorithms to enumerate all vertex optima [59, 63, 76, 77, 96]. To deal with dynamic bio-processes, researchers have successfully extended FBA to account for dynamic behavior by using dynamic flux balance analysis (DFBA) [64, 86, 109]. DFBA models describe the dynamic optimal flux distributions as a time sequence of LP problems. Concentration-dependent rate constraints have been proposed to further regulate the dynamic behavior [69, 81, 82]. However, the primal multiplicity issue still remains due to the underdeterminancy of the LP formulations.

The Weighted primal-dual method (WPDM) used in the current study to tackle multiplicity is a modified interior-point method based on auxiliary parameters. It was originally proposed in the late 1980s to speed-up computation [30, 72]. In this research, the WPDM is applied for the first time to address the primal multiplicity problem in DFBA models. WPDM approximates the LP by a strictly convex optimization problem with a set of auxiliary parameters to control which optimal solution is selected. In the calibration of the DFBA model, these auxiliary parameters are obtained by fitting of experimental data. Pointwise approximation, uniqueness, and continuity are proved mathematically in the paper. We show in the case study of *Bordetella pertussis* that the prediction accuracy can be significantly improved as compared with other methods used in this investigation. WPDM is data-driven and the auxiliary parameters can be found by fitting experimental data when prior knowledge of the strain is not readily available. Two potential drawbacks for WPDM are approximation error and computational expense. Being a type of interior-point method WPDM finds a solution that is close to an actual optimum. However, WPDM can have the same accuracy as a Simplex method if a sufficiently small penalty parameter is used. The computational expense of WPDM is potentially high since it is based on the solution of a set of algebraic equations by a Newton's method. However, after the weights are identified the computation for online implementation is fast as shown in the case study.

The paper is organized as follows. Section 2 presents background on dynamic flux balance analysis, different methods for primal multiplicity and model calibration. Section 3 presents the main properties of of the WPDM method and related mathematical proofs. Section 4 presents results of the application of WPDM to the modeling of a fed-batch

culture of *B. pertussis* and a comparison of WPDM to different methods reviewed above that were used to address multiplicity. Section 5 presents the conclusions.

3.3 Methods

3.3.1 Dynamic Flux Balance Analysis

Dynamic flux balance analysis (DFBA) is a dynamic extension of steady-state flux balance analysis (FBA). In both FBA and DFBA, cells are regarded as agents that optimally distribute metabolic fluxes to maximize a specific biological objective. There are two types of DFBA models reported in literature. In the first type of models the metabolites are described in intracellular and extracellular compartments and exchange fluxes are used to relate concentrations among compartments [46]. Using the assumption that internal reactions are generally much faster than cell growth, all intracellular metabolite concentrations are assumed constant. Continuous dynamic mass balances of extracellular concentrations are used to describe the dynamic evolution of the culture. It has been argued that intracellular metabolite concentrations are not constant and may change over time [34]. To address this limitation the second type of DFBA models [64], used in the current study, considers both intracellular and extracellular dynamic changes of concentrations. The concentrations are assumed constant only during a single time interval of discretization and discrete mass balance equations are used to describe the evolution of both intracellular and extracellular metabolites over time. In this second type of model kinetic rate expressions are used as upper bound of limiting reactions. Mass balance based constraints are used to ensure positivity of concentrations.

The DFBA model formulation used in this work is defined in Eq. (3.1) in canonical form. The evolution of metabolite concentrations with time is calculated by solving a series of LP problems posed in canonical form over consecutive time intervals. It should be noticed that in this type of DFBA model, for a given time step k, there are no equality constraints since Eq. (3.2a) and Eq. (3.2b) are only used outside of the LP problem to update the states for the solution of next time interval.

$$\max_{\boldsymbol{v_k}} \quad \boldsymbol{d}^T \boldsymbol{v_k} \tag{3.1a}$$

$$\text{subject to} \quad \boldsymbol{f}(\boldsymbol{\psi_{k-1}}) \leq \boldsymbol{S}\boldsymbol{v_k} \leq \boldsymbol{g}(\boldsymbol{\psi_{k-1}}) \tag{3.1b}$$

$$\boldsymbol{\psi_{k-1}} + h_{k-1}\Delta t \boldsymbol{S}\boldsymbol{v_k} \geq 0 \tag{3.1c}$$

$$\boldsymbol{0} \leq \boldsymbol{v_k} \leq \boldsymbol{v_{max}} \tag{3.1d}$$

where sampling time instants are $t_0, \cdots, t_{n_t}$ and $t_i - t_{i-1} = \Delta t$ for $\forall i = 1, \cdots, n_t$; $k = 0, 1, \cdots, n_t$ is the corresponding index of the sampling time interval; n_r is the number of reactions; n_m is the number of metabolites (excluding biomass) considered in the biological network; $\boldsymbol{d} \in \mathbb{R}^{n_r}$ is the vector of coefficients describing the relative contributions of different metabolites to growth rate; $\boldsymbol{v_k} \in \mathbb{R}^{n_r}$ is a flux vector at time interval k which upper bound is $\boldsymbol{v_{max}} \in \mathbb{R}^{n_r}$ and fluxes corresponding to reversible reactions are treated as two independent fluxes; $\boldsymbol{\psi_k} \in \mathbb{R}^{n_m}$ is the vector of metabolites' concentrations at time interval k; function $\boldsymbol{f}(\boldsymbol{v_k}) \in \mathbb{R}^{n_m}$ and function $\boldsymbol{g}(\boldsymbol{v_k}) \in \mathbb{R}^{n_m}$ are upper and lower bounds given as functions of the corresponding metabolite concentrations and these may represent kinetic rate or other biological constraints; h_k is the biomass concentration at time interval k; $\boldsymbol{S} \in \mathbb{R}^{n_m \times n_r}$ is the stoichiometry coefficients' matrix of the metabolic network. Eq. (3.1c) are constraints for ensuring that concentrations are positive.

Eqs. (3.2a) and (3.2b) are discrete mass balance based state equations describing the evolution of metabolite and biomass concentrations over time. After fluxes $\boldsymbol{v_{k-1}}$ are obtained by solving LP as Eq. (3.1), concentrations of biomass and metabolites at time step k can be obtained.

$$\boldsymbol{\psi_k} = \boldsymbol{\psi_{k-1}} + h_{k-1}\Delta t \boldsymbol{S} \boldsymbol{v_k} \tag{3.2a}$$

$$h_k = h_{k-1} + h_{k-1}\Delta t \boldsymbol{d^T} \boldsymbol{v_k} \tag{3.2b}$$

For clarity, both the canonical form and standard form of the LP are given. While the canonical formulation of DFBA as Eq.(3.1) is suitable for describing the biological meaning of the model, the standard form is more convenient for describing the optimization solution and the proofs. The main difference between the canonical form and the standard form is in the inequality constraints. After adding non-negative slack variables to convert the inequality constraints into equality constraints [6] the standard form is defined in Eq. (3.3).

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{b} \\ & \boldsymbol{z} \geq \boldsymbol{0} \quad (-\boldsymbol{I}\boldsymbol{z} \leq \boldsymbol{0}) \end{aligned} \tag{3.3}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a matrix with rank m ($m < n$); $\boldsymbol{b} \in \mathbb{R}^m$ and $\boldsymbol{c} \in \mathbb{R}^n$ are vector; $\boldsymbol{x} \in \mathbb{R}^n$ are decision variables; $\boldsymbol{z} \in \mathbb{R}^m$ are slack variables; $\boldsymbol{I}$ is identity matrix at proper dimension. Using slack variables, the standard form of the DFBA model is in the following Eq. (3.4).

$$\max_{\boldsymbol{v_k},\ \boldsymbol{\xi}} \boldsymbol{d}^T \boldsymbol{v_k} + \mathbf{0}\boldsymbol{\xi} \tag{3.4a}$$

subject to

$$\begin{bmatrix} -\boldsymbol{S} \\ \boldsymbol{S} \\ -h_{k-1}\Delta t \boldsymbol{S} \\ \boldsymbol{I} \\ -\boldsymbol{I} \end{bmatrix} \boldsymbol{v_k} + \boldsymbol{\xi} = \begin{bmatrix} -\boldsymbol{f}(\boldsymbol{\psi_{k-1}}) \\ \boldsymbol{g}(\boldsymbol{\psi_{k-1}}) \\ \boldsymbol{\psi_{k-1}} \\ \boldsymbol{v_{max}} \\ \mathbf{0} \end{bmatrix} \tag{3.4b}$$

$$\boldsymbol{\xi} \geq \mathbf{0} \qquad (-\boldsymbol{I}\boldsymbol{\xi} \leq \mathbf{0}) \tag{3.4c}$$

where Eqs. (3.1b)-(3.1d) are converted to equality constraints as Eqs. (3.4b) by adding slack variables $\boldsymbol{\xi}$ in proper dimension. Each element of slack variables $\boldsymbol{\xi}$ corresponds to an inequality constraint in Eq. (3.1). $\boldsymbol{v_k}$ corresponds to decision variable $\boldsymbol{x}$ in Eq. (3.3).

Experimental Data

In this paper, the experimental data and DFBA models of fed-batch fermentation of *B. pertussis* reported in paper [13] were used to calibrate the model and for comparing different solvers and different methods to address multiplicity. 17 amino acids are required for *B. pertussis* biomass synthesis but the main limiting substrate and carbon source is glutamate. Most amino acids can be biosythesized except histidine, methionine, phenylalanine, and tryptophan. As later shown in the results section it was found that the depletion of phenylalanine led to infeasible solutions. The bioreactor has a working volume of 6L. The stirring rates were manipulated within 200 rpm to 600 rpm by a PI controller to maintain a 35% dissolved oxygen (DO) target. The air flow was set constant at 6 slpm. The pH was controlled at 7.1 by phosphoric acid. Feeding of glutamate at 4.3 g/h was started following the depletion of glutamate that can be inferred from changes of DO. The fermentation temperature was set at 36°C. Biomass concentration was determined by optical density measurements at $600nm$ using a spectrophotometer. HPLC with a high efficiency Nova-Pak TM column was used to measure amino acids concentrations in the supernatant. AccQ Fluor (6-aminoquinolyl-N-hydroxysuccinimidyl carbamate) was used for precolumn derivatization and the separated derivatives were monitored by fluorescence detection resulting in a measurement error of less than 5%. Ammonia, lactate, glucose, and glutamine were measured in a Bio-profile Flex (Nova Biomedical). The experiments were repeated two times, referred to as experiments 1 and 2 in this paper.

Table 3.1: Parameters Used in the DFBA Model of *B. pertussis*

Parameters			Parameters		
$v_{max,ala}$	2.67×10^{-2}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,phe}$	8.25×10^{-4}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,arg}$	1.12×10^{-4}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,pro}$	9.43×10^{-2}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,asp}$	5.66×10^{-3}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,ser}$	6.39×10^{-2}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,gly}$	8.09×10^{-3}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,thr}$	2.54×10^{-3}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,his}$	2.19×10^{-3}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,tyr}$	2.63×10^{-4}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,ile}$	9.20×10^{-4}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,val}$	2.41×10^{-2}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,leu}$	1.90×10^{-2}	$(h \cdot mmol\ cell)^{-1}$	$v_{max,glu}$	8.80×10^{-5}	$(h \cdot mmol\ cell)^{-1}$
$v_{max,lys}$	1.24×10^{-2}	$(h \cdot mmol\ cell)^{-1}$	K_{glu}	1.341	$mmol/L$
$v_{max,met}$	1.20×10^{-2}	$(h \cdot mmol\ cell)^{-1}$			

In paper [13], only one kinetic expression Eq. (3.5) was used as a constraint according to Eq. (3.1b) corresponding to the conversion of glutamate to tyrosine which was found to be the limiting reaction. Kinetic parameters and other parameters of DFBA model used in the paper are listed in Table (3.1). These parameters $\boldsymbol{v_{max}}$ listed in Table (3.1) are used in constraints Eq. (3.1d). Additional details about the chosen metabolic network and model formulation can be found in [13].

$$\boldsymbol{S_{tyr}v_k} \leq \frac{v_{max,glu}\psi_{glu,k-1}}{K_{glu} + \psi_{glu,k-1}} \tag{3.5}$$

where $\boldsymbol{S_{tyr}}$ is the row vector in the stoichiometric matrix corresponding to tyrosine; $\psi_{glu,k-1}$ is glutamate concentration at the time instant $k-1$; v_{max} and K are the maximum flux and kinetic parameter in the rate expression respectively.

3.3.2 Weighted Primal-Dual Method

The most commonly used interior-point method is the naive primal-dual method (NPDM). The WPDM is a modified version of NPDM that was originally proposed to speed-up computation [30, 72]. In this research, the WPDM is applied for the first time to address the multiplicity problem [1, 20, 57]. Typically, in all interior-point methods, the standard form of the LP problem Eq. (3.3) is augmented by adding a logarithmic barrier function of constraints. The approximate LP formulation in the WPDM form is defined as per Eq.

(3.6),

$$\begin{aligned} \inf_{\boldsymbol{x},\boldsymbol{z},\mu\to 0} \quad & \boldsymbol{c}^T\boldsymbol{x} - \mu\sum_{j=1}^{m} w_j ln(z_j) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{b} \\ & \boldsymbol{z} > \boldsymbol{0} \quad (-\boldsymbol{I}\boldsymbol{z} < \boldsymbol{0}) \end{aligned} \tag{3.6}$$

where μ is a positive parameter controlling the pointwise approximation accuracy; $\boldsymbol{x}$ is assumed to be bounded. As μ tends to zero, the objective function approximates to a neighborhood of the optimum of the actual objective function Eq. (3.3). The bounded vector $\boldsymbol{w} = [w_1 \cdots w_n]^T > \boldsymbol{0}$ is a vector of internal point weights that serve as auxiliary parameters used for choosing a specific optimum among all possible optima. Each slack variable z_j corresponds to an interior-point weight w_j in Eq. (3.6). When all the weights are selected to be equal to one, the WPDM is reduced to the NPDM. The decision variables $\boldsymbol{z}$ cannot be zero due to the presence of the logarithmic function. However, $\boldsymbol{x}$ can be made infinitesimally small based on the choice of μ so that the accuracy can always satisfy users requirements. It should be noticed that the objective is defined by the infimum instead of the minimum since the infimum is achieved even though case $\boldsymbol{x} = \boldsymbol{0}$ is not reached. The case $\boldsymbol{x} = \boldsymbol{0}$ can also be addressed systematically by the idea of field extension [22].

The optimization problem defined in Eq. (3.6) is a strictly convex optimization problem so that the solution is unique if the feasible space is not empty. The standard KKT conditions of Eq. (3.3) is:

$$\boldsymbol{c} + \boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{\lambda} = \boldsymbol{0} \tag{3.7a}$$

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{b} \tag{3.7b}$$

$$-\boldsymbol{I}\boldsymbol{z} < \boldsymbol{0} \tag{3.7c}$$

$$\lambda_j \geq 0 \quad \forall j = 1, \cdots, n \tag{3.7d}$$

$$\lambda_j z_j = 0 \quad \forall j = 1, \cdots, n \tag{3.7e}$$

where λ_i are positive multipliers for inequality constraints $\boldsymbol{z} > \boldsymbol{0}$; Eq. (3.7e) is strong complementary slackness conditions. For WPDM, the standard KKT is adapted by choosing Lagrange multipliers $\lambda_j = \mu w_j / z_j \geq 0$ and replacing Eq. (3.7e) by weak complementary slackness conditions $\lambda_j z_j = \mu w_j, \mu \to 0$. Note that μ cannot be zero, otherwise multiplicity issue relapses. Eq. (3.7) can be reformulated as Eq. (3.8) for calculation used in later

sections.

$$\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{\lambda} = -\boldsymbol{c} \tag{3.8a}$$
$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{b} \tag{3.8b}$$
$$\boldsymbol{\Lambda}\boldsymbol{Z} = \mu\boldsymbol{w} \tag{3.8c}$$
$$-\boldsymbol{I}\boldsymbol{z} < \boldsymbol{0} \tag{3.8d}$$
$$\boldsymbol{\lambda} \geq \boldsymbol{0} \tag{3.8e}$$

where $\boldsymbol{Z}$ and $\boldsymbol{\Lambda}$ denote diagonal matrices, namely $\boldsymbol{Z} = diag(\boldsymbol{z})$, $\boldsymbol{\Lambda} = diag(\boldsymbol{\lambda})$; $\boldsymbol{e} = [1 \cdots 1]^T \in \boldsymbol{R}^n$ is a vector. Eqs. (3.8a)-(3.8c) are solved by the Newton method. A classic line search algorithm is used to control the step size in the Newton method as done in most interior-point methods such that Eqs. (3.8d)-(3.8e) are accounted [40].

The direct solution of Eq. (3.8) with a very small value of μ is often difficult to obtain when calculating the inverse of an ill-conditioned Jacobian matrix of a nonlinear set of equations. Instead, the WPDM algorithm adopts a path-following method that gradually decreases the parameter μ [29]. The steps of the WPDM algorithm are shown in Algorithm 1 where σ is a scaling parameter between 0 and 1 that is controlling the decreasing rate of μ.

The method used to solve Eq.(3.8) is a path-following method for linear programming as described in [29]. In each iteration of the path following algorithm, Newton's method is used to search for the decreasing direction $\boldsymbol{\Delta\lambda}$, $\boldsymbol{\Delta x}$, and $\boldsymbol{\Delta z}$. The line search algorithm is used to calculate the largest allowable step size α_λ and α_z so that $\boldsymbol{\lambda}$ and $\boldsymbol{z}$ are always positive at each iteration. Then, the point $(\boldsymbol{\lambda}^i, \boldsymbol{x}^i, \boldsymbol{z}^i)^T$ is the unique optimum for the current μ. In the next iteration, the new optimum is used as an initial guess for the next iteration. μ is progressively reduced from one iteration to the next to ultimately approximate the true optimum of the original problem. Thus, the optima obtained at each iteration follow a central path that finally approximates to the optimal solution. Interior-point weights $\boldsymbol{w}$ can be used as auxiliary parameters to change the curvature of the strictly convex surface of the objective function so that any optimal solution from multiple optima can be placed at the minimum position through tuning of $\boldsymbol{w}$. Algorithm 2 presents the application of WPDM to solve the DFBA model.

3.3.3 Hierarchical Optimization

Hierarchical optimization (HO) has been used for tackling primal multiplicity [35, 42, 46]. Instead of solving a single LP, HO involves the solution of a series of LP problems with

Algorithm 1 WPDM (Path-following Method)

Given $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{w}$, σ, μ, η_0, tolerance and feasible initial interior point $(\boldsymbol{\lambda}^0, \boldsymbol{x}^0, \boldsymbol{z}^0)$
$i = 0$
repeat
    Solve the following set of equations by the Newton method for $\boldsymbol{\Delta\lambda^i}$, $\boldsymbol{\Delta x^i}$, and $\boldsymbol{\Delta z^i}$:

$$\begin{bmatrix} \boldsymbol{A^T} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} & \boldsymbol{I} \\ \boldsymbol{Z^i} & \boldsymbol{0} & \boldsymbol{\Lambda^i} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta\lambda^i} \\ \boldsymbol{\Delta x^i} \\ \boldsymbol{\Delta z^i} \end{bmatrix} = \begin{bmatrix} \boldsymbol{-c - A^T\lambda^i} \\ \boldsymbol{b - z^i - Ax^i} \\ \boldsymbol{\mu w - \Lambda^i z^i} \end{bmatrix} \tag{3.9}$$

    Use line search algorithm to find the step size α_λ and α_z that satisfy :

$$\boldsymbol{\lambda}^i + \alpha_\lambda \boldsymbol{\Delta\lambda}^i \geq \boldsymbol{0} \tag{3.10}$$

$$\boldsymbol{z}^i + \alpha_z \boldsymbol{\Delta z}^i \geq \boldsymbol{0} \tag{3.11}$$

    Calculate $\eta = max\{\eta_0, 1-\mu\}$, $\alpha_\lambda^i = min\{1, \eta\alpha_\lambda\}$ and $\alpha_z^i = min\{1, \eta\alpha_z\}$
    Update $\boldsymbol{\lambda}^{i+1} = \boldsymbol{\lambda}^i + \alpha_\lambda^i \boldsymbol{\Delta\lambda}^i$, $\boldsymbol{x}^{i+1} = \boldsymbol{x}^i + \alpha_z^i \boldsymbol{\Delta x}^i$, and $\boldsymbol{z}^{i+1} = \boldsymbol{z}^i + \alpha_z^i \boldsymbol{\Delta z}^i$
    Update $\mu = \mu\sigma$, $i = i + 1$
until $\mu \leq tolerance$
Output $\boldsymbol{\lambda}^i$, $\boldsymbol{x}^i$, $\boldsymbol{z}^i$

Algorithm 2 Apply WPDM to solve DFBA:

Given $\boldsymbol{w}$, $\boldsymbol{d}$, h_0, $\boldsymbol{\psi_0}$, $\boldsymbol{S}$, Δt, n_t, $\boldsymbol{v_{max}}$, function $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$.
$k = 0$
repeat
    Calculate function $\boldsymbol{f}(\boldsymbol{\psi_k})$ and $\boldsymbol{g}(\boldsymbol{\psi_k})$
    Convert all inequality constraints Eqs. (3.1b), (3.1c), and (3.1d) into equality constraints $\boldsymbol{Ax} + \boldsymbol{z} = \boldsymbol{b}$ by adding slack variables
    Apply WPDM to solve the standard LP problem and get $\boldsymbol{v_k}$ from $\boldsymbol{x}$
    Calculate Eqs. (3.2a) and (3.2b) to update $\boldsymbol{\psi_k}$ and $\boldsymbol{h_k}$
    $k = k + 1$
until $k \geq n_t$
output $\{h^k\}$, $\{\boldsymbol{\psi^k}\}$

different auxiliary objectives to reduce the optimal space and to find a unique solution. Each of these LP problems have a specific biological objective and they are ranked according to a user predefined priority order. HO is defined as per Eq. (3.12) in standard form.

$$\begin{cases} p_0 = \min\limits_{\boldsymbol{x}} \quad \boldsymbol{c_0}^T \boldsymbol{x} & \text{(3.12a)} \\ \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{b} & \text{(3.12b)} \\ \quad \begin{cases} p_1 = \min\limits_{\boldsymbol{x}} \quad \boldsymbol{c_1}^T \boldsymbol{x} \\ \quad\quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{b} \\ \quad\quad \boldsymbol{c_0^T} \boldsymbol{x} = p_0 \\ \quad\quad \begin{cases} \vdots \end{cases} \end{cases} & \text{(3.12c)} \end{cases}$$

where p_0 and $\boldsymbol{c_0}$ are the minimum value and the vector of coefficients that define the objective of the original DFBA model; Eq. (3.12b) are the constraints of DFBA model in their standard form. Similarly, p_i and $\boldsymbol{c_i}$ $\forall i \in \mathbb{N}^+$ are the minimum values and vectors of coefficients defining the different objectives with respect to fluxes of auxiliary LP problems in a prescribed order. All auxiliary objectives and their ordering are defined by the user according to the expected importance. It should be noticed that the auxiliary LP problems at the lower priority levels do not only need to satisfy constraints as Eq. (3.12b), but they also must satisfy all constraints in the higher priority levels. For instance, the second layer LP does not only satisfies constraints as in the first layer LP, but also it must satisfy an extra equality constraint $\boldsymbol{c_0^T} \boldsymbol{x} = p_0$ to ensure the optimality of the first layer LP. HO cannot always ensure the uniqueness of solution of DFBA even though all auxiliary objectives have been used. The following priority order of objectives reported have been used in the case study: 1-maximum biomass yield, 2-maximum ATP yield, 3-minimum of the total flux, 4-maximum carbon dioxide yield, 5-maximum acetate yield, 6-minimum fluxes 1 to n_r sequentially [93]. The dual-simplex method of CPLEX is used to solve this problem.

3.3.4 Minimization of Enzyme Cost

The idea of minimization of enzyme cost [83] is also close to the idea of minimal flux adjustment. The method is based on the assumption that evolution has ruled out inefficient pathways so that cells need to biosynthesize less total enzymes while distributing fluxes in an efficient way. Minimization of enzyme cost or minimal flux adjustment are good auxiliary objectives to maximize the efficiency of the fluxes. However, kinetic parameters of reactions must be known to calculate exact enzyme cost. If these parameters are not

available, other auxiliary objectives have been used instead such as minimal total (absolute) flux, minimal number of active reactions, minimal norm of flux. To compare this method with WPDM in the case study, methods based on minimization of the total flux and minimization of the number of active reactions were used. Methods based on minimization of the total flux and minimization of the number of active reactions had been also referred as to the principle of flux minimization [47, 78]. Both these methods can be regarded as hierarchical optimizations with only two layers each. For the method of minimization of the total flux, the second layer objectives can be expressed as $\sum_{i=1}^{n_r} v_{k,i}$ in canonical form, where $v_{k,i}$ is the flux of i-th reaction at the time interval k.

The method of minimization of the number of active reactions is outlined in Eq. (3.13) using the canonical formulation.

$$
\begin{cases}
p_0 = \max\limits_{\boldsymbol{v_k}} \quad \boldsymbol{d}^T\boldsymbol{v_k} + \mathbf{0}_{\mathbf{1}\times\boldsymbol{n_r}}\boldsymbol{\kappa_k} & \text{(3.13a)}\\
\quad \boldsymbol{f}(\boldsymbol{\psi_{k-1}}) \le \boldsymbol{S}\boldsymbol{v_k} \le \boldsymbol{g}(\boldsymbol{\psi_{k-1}}) & \text{(3.13b)}\\
\quad \boldsymbol{\psi_{k-1}} + h_{k-1}\Delta t\boldsymbol{S}\boldsymbol{v_k} \ge 0 & \text{(3.13c)}\\
\quad \mathbf{0} \le \boldsymbol{v_k} \le \boldsymbol{v_{max}} & \text{(3.13d)}\\
\quad \boldsymbol{\kappa_k} \text{ are binary variables} & \text{(3.13e)}\\
\quad \boldsymbol{\psi_k} = \boldsymbol{\psi_{k-1}} + h_{k-1}\Delta t\boldsymbol{S}\boldsymbol{v_k} & \text{(3.13f)}\\
\quad h_k = h_{k-1} + h_{k-1}\Delta t\boldsymbol{d^T}\boldsymbol{v_k} & \text{(3.13g)}\\
\quad \begin{cases}
p_1 = \min\limits_{\boldsymbol{\kappa_k}} \quad \mathbf{0}_{\mathbf{1}\times\boldsymbol{n_r}}\boldsymbol{v_k} + \mathbf{1}_{\mathbf{1}\times\boldsymbol{n_r}}\boldsymbol{\kappa_k}\\
\quad -\boldsymbol{I}\boldsymbol{v_k} + \epsilon\boldsymbol{I}\boldsymbol{\kappa_k} \le \mathbf{0}\\
\quad \boldsymbol{I}\boldsymbol{v_k} - M\boldsymbol{I}\boldsymbol{\kappa_k} \le \mathbf{0}\\
\quad \boldsymbol{d}^T\boldsymbol{v_k} + \mathbf{0}_{\mathbf{1}\times\boldsymbol{n_r}}\boldsymbol{\kappa_k} = p_0\\
\quad \boldsymbol{f}(\boldsymbol{\psi_{k-1}}) \le \boldsymbol{S}\boldsymbol{v_k} \le \boldsymbol{g}(\boldsymbol{\psi_{k-1}})\\
\quad \boldsymbol{\psi_{k-1}} + h_{k-1}\Delta t\boldsymbol{S}\boldsymbol{v_k} \ge 0\\
\quad \mathbf{0} \le \boldsymbol{v_k} \le \boldsymbol{v_{max}}\\
\quad \boldsymbol{\kappa_k} \text{ are binary variables}\\
\quad \boldsymbol{\psi_k} = \boldsymbol{\psi_{k-1}} + h_{k-1}\Delta t\boldsymbol{S}\boldsymbol{v_k}\\
\quad h_k = h_{k-1} + h_{k-1}\Delta t\boldsymbol{d^T}\boldsymbol{v_k}
\end{cases} & \text{(3.13h)}
\end{cases}
$$

where $\boldsymbol{\kappa_k}$ is a vector of binary variables at time interval k, defining on-off switches for each reaction; ϵ is a tolerance to determine whether a flux can be treated as zero or not; $\mathbf{0}$ and $\mathbf{1}$ are matrices whose elements are 0 and 1 respectively; M is a very large value, e.g. 100000

is used in the case study. When $\kappa_{k,i}$ is zero, the i-th reaction is turned off while if $\kappa_{k,i}$ is one, the i-th reaction is turned on. In this research, the second layer problem is solved by a mixed integer linear programming solver of CPLEX.

3.3.5 Model Calibration

To compare among solutions obtained with different solvers and different methods to address multiplicity, two groups of replicate experimental data of fed-batch fermentation of *B. pertussis* reported in the paper of [13] were used to calibrate the model. These two groups of replicate experimental data are referred to experiment 1 and 2 in the following sections. Additional details on the experimental data can be found in [13]. Although all the parameters of the original DFBA model and auxiliary parameters $\boldsymbol{w}$ of WPDM could be adjusted to fit the experimental data, this is very impractical because only a small subset of the parameters has a significant impact on the solution [44, 69]. Thus, a sensitivity analysis is conducted to determine the top 5 model parameters and the top 5 auxiliary parameters that have the largest effect on the solution. The sensitivity S^{θ_i} is the sensitivity of parameter θ_i defined as in Eq. (3.14).

$$S^{\theta_i} = \sum_{m=1}^{n_m} \sum_{k=1}^{n_t} \left| S_{\psi_m}^{\theta_i}(t_k) \right| + \sum_{k=1}^{n_t} \left| S_h^{\theta_i}(t_k) \right| \tag{3.14}$$

where $S_{\psi_m}^{\theta_i}(t_k)$ is the sensitivity of concentrations ψ_m of metabolite m to the i-th parameter θ_i at sampling time interval k. Similarly, $S_h^{\theta_i}(t_k)$ is the sensitivity of biomass to i-th parameter at sampling time interval k; and the calculations of sensitivities for biomass and metabolites follow Eqs. (3.15) and (3.16) respectively.

$$S_{\psi_m}^{\theta_i}(t_k) = \frac{\partial \psi_m(t_k)}{\partial \theta_i} \frac{\theta_i}{\overline{\psi}_m} \tag{3.15}$$

where $\overline{\psi}_m$ is the average metabolite concentrations over all sampling time intervals according to Eq. (3.16).

$$\overline{\psi}_m = \frac{1}{n_t} \sum_{k=1}^{n_t} \psi_m(t_k) \tag{3.16}$$

The sum of squared errors (SSE) of key metabolites and biomass as defined in Eq. (3.17) is used for model calibration and for comparing the fitting accuracy of different LP solvers and different methods for addressing multiplicity, where $\psi_{m,exp}(t_k)$ and $h_{exp}(t_k)$ are

two sets of the experimental data of metabolites m and biomass at time t_k respectively.

$$SSE = \sum_{m=1}^{n_m} \sum_{k=1}^{n_t} (\psi_m(t_k) - \psi_{m,exp}(t_k))^2 + \sum_{k=1}^{n_t} (h(t_k) - h_{exp}(t_k))^2 \tag{3.17}$$

The selected top 5 sensitive parameters are then adjusted to fit the experimental data with different solvers, including the interior-point method (IPM) of CPLEX, the interior-point method of MATLAB R2018a, and the NPDM; the aforementioned methods for addressing multiplicity, including minimization of the total flux, minimization of the number of active reactions, and hierarchical optimization. For the WPDM, both the top 5 parameters of the DFBA model and the top 5 auxiliary parameters interior-point weights that were obtained from the sensitivity analysis are tuned to fit the data. In principle, all weights (over 200) can be tuned . But for simplicity, as comparison, only top 5 weights . It should be emphasized that while the interior-point weights can be adjusted to obtain different optima, the weights to be chosen for a particular DFBA problem are the ones that result in a better fit of the experimental data. Thus, these auxiliary parameters do not change the definition of the original DFBA problem, but only provide a means to choose one particular flux distribution among all possible distributions. The selected top 5 sensitive parameters are then adjusted to fit the experimental data with different solvers, including the interior-point method (IPM) of CPLEX, the interior-point method of MATLAB R2018a, and the NPDM; the aforementioned methods for addressing multiplicity, including minimization of the total flux, minimization of the number of active reactions, and hierarchical optimization. For the WPDM, both the top 5 parameters of the DFBA model and the top 5 auxiliary parameters interior-point weights that were obtained from the sensitivity analysis are tuned to fit the data. In principle, all weights (over 200) can be tuned . But for simplicity, as comparison, only top 5 weights . It should be emphasized that while the interior-point weights can be adjusted to obtain different optima, the weights to be chosen for a particular DFBA problem are the ones that result in a better fit of the experimental data. Thus, these auxiliary parameters do not change the definition of the original DFBA problem, but only provide a means to choose one particular flux distribution among all possible distributions. The selected top 5 sensitive parameters are then adjusted to fit the experimental data with different solvers, including the interior-point method (IPM) of CPLEX, the interior-point method of MATLAB R2018a, and the NPDM; the aforementioned methods for addressing multiplicity, including minimization of the total flux, minimization of the number of active reactions, and hierarchical optimization. For the WPDM, both the top 5 parameters of the DFBA model and the top 5 auxiliary parameters interior-point weights that were obtained from the sensitivity analysis are tuned to fit the

data. In principle, all weights (over 200) can be tuned . But for simplicity, as comparison, only top 5 weights . It should be emphasized that while the interior-point weights can be adjusted to obtain different optima, the weights to be chosen for a particular DFBA problem are the ones that result in a better fit of the experimental data. Thus, these auxiliary parameters do not change the definition of the original DFBA problem, but only provide a means to choose one particular flux distribution among all possible distributions. The selected top 5 sensitive parameters are then adjusted to fit the experimental data with different solvers, including the interior-point method (IPM) of CPLEX, the interior-point method of MATLAB R2018a, and the NPDM; the aforementioned methods for addressing multiplicity, including minimization of the total flux, minimization of the number of active reactions, and hierarchical optimization. For the WPDM, both the top 5 parameters of the DFBA model and the top 5 auxiliary parameters interior-point weights that were obtained from the sensitivity analysis are tuned to fit the data. In principle, all weights (over 220 weights in this example) can be tuned. But for simplicity, only top 5 weights were selected as a comparison. It should be emphasized that while the interior-point weights can be adjusted to obtain different optima, the weights to be chosen for a particular DFBA problem are the ones that result in a better fit of the experimental data. Thus, these auxiliary parameters do not change the definition of the original DFBA problem, but only provide a means to choose one particular flux distribution among all possible distributions.

3.4 Theoretical Properties of WPDM

As mentioned in the literature review, some desirable properties for a method that addresses the primal multiplicity issue in DFBA are: uniqueness and continuity of the optimal solution, availability of tuning parameters to select a specific optimum among the multiple optima so as to permit consistency with experimental data and computational efficiency. In this section, WPDM is shown to have properties of uniqueness, continuity and tunability to apprimate to a specific optimum. One drawback of WPDM as compared to other methods is the computational expense required due to the use of a Newton's method to solve a set of nonlinear equations. However, it should be noticed that the main computational effort is required for off-line calibration of the weights of WPDM with experimental data which, as shown later in the case study, it is not a factor for online applications where the weights values are kept constant with time.

Lemma 1 For a convex optimization problem, if the objective is strictly convex, then the optimal set contains at most one point [12].

Theorem 1 Uniqueness For given positive $\boldsymbol{w}$ and μ, the optimization problem in the weighted logarithmic form defined in Eq. (3.6) has at most one optimal solution.

Proof The logarithmic function is a strictly concave function. Because $\mu > 0$ and $\boldsymbol{w} > \mathbf{0}$, $\mu \sum_{j=1}^{n} w_j ln(z_j)$ is strictly concave function. The negative sign of the logarithmic term in the objective function Eq. (3.6) makes the function strictly convex. $\boldsymbol{c}^T\boldsymbol{x}$ and $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{z}$ are affine functions, which are convex but not strictly convex. The objective function is still strictly convex because it is the summation of a strictly convex function and a convex function.

Because the constraints are convex and the objective is strictly convex, the weighted logarithmic form defined in Eq. (3.6) is a strictly convex optimization problem. According to Lemma 1, this type of optimization problem with the weighted logarithmic formulation defined in Eq. (3.6) has at most one optimal solution.

Remark 1 Theorem 1 proves that for a given μ, every $\boldsymbol{w}$ corresponds to at most one optimal solution. Uniqueness is a key property of WPDM.

Theorem 2 Convergence to an approximate solution Assume the feasible space of problems Eq. (3.3) and Eq. (3.6) is not empty, $\boldsymbol{w}$, $\boldsymbol{z}$, and $\boldsymbol{x}$ are bounded. $\exists \bar{\mu} > 0$, so that solution of Eq. (3.6) is also an approximate solution of Eq. (3.3) if $\mu \in [0, \bar{\mu}]$.

Proof Because $\sum_{j=1}^{n} w_j ln(z_j)$ in the objective of Eq. (3.6) is a strictly convex function as proven in Theorem 1, Theorem 2 can be regarded as a special case of the proof of Theorem 1 in [67].

Remark 2 Theorem 2 establishes the approximation property of WPDM. When multiple optima exist, $\boldsymbol{w}$ can be tuned to obtain different bounded optima $\boldsymbol{x}$. A bounded $\boldsymbol{w}$ only influences which solution is selected among the multiple solutions but it cannot affect the optimality of $\boldsymbol{x}$. Thus, if the original LP problem has a unique solution, the solution of Eq. (3.6) is still the solution of Eq. (3.3) for any bounded $\boldsymbol{w}$. Following the approximation property proven in Theorem 2, once μ and $\boldsymbol{w}$ are specified, the resulting solution of LP problem by the WPDM is unique. Thus a specific time trajectory for each metabolite, among the many possible feasible trajectories, can be obtained by applying WPDM. Additional proofs of approximation and other properties of WPDM are in [1, 20, 57].

Ideally the optimal solution is obtained when μ is infinitesimal. In practice, a very small value ($\mu = 1e^{-8}$ in the case studies) is used. From Eq. (3.8c), the duality gap $\lambda^T z = \mu \sum w_j$ of WPDM is a measure of the objective function error [20]. Thus the approximation error is controllable by the value of μ selected by the user.

Theorem 3 Assume the solution set of Eq.(3.6) is not empty, the optimal solution is locally continuous with respect to $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{w}$. The proof of local continuity is given in Appendix A and it is based on [90].

Remark 3 The optimal solutions found by Simplex methods are not continuous when primal multiplicity occurs. If the DFBA model is to be used for control, small perturbations due to disturbances or noise can cause discontinuity in solutions and non-smooth trajectories. In the application of DFBA models in process control, it is necessary that the WPDM will be robust to these perturbations, i.e. the resulting optimal solution is continuous in the presence of perturbations. Theorem 3 proves the continuity at the optimal solution with respect to perturbations of $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{w}$.

3.5 Results and Discussion

3.5.1 Primal Multiplicity of the DFBA of *B. pertussis*

First the same DFBA model of *B. pertussis* [13] is solved by different solvers to show that different optima are arbitrarily chosen by these solvers. The tested LP solvers are the dual-simplex method, the interior-point method of MATLAB R2018a and the dual-simplex method and the interior-point method of CPLEX 12.8 (IBM). Commercial software such as CPLEX uses an additional post-processing algorithm referred to as a crossover algorithm that can find an optimum at a vertex starting from an optimum not located at a vertex [48] Corporation, 2016). In this paper, all computations are conducted without crossover.

The DFBA model of *B. pertussis* reported in [13] was originally fitted using the interior-point method of MATLAB version R2018a. To show the existence of multiple optima in the original DFBA model, different LP solvers were used to calculate the optimal flux distribution at the first time interval. Fig. 3.1 displays the radar chart of the calculation results. As shown in Fig. 3.1, even though the initial conditions and growth rates (objective) are the same, significantly different flux distributions can be obtained in the first time interval and thereafter. Moreover, the optimal flux distribution obtained is arbitrarily determined

by the particular algorithm while the user has limited ability to choose a different solution from the one provided by the solver.

The variability in flux distributions obtained for the first time interval with the different solvers is critical since it results in drastically different trajectories of metabolite concentrations over time in the fed-batch operation. For example, Fig. 3.2 and Fig. 3.3 show the evolution of metabolites and biomass concentrations obtained with the different LP solvers for the original DFBA model. For confidentiality reasons, all concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless. It can be seen from these figures that the trajectories obtained with the two interior-point methods are somewhat similar to each other and fit better to the experimental data as compared to the trajectories calculated with the two dual-simplex methods. Furthermore, the dual-simplex method of MATLAB R2018a was not able to find any feasible solution from 50 hours and on due to the early depletion of phenylalanine which cannot be biosythesized while it is essential for biomass synthesis. Hence, the choice of a commercial solver is critical not only in terms of the accuracy of the optimal solutions but also in terms of the feasibility of the optimization problem over time.

3.5.2 Application of WPDM

As established in the previous section, the WPDM can find a unique optimum once the interior-point weights $\boldsymbol{w}$ and μ are chosen. First, a toy example reported in the literature is used to illustrate the performance of the WPDM. Later, different solvers and different methods for primal multiplicity are used to calibrate a DFBA model of *B. pertussis*. In this paper, $\sigma = 0.1$, tolerance=$1e^{-8}$, initial $\mu = 10$, $\eta_0 = 0.995$ are used for WPDM.

Example of Simple LP Problem with Multiple Optima

A linear programming problem with multiple optima from Motamedian's paper is used as a preliminary case study (Motamedian and Naeimpoor, 2018). The LP problem is defined in Eq. (3.18). There are 4 vertex optima, $[4 \quad 0 \quad 2]^T$, $[0 \quad 4 \quad 4]^T$, $[0 \quad 4 \quad 0]^T$, $[4 \quad 0 \quad 0]^T$.

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & -2x_1 - 2x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 4 \\ & x_1 + 2x_3 \leq 8 \\ & \boldsymbol{x} \geq \mathbf{0} \end{aligned} \tag{3.18}$$

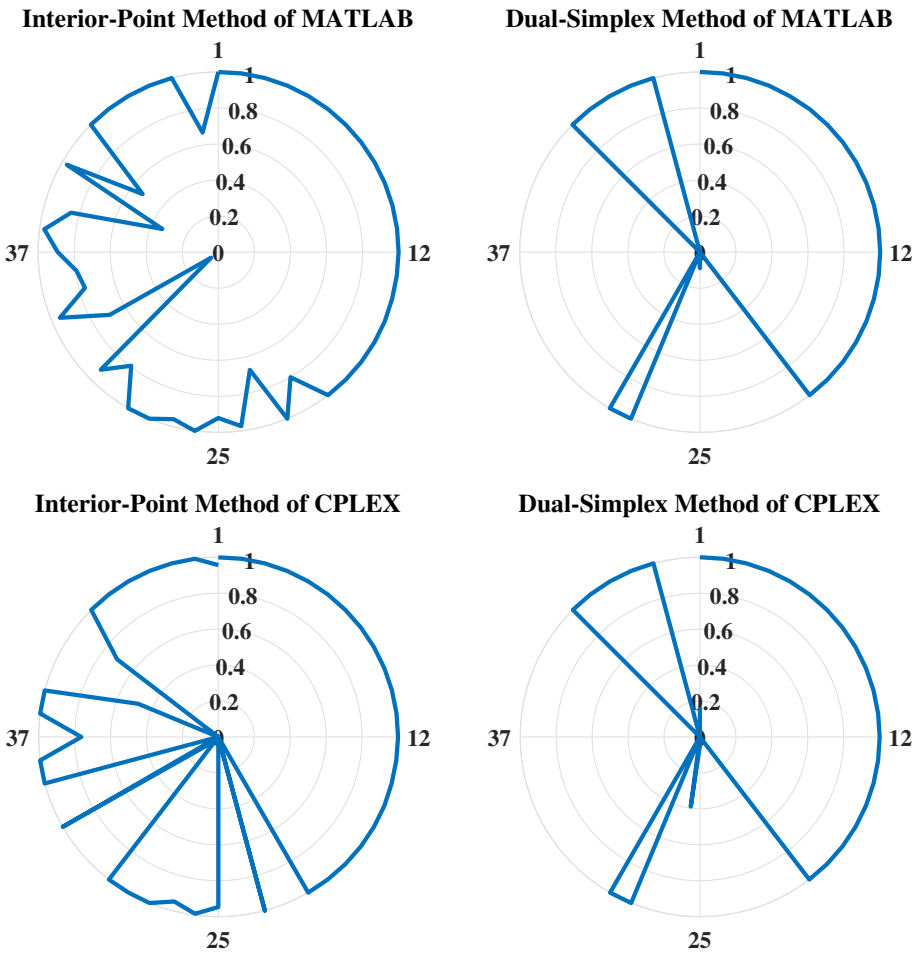


Figure 3.1: Radar charts of optimal flux distribution in the first time interval obtained by different solvers, including interior-point method and dual-simplex method of MATLAB and CPLEX. There are 49 reactions in the metabolic networks, but only reaction 1, 12, 25 and 37 are labeled in the radar plots. To facilitate the comparison, all fluxes were normalized within the interval $[0, 1]$ by dividing each flux by the maximum flux obtained for each reaction. Even though the initial conditions are the same, the optimal flux distributions arbitrarily selected by the solver are significantly different.

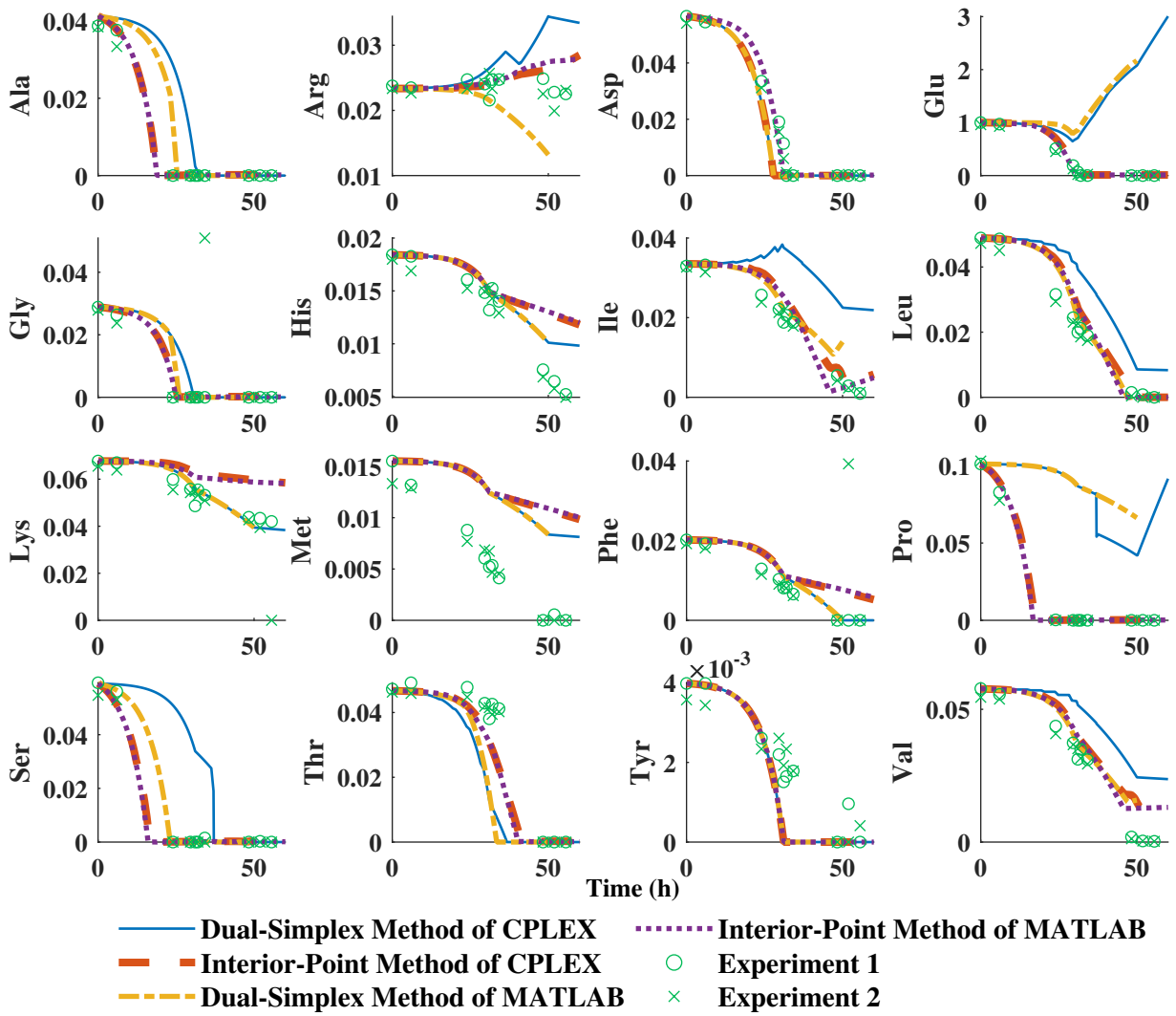


Figure 3.2: Evolution of key metabolites concentrations with time obtained by four different LP solvers. Different LP solvers were used to solve the original DFBA model for fed-batch fermentation of *B. pertussis*. Experiment 1 and 2 are two replicate experiments used in building the original DFBA model. The trajectories obtained by the dual-simple of MATLAB are discontinuous due to infeasibility around 50h. All concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless.

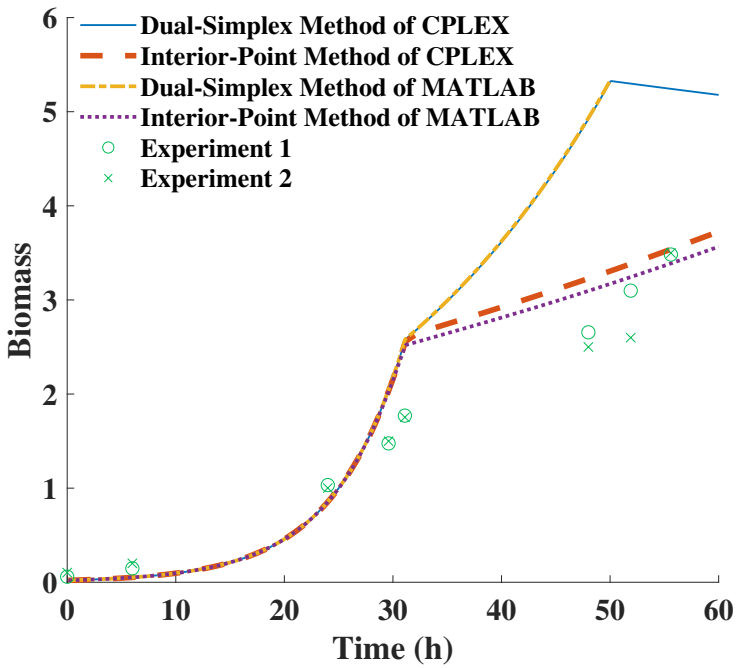


Figure 3.3: Evolution of biomass concentration with time obtained by four different LP solvers. Different LP solvers were used to solve the original DFBA model for fed-batch fermentation of *B. pertussis*. Experiment 1 and 2 are two replicate experiments used in building the original DFBA model. The trajectories obtained by dual-simple of MATLAB are discontinuous due to infeasibility around 50h. All concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless.

First, the original problem is converted into the standard form according to Eq. (3.19), where $\boldsymbol{z}$ are slack variables. These five variables correspond to five interior-point weights $w_1, \cdots, w_5$.

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & -2x_1 - 2x_2 \\ \text{subject to} \quad & x_1 + x_2 + z_1 = 4 \\ & x_1 + 2x_3 + z_2 = 8 \\ & -x_1 + z_3 = 0 \\ & -x_2 + z_4 = 0 \\ & -x_3 + z_5 = 0 \\ & \boldsymbol{z} \geq \mathbf{0} \end{aligned} \tag{3.19}$$

In Fig. 3.4, the polyhedron shows the feasible space of this LP problem as per Eq. (3.18); the shaded hyperplane is the active optimal hyperplane defined by the constraint $x_1 + x_2 \leq 4$. The other boundaries of this polyhedron are defined by other inactive constraints. Four vertexes of this optimal hyperplane are basic optima while the remaining part of the optimal hyperplane corresponds to optima not located at vertexes. Six possible paths denoted by different symbols are shown in Fig. 3.4. Every point of the path corresponds to different optima obtained by using different interior-point weights. The intersection of the six paths shown in the figure corresponds to the analytic center of the optimal hyperplane when all the elements of w are set to 1. By increasing an interior-point weight from 1 to 1000 while keeping the other weights constant, the obtained optimum can be directed towards a specific optimum. From Fig. 3.4, it can be seen that increasing the values of the weights lead to an increase of the corresponding values of the slack variables. For example, as w_2 increases, the corresponding slack variable z_2 increases which means that the optimum point stays away from the boundary of $x_1 + 2x_3 \leq 8$. It is also found that the ratio between different interior-points can be adjusted to determine which optimum is obtained rather than adjusting the weights individually.

Model Calibration by Different Solvers

As shown in the previous example, different optima can be achieved by changing the interior-point weights of the WPDM when multiple optima exist. This property can be used to tackle the multiplicity of solutions occurring in the DFBA model of *B. pertussis* by selecting a set of weights that result in solutions that best fit the experimental data. In this case study, six solvers are compared in terms of their ability to fit the experimental data reported in [13]. It is found that the dual-simplex method of CPLEX and the dual-simplex

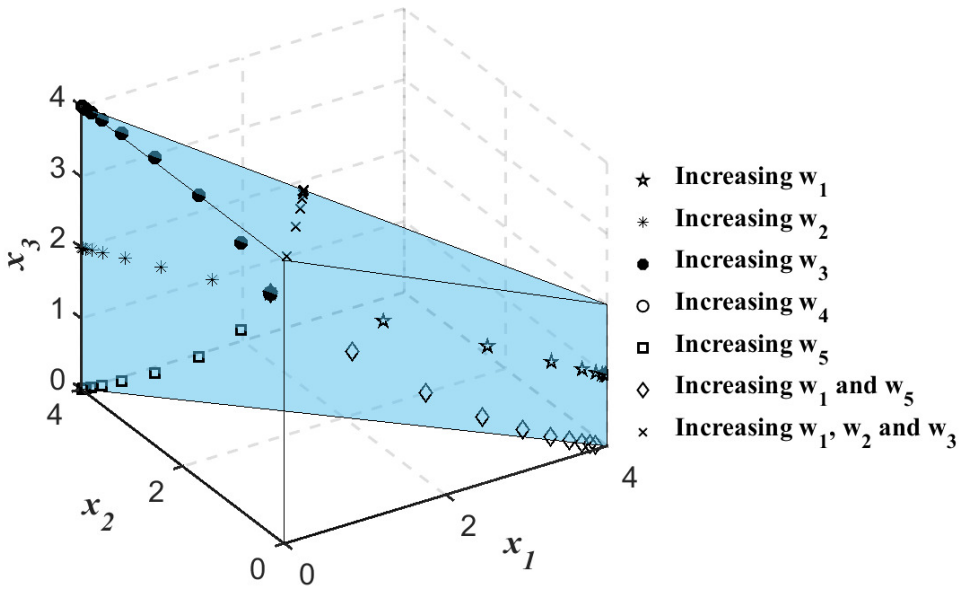


Figure 3.4: Control of interior-point weights $\boldsymbol{w}$ to approximate to different optima. The polyhedron is a feasible space formed by three decision variables x_1, x_2 and x_3. Increasing a particular interior-point weight leads to an increase in the corresponding slack variables $\boldsymbol{z}$ at the optimal solution so that any optimum can be obtained by tuning $\boldsymbol{w}$.

method of MATLAB cannot fit the experimental data and deviate significantly from the experiments. Thus, for brevity, only the remaining four solvers have been presented here: the interior-point method of CPLEX, the interior-point method (IPM) of MATLAB, the NPDM, and the WPDM.

Table 3.2 lists the SSE values for all metabolites and biomass as calculated for the different LP solvers. As shown in Table 3.2, the SSE obtained with WPDM is the smallest among the four solvers. In particular, the prediction of biomass is significantly more accurate by NPDM and WPDM than the values predicted with IPM of MATLAB and CPLEX. In general, for most metabolites, NPDM and WPDM resulted in significantly better fit. Although as shown in Fig. 3.5 and Fig. 3.6, there are no discernible differences for biomass predictions by NPDM and WPDM, the SSE of some metabolites by NPDM are much higher than for WPDM. For example, the SSE of isoleucine, leucine, threonine, and valine by NPDM are much higher than WPDM, as much as 224% larger as compared to WPDM. The fit is illustrated in Fig. 3.5 showing the profiles of these amino acids as compared to the experimental data. This corroborates that WPDM can find a unique set of trajectories among all possible optimal trajectories that best approximates the data. The tuning of the interior-point weights can be viewed as a way to compensate for insufficient information about the assumed DFBA model structure.

3.5.3 Model Calibration with alternative methods used to address Primal Multiplicity

A comparison is conducted between models calibrated by different methods for tackling primal multiplicity. These methods include WPDM, minimization of the total flux (MTF), minimization of the number of active reactions (MNAR) as defined in Eq. (3.13), and hierarchical optimization (HO) as defined in Eq. (3.12). For all methods the top 5 sensitive model parameters for each model were adjusted to fit the data.

As shown in Tab. (3.3) and Figs. (3.7) and (3.8), models calibrated by MTF and MNAR cannot fit well the metabolites and biomass concentrations. For example, for glutamate, the main limiting substrate, the deviation from experimental data is significant. It can be seen from Fig. (3.8), that the logarithmic phase of biomass occurs earlier than the actual growth and the growth remains too fast even after 20h. From Figs. (3.7) and (3.8), the model calibrated by HO can better describe the dynamics of metabolites and biomass

[1]IPM, interior-point method
[2]WPDM, weighted primal-dual method
[3]NPDM, naive primal-dual method

Table 3.2: Summation of Squared Errors (SSE) of Fitting by Different LP Solvers

	SSE of IPM[2], CPLEX	**Compared with WPDM[3]**	**SSE of IPM, MATLAB**	**Compared with WPDM**	**SSE of NPDM[4]**	**Compared with WPDM**	**SSE of WPDM**
Biomass	7419	123%	15446	363%	3345	0%	3333
Ala	0.181	1%	0.174	-3%	0.180	0%	0.180
Arg	0.484	-36%	0.970	29%	0.756	0%	0.754
Asp	3.401	95%	1.778	2%	1.670	-4%	1.746
Glu	2482	509%	457.4	12%	401.6	-1%	407.6
Gly	15.13	6%	13.98	-2%	14.30	0%	14.30
His	1.869	9%	1.519	-11%	1.713	0%	1.712
Ile	2.483	303%	1.333	116%	1.819	195%	0.616
Leu	8.112	355%	2.844	60%	5.778	224%	1.782
Lys	34.34	10%	28.79	-8%	31.16	0%	31.14
Met	7.094	4%	6.275	-8%	6.832	0%	6.831
Phe	8.599	4%	7.596	-8%	8.286	0%	8.284
Pro	1.139	-15%	0.821	-39%	1.394	4%	1.344
Ser	31.74	16245%	0.310	60%	0.193	0%	0.194
Thr	2.490	5%	6.979	193%	5.575	134%	2.381
Tyr	0.057	-24%	0.145	93%	0.075	0%	0.075
Val	13.17	137%	6.954	25%	9.776	76%	5.550
Total	10031	-	15983	-	3837	-	3818

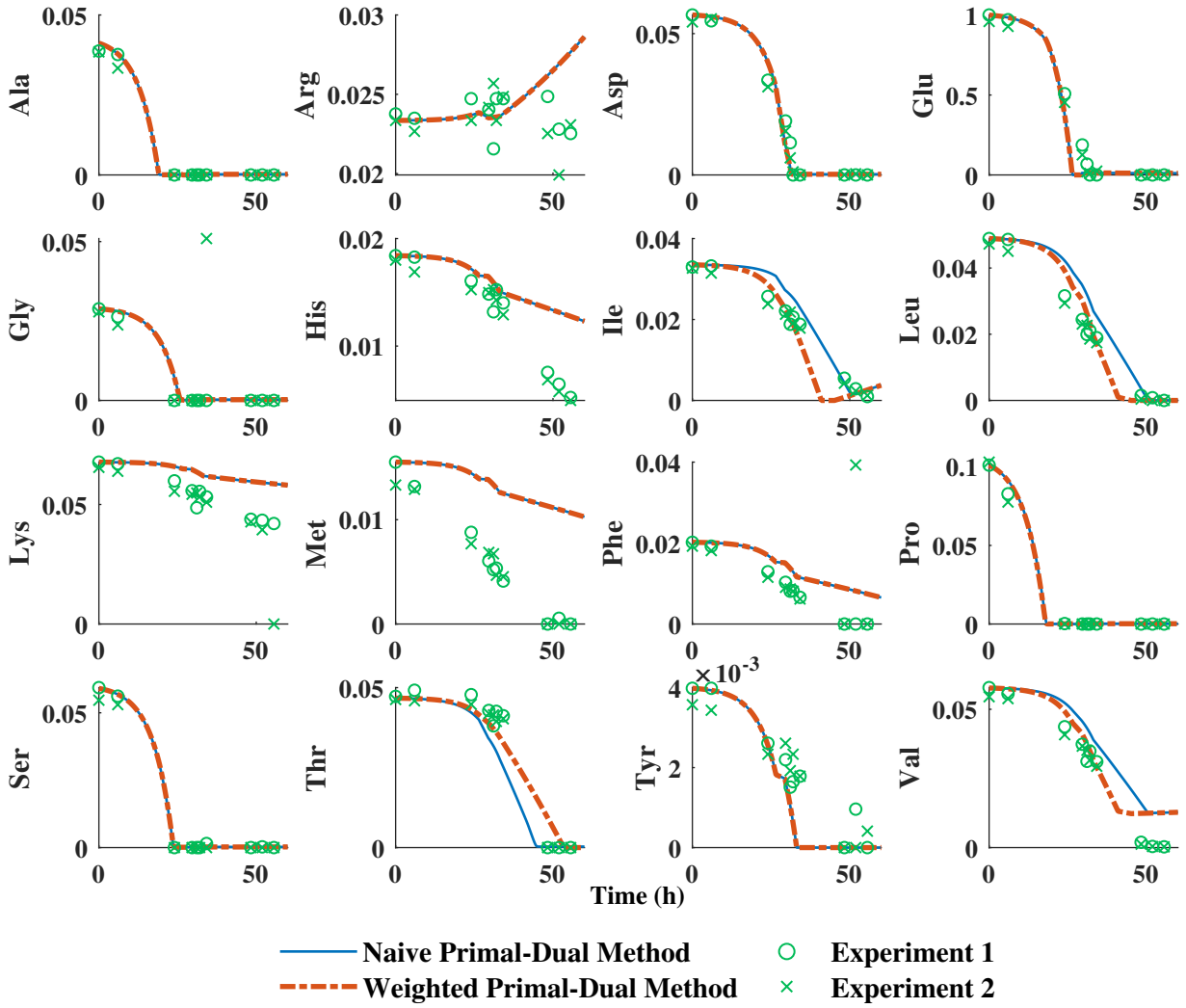


Figure 3.5: Evolution of metabolite concentrations with time as obtained by NPDM and WPDM. The DFBA models are calibrated based on the tuning of the top 5 most sensitive parameters by NPDM and WPDM respectively. For confidentiality, all concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless.

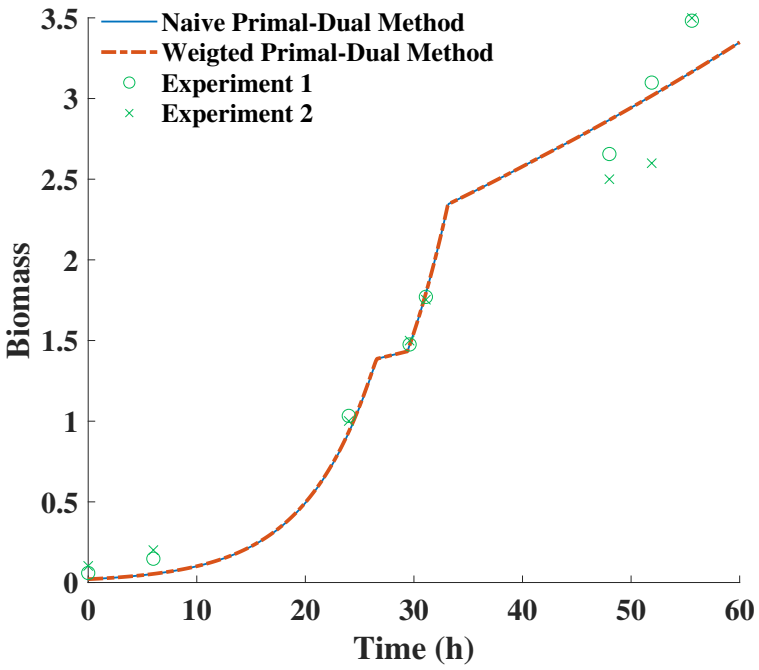


Figure 3.6: Evolution of biomass concentration with time as obtained by NPDM and WPDM based on tuning of the top 5 sensitive parameters. The biomass trajectory with time for NPDM and WPDM are overlapping. For confidentiality, all metabolites' concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless.

as compared to the MTF and MNAR methods. However, from Tab. (3.3), the fitting accuracy of HO is worse than WPDM, especially with respect to biomass. The SSE of biomass predicted by the WPDM is 350% lower than the one obtained for HO. The model based on HO fit well some amino acids' concentrations with SSE values slightly lower as compared to WPDM. However, for some amino acids the SSE obtained with HO is much higher as compared to the corresponding SSE obtained with WPDM.

In general, it was found that the flux distributions obtained by these methods are significantly different from each other. From Fig. (3.9), yield of CO_2 and NH_3 from 30h and on are much higher when calculated with HO and WPDM as compared to the levels calculated with MTF and MNAR. It was also observed that after the transition from rapid growth to the stationary phase, the flux distributions selected by MTF and MNAR are still dominated by anabolic reactions whereas for HO and WPDM they are dominated by catabolic reactions. Because methods based on MTF and MNAR give preference to efficient pathways, slow consumption of glutamate results in relatively abundant biomass and accumulation of glutamate during feeding in fed-batch operation. Methods based on HO and WPDM seem to better capture the transition from anabolism to catabolism and glutamate is found to be effectively depleted as an energy source after 30h. Transcript abundance regulation found between growth phase and nutrient-limited phase in experiments of *B. pertussis* also supports the observed downregulated anabolism [79]. For the HO method, the catabolism-dominated flux distribution is selected because of the objective considered in the fourth layer involving maximization of carbon dioxide yield. For WPDM, the catabolism-dominated flux distribution is enforced by tuning of the interior-point weights $\boldsymbol{w}$. It should also be noticed that the fitting of the HO method could be potentially improved by changing the auxiliary objectives and their ordering by either trial-and-error or based on prior knowledge. However, the selection of auxiliary objectives and their ordering for fitting is a relatively difficult combinatorial problem. In contrast, the optimal interior-point weights $\boldsymbol{w}$ are found by fitting experimental data without any prior-knowledge about the system. This data-driven feature makes WPDM attractive when prior biological knowledge is not readily available.

On the other hand, WPDM is more computationally expensive as compared to the other methods since it requires the solution of a set of nonlinear equations by a Newton's method. For example, 18.4h CPU time is needed for calibrating the weights by MATLAB as compared to 12.3min for MTF by CPLEX, 14.4min for MNAR by CPLEX, and 10.9min for HO by CPLEX. However, it should be remembered that the higher computational effort is only a factor while calibrating the weights which may not be critical for not too large networks since this step is performed off-line. After the weights are obtained, the execution of the algorithm is relatively fast. For example, for a simulation of 60h fed-batch

fermentation, 45.1s CPU time is needed for execution of WPDM with Matlab, 2.3s CPU time for MTF by CPLEX, 9.8s CPU time for MNAR by CPLEX and 8.4s CPU time for HO by CPLEX. Thus the computational expense is not a major limitation for online implementation of the WPDM algorithm, e.g. for online estimation or control. Also, it should be remembered that WPDM was coded in MATLAB which may result in generally slower code as compared to the other algorithms that were implemented with CPLEX.

Table 3.3: Summation of Squared Errors (SSE) of Fitting by Different Methods for Primal Multiplicity

	SSE of MTF[4]	**Compared with WPDM**[5]	**SSE of MNAR**	**Compared with WPDM**	**SSE of HO**[6]	**Compared with WPDM**	**SSE of WPDM**
Biomass	34287	929%	27683	730%	15010	350%	3333
Ala	0.376	109%	0.376	109%	0.123	-32%	0.180
Arg	3.731	395%	1.778	136%	2.043	171%	0.754
Asp	14.43	726%	13.72	686%	4.711	170%	1.746
Glu	277621	68013%	287331	70395%	4174	924%	407.6
Gly	13.89	-3%	13.90	-3%	13.84	-3%	14.30
His	1.275	-26%	1.362	-20%	1.765	3%	1.712
Ile	6.268	918%	6.907	1022%	1.743	183%	0.616
Leu	5.696	220%	6.931	289%	1.106	-38%	1.782
Lys	12.07	-61%	12.69	-59%	15.97	-49%	31.14
Met	5.919	-13%	6.064	-11%	6.645	-3%	6.831
Phe	7.529	-9%	7.557	-9%	7.786	-6%	8.284
Pro	597.7	44372%	603.2	44784%	0.153	-89%	1.344
Ser	0.144	-26%	0.144	-26%	0.232	20%	0.194
Thr	79.78	3251%	34.38	1344%	63.65	2573%	2.381
Tyr	0.243	223%	0.243	223%	0.182	142%	0.075
Val	19.56	253%	21.66	290%	5.365	-3%	5.550
Total	312678	-	315745	-	19309	-	3818

[4]MTF, minimization of the total flux
[5]MNAR, minimization of the number of active reactions
[6]HO, hierarchical optimization

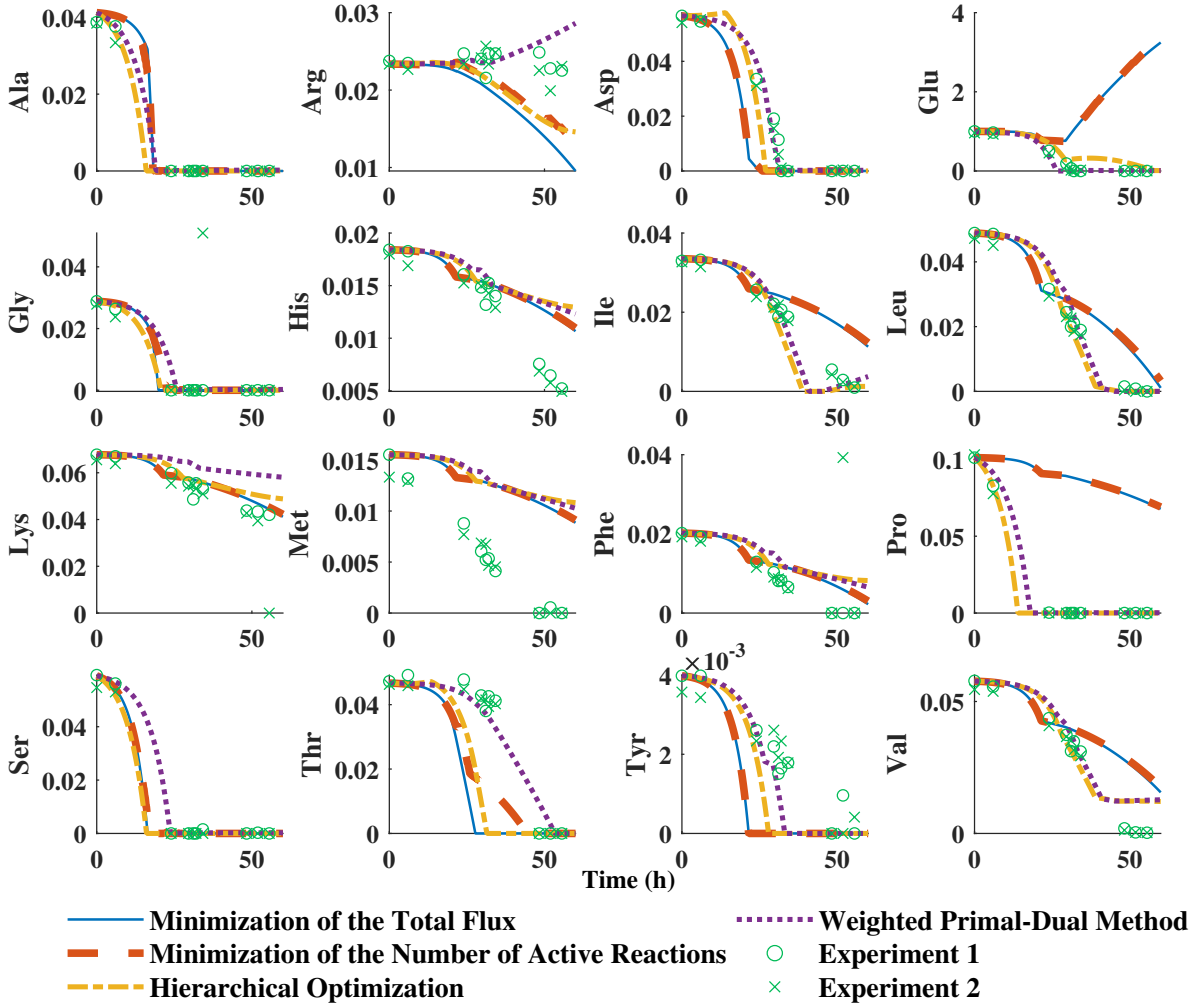


Figure 3.7: Comparison of the time evolution of metabolite concentrations with time obtained by different methods for primal multiplicity, including minimization of the total flux (MTF), minimization of the number of active reactions (MNAR), hierarchical optimization (HO) and WPDM. The top 5 most sensitive parameters are tuned for the calibration of the DFBA models used with these methods. The ordering of the objectives used for HO are: maximum biomass yield, maximum ATP yield, minimum of the total flux, maximum carbon dioxide yield, maximum acetate yield, minimum fluxes 1 to n_r sequentially to assure the unique solution. The first five objectives are reported as good fitting with experimental data in [93]. For confidentiality, all concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless.

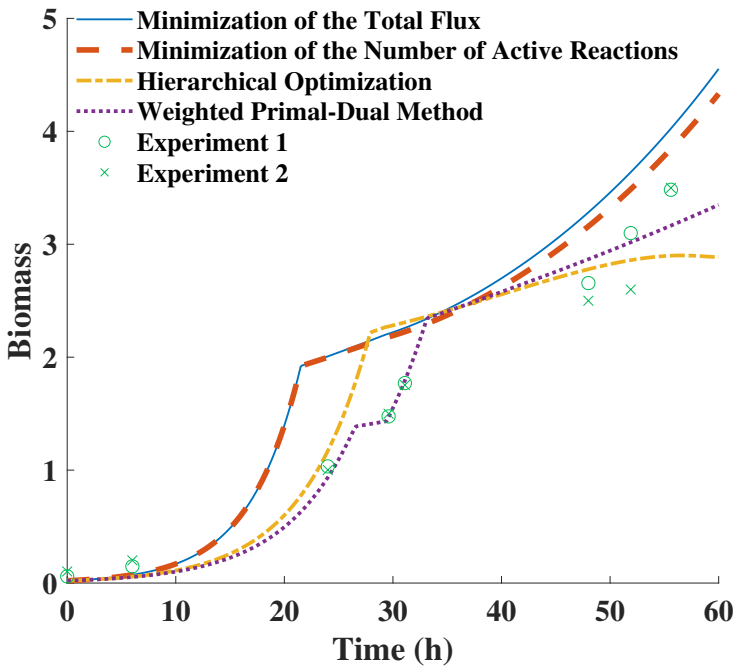


Figure 3.8: Comparison of the time evolution of biomass concentration by different methods for primal multiplicity : minimization of the total flux (MTF), minimization of the number of active reactions (MNAR), hierarchical optimization (HO) and WPDM.

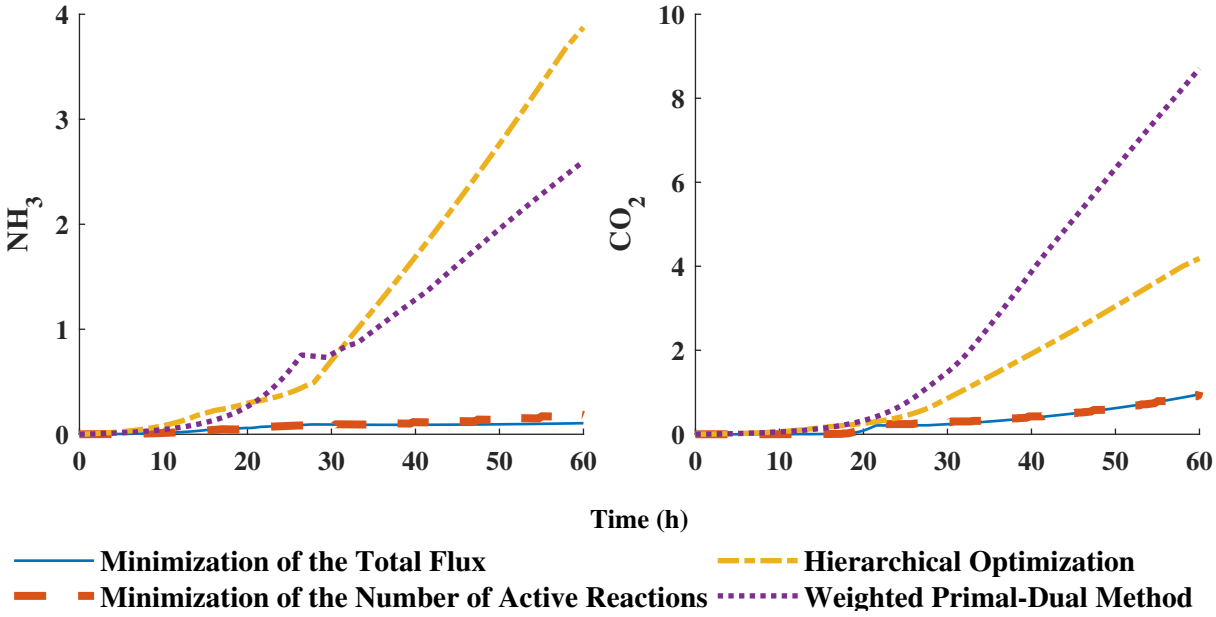


Figure 3.9: Comparison of time evolution of NH_3 and CO_2 obtained by different methods for primal multiplicity, including minimization of the total flux (MTF), minimization of the number of active reactions (MNAR), hierarchical optimization (HO) and WPDM. For confidentiality, all concentrations are divided by the initial glutamate concentration so that concentrations are dimensionless.

3.6 Conclusions

The multiplicity of optimal solutions is common for dynamic flux balance analysis models since the problems are often under-determined. The existence of multiple solutions results in infinite possible time trajectories of metabolites' concentrations. Thus, multiplicity poses a challenge for the application of DFBA in model-based control, estimation, monitoring, and optimization.

This paper proposes the use of a variant of the interior-point algorithm referred to as the WPDM where approximation to a particular optimum among all possible optima can be controlled by a proper choice of interior-point weights. The uniqueness, approximation, and continuity of WPDM are proven mathematically. The algorithm is computationally more expensive as compared to other methods since it requires the solution of a set of nonlinear equations by Newton's method. However, while the computations will extend offline calibration of the model they are not a major limiting factor for online applications with a fixed set of weights.

The methodology is illustrated for a DFBA model of *B. pertussis*. It is shown that the choice of interior-point weights in the proposed method can be effectively used to improve the fitting of the model predictions to data as compared to other solvers and other methods used for tackling primal multiplicity.

Acknowledgements

This work was supported NSERC Discovery Grants Program under grant 50503-10882, Sanofi-Pasteur and Mitacs.

Chapter 4

A Type of Set Membership Estimation Designed for Dynamic Flux Balance Models

4.1 Overview

[1]Dynamic flux balance models (DFBM) are used in this study to infer metabolite concentrations that are difficult to measure online. The concentrations are estimated based on a few available measurements. To account for uncertainty in initial conditions the DFBM is converted into a variable structure system based on multiparametric linear programming (mpLP) where different regions of the state space are described by correspondingly different state space models. Using this variable structure system a special set membership based estimation approach is proposed to estimate unmeasured concentrations from few available measurements. For unobservable concentrations upper and lower bounds are estimated. The proposed set membership estimation has been applied to batch fermentation of *E.coli* based on DFBM.

[1]Adapted from Shen, X., & Budman, H. (2021). Set Membership Estimation with Dynamic Flux Balance Models. Processes, 9(10), 1762.

4.2 Introduction

The increasing demand for bio-pharmaceutical products requires continuous improvement in monitoring and control strategies for fermentation processes. Model-based control and optimization strategies are crucial to boost productivity. Unlike traditional unstructured biochemical models, dynamic flux balance models (DFBM) have gained increasing attention since they contain more detailed information about the distribution of metabolic fluxes [86, 46]. The strength of DFBM relies on its use of stoichiometric information about the cell metabolic network. The use of this information often results in models that require a smaller number of parameters as compared to other types of modeling approaches and thus are less prone to over-fitting. However, regardless of the choice of model, monitoring, and control of industrial fermentation processes remains challenging because feedback control strategies require many states to be measured online. In reality, most states cannot be measured online either due to the expense of measuring equipment and its maintenance or the lack of online measurement devices [105, 52, 25]. Some states, including the concentration of amino acids, metals, vitamins, ATP, and precursors have great effect on the fermentation process but are either difficult or impossible to be measured online.

In this research, a set membership estimation approach is proposed for nonlinear systems described by DFBM models. The DFBM is converted into a variable structure system composed of several continuous systems in different regions of state space by multiparametric linear programming. To address the lack of measurements an Extended Kalman Filter (EKF) is used to estimate nominal values of some states which are important for determining metabolic fluxes. Then, a set membership estimation algorithm is applied for DFBM to estimate the bounds of all states. A detector is proposed to detect the switch between different subsystems.

The paper is organized as follows. Section 2.1 introduces the background of DFBM. Section 2.2 describes the use of multiparametric linear programming to convert the DFBM into a variable structure system composed of subsystems. Section 2.3 describes the EKF used to estimate some states which are important for determining metabolic fluxes. Section 2.4 presents the main ideas of set propagation and error compensation for the calculation of states' bounds. Section 2.5 presents the algorithm for detecting the switch between different subsystems. Section 3 provides the application of the proposed techniques to the batch fermentation of *E. coli*. Section 4 presents a Discussion of the results followed by Conclusions.

4.3 Materials and Methods

4.3.1 Dynamic Flux Balance Models

Dynamic flux balance models (DFBM) are structured genome-based metabolic models developed from flux balance models. The key assumption of DFBM is that the cells act as agents distributing resources through metabolic reaction networks to boost a biological objective, e.g. growth rate [86]. Accordingly, the DFBM is formulated as an optimization problem. In the literature [64], both dynamic and static optimization approaches are reported. In the dynamic approach, the nonlinear programming problem is solved over a relatively large time period which is computationally expensive and thus less convenient for uncertainty propagation. In this investigation, a static optimization approach is adopted for its simplicity. DFBM is interpreted as a local linear programming problem to maximize a biological objective. In terms of the dynamics of intracellular metabolites, there are two types of DFBM models in the literature. One type of DFBM differentiates intracellular and extracellular environments and assumes that the intracellular metabolic reactions are fast enough such as it can be assumed at a quasi-steady state [46, 45]. Accordingly, only the extracellular metabolites and the biomass are described by dynamic state equations. It has been argued that the intracellular metabolite concentrations are not constant and may change over time [34]. Accordingly, there is a second type of DFBM, used in the current study, which does not differentiate between intracellular and extracellular compartments and the dynamics of all the metabolites are considered [64, 13]. The governing equations of DFBM are based on discretized mass balances for all metabolites and these are defined by Eq. (4.1).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B}\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{A}\boldsymbol{v_k} + \boldsymbol{h} \tag{4.1a}$$

$$\boldsymbol{y_k} = \boldsymbol{C}\boldsymbol{x_k} + \boldsymbol{r_k} \tag{4.1b}$$

$$\boldsymbol{x_0} \in \mathcal{P}_0 \tag{4.1c}$$

$$\boldsymbol{r_k} \sim TN(\boldsymbol{0}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}) \quad k = 0, 1, 2 \cdots \tag{4.1d}$$

Where $\boldsymbol{x_k}$ is a vector of n_x state variables at time step k. The state vector $\boldsymbol{x}$ includes concentrations of metabolites and biomass x_{bio}. $\boldsymbol{y}$ is a vector of n_y measured variables. $\boldsymbol{B} \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ is a constant diagonal matrix with diagonal elements b_j, $j = 1, \cdots, n_x$. Δt is constant discrete time step size. $\boldsymbol{A} \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_{rct}}$ is stoichiometry coefficient matrix, where n_{rct} is the number of reactions considered in the metabolic network. $\boldsymbol{v} \in \mathbb{R}^{n_{rct}}$ is the metabolic flux vector and its calculation is discussed below. $\boldsymbol{h} \in \mathbb{R}^{n_x}$ is a constant vector. The initial state $\boldsymbol{x_0}$ is assumed to be bounded by a finite polyhedron $\mathcal{P}_0$ as Eq. (4.1c).

The underlying assumption is that in practice the initial concentrations of the culture medium components are known to be within specific ranges of values $\mathcal{P}_0$. This assumption is based on the fact that some variation in media formulation occurs due to human factors and variability in raw materials. Hence, this research focuses on the initial uncertainty and we assume all parameters in the state equations to be known accurately. In other words, the method proposed in this research cannot deal with model structure uncertainty like uncertainty in matrix A. But the method can be extended to deal indirectly with uncertainty in parameter $\boldsymbol{\theta}$ defined in the following paragraphs.

$\boldsymbol{r_k} \in \mathbb{R}^{n_y}$ are measurement noise vectors which elements follow the truncated multivariate normal distribution (TN) [112, 11]. The probability density function p for $TN(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u})$ are defined as per Eq. (4.2).

$$p(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}) = \frac{exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\}}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\}} \tag{4.2}$$

For $\boldsymbol{r_k}$, the mean vector of TN is $\mathbf{0} \in \mathbb{R}^{n_y}$; the covariance is $\boldsymbol{\Sigma} \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_y}$; corresponding variance vector is $\boldsymbol{\sigma}^2 \in \mathbb{R}^{n_y}$; lower bound and upper bound are $\boldsymbol{l} \in \mathbb{R}^{n_y}$ and $\boldsymbol{u} \in \mathbb{R}^{n_y}$ respectively. $|\cdot|$ indicates the absolute value of a vector. It is assumed that $|\boldsymbol{l}| \leq 3\boldsymbol{\sigma}$ and $|\boldsymbol{u}| \leq 3\boldsymbol{\sigma}$, which indicate that the absolute values of the lower bound and upper bound respectively are within the range of $3\boldsymbol{\sigma}$. For simplicity, the current study assumes the process noise to be zero. Process noise could be included as an additional state but this is beyond the scope of the current work.

Following the assumption that the cell allocates resources optimally, the metabolic flux $\boldsymbol{v}$ vector at each time step is obtained by solving a linear programming (LP) problem, defined by Eq. (4.3).

$$\max_{\boldsymbol{v_k}} \quad \boldsymbol{c}^T\boldsymbol{v_k} \tag{4.3a}$$

$$\text{subject to} \quad \boldsymbol{G}\boldsymbol{v_k} \leq \boldsymbol{F}\boldsymbol{\theta_k}(\boldsymbol{x_k}) + \boldsymbol{z} \tag{4.3b}$$

where $\boldsymbol{c} \in \mathbb{R}^{n_{rct}}$, $\boldsymbol{F} \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_\theta}$, $\boldsymbol{z} \in \mathbb{R}^{n_G}$, $\boldsymbol{G} \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_{rct}}$, $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^{n_\theta}$. n_G is the number of linear constraints. The parameter vector $\boldsymbol{\theta}$ is a nonlinear vector-valued function of states $\boldsymbol{x}$. n_θ denotes the number of elements in the parameter vector $\boldsymbol{\theta}$. Usually, each element θ is only a function of two states at most and one of these two states is biomass concentration. $\boldsymbol{\Theta}$ denotes the parameter space where the optimal solution of the LP resides. Eq.(4.3a) denotes the objective of the LP that cells are optimizing where the most commonly used objective is the biomass production rate, i.e. growth rate. Thus, cells try to maximize growth rate by allocating limited resources. The LHS (left-hand-side) in

Eq.(4.3b) describes either the rate of change of metabolite concentrations or the change of metabolite concentrations over a discretization time step Δt. Matrices $\boldsymbol{G}$ are constant matrices containing information on the stoichiometry of reactions. RHS in Eq.(4.3b) is a function of $\boldsymbol{x_k}$, denoting the metabolic reaction bounds for each step. The matrix $\boldsymbol{F}$ is a matrix in which elements are part of the right-hand side of the constraints that are functions of states at the previous time interval. $\boldsymbol{z}$ is a vector containing constant values such as constant uptake rate limits. Therefore, linear constraints of flux $\boldsymbol{v}$ in Eq.(4.3b) are reaction rate limits or bounds on available resources (nutrients). Numerical examples of these matrices and vectors are shown for the *E.coli* model in the results section.

4.3.2 Multiparametric Linear Programming for DFBM

Multiparametric Linear Programming [2, 10, 84]

While set-based methods are available for uncertainty propagation for linear state space equations, these methods are not directly applicable to DFBM. The reason is that the fluxes used in the state equations are obtained from an LP and thus the problem is non-linear due to the nonlinear function $\boldsymbol{\theta(x)}$ and the occurrence of different sets of active constraints. To tackle the dependency of the state equations on the LP, the concept of multiparametric linear programming (mpLP) is used to convert the DFBM into a variable structure system that is composed of subsystems. Multiparametric linear programming divides the parameter space ($\boldsymbol{\Theta}$) into different regions corresponding to different sets of active constraints and generates explicit expressions for calculating optimal solutions ($\boldsymbol{v}$) for each region [2, 10, 84].

Let's assume a given optimal solution $\boldsymbol{v}$ of the LP (Eq. (4.3)) where subscript $\mathcal{A}$ and $\mathcal{I}$ denote indices of active and inactive constraints respectively. Using this notation Eq. (4.3b) is decomposed into two parts, equalities $\boldsymbol{G_{\mathcal{A}} v_k = F_{\mathcal{A}} \theta_k(x_k) + z_{\mathcal{A}}}$ and inequalities $\boldsymbol{G_{\mathcal{I}} v_k \leq F_{\mathcal{I}} \theta_k(x_k) + z_{\mathcal{I}}}$. Without loss of generality, let's assume that $\boldsymbol{G_{\mathcal{A}}}$ is linearly independent (linear redundant rows can always be removed by Gaussian elimination). Let $\boldsymbol{H = G_{\mathcal{A}}^{-1} F_{\mathcal{A}}}$ and $\boldsymbol{g = G_{\mathcal{A}}^{-1} z_{\mathcal{A}}}$, then the optimal solution can be obtained by Eq. (4.4). Following the literature and our previous studies, for a given $\boldsymbol{\theta}$, multiple optimal solutions can coexist [78, 98]. In other words, multiple Eq. (4.4) can coexist which results in different ways to divide the parameter space $\boldsymbol{\Theta}$. When such a multiplicity issue occurs it results in different time trajectories. For simplicity, multiplicity is not addressed in the current study and it is addressed in a separate work by different methods from the one presented here.

The inverse of $\boldsymbol{G_{\mathcal{A}}}$ exists because here we assume that the solution to the LP is unique.

$$\boldsymbol{v_k} = \boldsymbol{H}\boldsymbol{\theta_k}(\boldsymbol{x_k}) + \boldsymbol{g} \tag{4.4}$$

Substituting Eq. (4.4) into the inequality constraints results in Eq. (4.5).

$$(\boldsymbol{G_{\mathcal{I}}}\boldsymbol{H} - \boldsymbol{F_{\mathcal{I}}})\boldsymbol{\theta_k}(\boldsymbol{x_k}) < \boldsymbol{z_{\mathcal{I}}} - \boldsymbol{G_{\mathcal{I}}}\boldsymbol{g} \tag{4.5}$$

Eq. (4.5) defines a polyhedral region of $\boldsymbol{\theta}$ where the existence of the optimal solution is ensured by Eq. (4.4). The region defined by Eq. (4.5) is referred to as a critical region in the multiparametric programming literature. Different critical regions are defined by different combinations of $\mathcal{A}$ and $\mathcal{I}$. Then, the entire parameter space $\boldsymbol{\Theta}$ can be decomposed into connected critical regions denoted by $\{\boldsymbol{\Theta}^i\}$, $i = 1, \cdots, n_{\boldsymbol{\Theta}}$. $n_{\boldsymbol{\Theta}}$ denotes the total number of critical regions in $\boldsymbol{\Theta}$. In practice, critical regions that are very small are ignored and assumed to be covered by the adjacent critical region. Correspondingly, superscript i is used to denote the i-th critical region. Assume for a specific $\boldsymbol{\theta} \in \boldsymbol{\Theta}^i$, the optimal flux $\boldsymbol{v}$ vector can be calculated analytically by $\boldsymbol{v_k^i} = \boldsymbol{H^i}\boldsymbol{\theta_k} + \boldsymbol{g^i}$ thus bypassing the need for solving the LP.

By substituting the optimizer equation $\boldsymbol{v_k^i} = \boldsymbol{H^i}\boldsymbol{\theta_k} + \boldsymbol{g^i}$ into Eq. (4.1a), we obtained a set of governing state equations as per Eq. (4.6). Since different $\boldsymbol{\theta_k}$ are within different critical regions as Eq. (4.6b), each critical region corresponds to different state equations Eq. (4.6a). Thus the set $\{\boldsymbol{\Theta^i}\}$ defines a family of state space models and this family is referred to as a variable structure system. A variable structure system is a piecewise continuous system composed of subsystems where each subsystem corresponds to a different region of the state space. And the region of the state space corresponding to a specific subsystem is referred to as a critical region. Each subsystem is described by a different set of state equations. Accordingly, the state equations need to be changed as soon as the states enter a new critical region. Here, the superscript i denotes the *i-th* subsystem corresponding to critical region $\boldsymbol{\Theta^i}$. Eqs. (4.6c)-(4.6e) remain the same form as Eqs. (4.1b)-(4.1d).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B}\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{A}(\boldsymbol{H^i}\boldsymbol{\theta_k}(\boldsymbol{x_k}) + \boldsymbol{g^i}) + \boldsymbol{h} \tag{4.6a}$$

$$\boldsymbol{\theta_k}(\boldsymbol{x_k}) \in \boldsymbol{\Theta}^i \qquad i = 1, \cdots, n_{\Theta} \tag{4.6b}$$

$$\boldsymbol{y_k} = \boldsymbol{C}\boldsymbol{x_k} + \boldsymbol{r_k} \tag{4.6c}$$

$$\boldsymbol{x_0} \in \mathcal{P}_0 \tag{4.6d}$$

$$\boldsymbol{r_k} \sim TN(\boldsymbol{0}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}) \quad k = 0, 1, 2 \cdots \tag{4.6e}$$

Reaction Rate Estimability

To further simplify the system described by Eq. (4.6) it is possible to exploit the sparseness (Columns are zeros) of the $\boldsymbol{H}$ matrix. For instance, to take advantage of zero columns of $\boldsymbol{H}$, Eq. (4.4) can be re-written as shown in Eq. (4.7). For conciseness, the subscript k is omitted here because Eq. (4.7) applies for all time steps.

$$\boldsymbol{v^i} = \boldsymbol{H^i}\boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{g^i} = \begin{bmatrix} \boldsymbol{H_N^i} & \boldsymbol{H_Z^i} \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta_N^i}(\boldsymbol{x_N^i}) \\ \boldsymbol{\theta_Z^i}(\boldsymbol{x}) \end{bmatrix} + \boldsymbol{g^i} = \boldsymbol{H_N^i}\boldsymbol{\theta_N^i}(\boldsymbol{x_N^i}) + \boldsymbol{g^i} \tag{4.7}$$

In Eq. (4.7) N and Z denote the indices of the nonzero and zero columns of the $\boldsymbol{H}$ matrix respectively. Because $\boldsymbol{H_Z}$ is a submatrix containing the zero columns of $\boldsymbol{H}$, the flux $\boldsymbol{v}$ is only a function of parameters $\boldsymbol{\theta_N}(\boldsymbol{x_N})$ according to Eq. (4.7). Moreover, while the parameters $\boldsymbol{\theta}$ are a function of states $\boldsymbol{x}$ (see Eq. (4.1) and (4.3)), only some elements of $\boldsymbol{x}$ actually determine the entire flux vector $\boldsymbol{v}$. The vector $\boldsymbol{x_N}$ contains, according to Eq. (4.7), the states that determine the flux vector. Notice that for different critical regions flux-determining vector $\boldsymbol{x_N}$ contains different states. Therefore, Eq. (4.6a) can be simplified into Eq. (4.8).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B}\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{A}(\boldsymbol{H_N^i}\boldsymbol{\theta_N^i}(\boldsymbol{x_{N,k}^i}) + \boldsymbol{g^i}) + \boldsymbol{h} \tag{4.8}$$

The biological interpretation of the flux-determining state vector $\boldsymbol{x_N}$ is that only some resources are limiting the growth of cells, either because they are limited or because the activity of enzymes in the related reactions (fluxes) is limiting. As the fermentation progresses, the states transit into new critical regions from old critical regions. Different critical regions can be interpreted as different metabolic stages where $\boldsymbol{x_N}$ are different. Similar interpretations have been reported in [2] in the context of steady-state flux balance analysis.

In Eq. (4.8), the term $\Delta t x_{bio,k}\boldsymbol{A}(\boldsymbol{H_N^i}\boldsymbol{\theta_N^i}(\boldsymbol{x_{N,k}^i}) + \boldsymbol{g^i})$ denotes the change of metabolite concentrations contributed by metabolic reactions. Therefore, the reaction rates are $x_{bio,k}\boldsymbol{A}(\boldsymbol{H_N^i}\boldsymbol{\theta_N^i}(\boldsymbol{x_{N,k}^i}) + \boldsymbol{g^i})$. It is noted that this nonlinear reaction rate term is not only a function of the flux-determining states vector $\boldsymbol{x_N}$ but also of biomass concentration x_{bio}, because the fluxes are defined per unit biomass, i.e. more biomass demands more nutrients to satisfy the requirement of the growth. Once the states that determine the reaction rates, i.e. the states $\boldsymbol{x_N}$ together with the value of x_{bio}, can be estimated, the estimation problem can be simplified greatly. Since in some cases $\boldsymbol{x_N}$ contains x_{bio} but in some cases it does not, we define a reaction-rate-determining state vector $\boldsymbol{x_M}$ in Eq. (4.9). Hence,

the reaction-rate-determining state vector $\boldsymbol{x_M}$ always contains the flux-determining states $\boldsymbol{x_N}$ and the biomass state x_{bio} without any redundancy.

$$\boldsymbol{x_M} = \begin{cases} \boldsymbol{x_N}, & \text{if } \boldsymbol{x_N} \text{ contains the biomass state } x_{bio} \text{ .} \\ \begin{bmatrix} \boldsymbol{x_N} \\ x_{bio} \end{bmatrix}, & \text{otherwise.} \end{cases} \tag{4.9}$$

The vector $\boldsymbol{x_M}$ for critical region $\boldsymbol{\Theta^i}$ is denoted by $\boldsymbol{x_M^i}$. We define reaction rate estimability as the ability to determine the reaction rates $x_{bio,k}\boldsymbol{A}(\boldsymbol{H_N^i}\boldsymbol{\theta_N^i}(\boldsymbol{x_{N,k}^i}) + \boldsymbol{g^i})$ in the metabolic networks which are needed for the calculation of Eq. (4.8). Following the above, once reaction-rate-determining state vector $\boldsymbol{x_M}$ at time step k can be estimated, the dynamic evolution of the culture at step $k+1$ as per Eq. (4.8) can be predicted. Also, it should be noticed that it is not necessary to measure all the reaction-rate-determining states for reaction rate estimability and instead some states can be estimated by an observer from available measurements. However, if an observer is used to estimate $\boldsymbol{x_M^i}$, some particular combination of measurements is necessary for the observability of $\boldsymbol{x_M^i}$. Considering different measurement combinations Ω_1^i, Ω_2^i... for critical region $\boldsymbol{\Theta^i}$, only some combinations provide full observability of $\boldsymbol{x_M^i}$. Let define $\Omega_{\mathcal{O}}^i$ as a family of sets of measurements, which contains all measurement combinations that fulfill the observability of $\boldsymbol{x_M^i}$.

Although many different critical regions and corresponding combinations of measurements could be considered, in practice, the possibilities will be limited because industrial fermentations usually operate in a narrow range of operating conditions. Thus, the dynamic trajectories of states only pass through a limited set of critical regions. Assume for $\forall \boldsymbol{x_0} \in \mathcal{P}_0$, the set of critical regions that the trajectories traverse are Γ. Then, the minimum set of measurements required for reaction rate estimability of the critical region set Γ is Ω_Γ as per Eq. (4.10).

$$\Omega_\Gamma = \min_j |\bigcup_i \Omega_j^i| \tag{4.10a}$$

$$\text{subject to} \qquad i \in \Gamma \tag{4.10b}$$

$$\Omega_j^i \in \Omega_{\mathcal{O}}^i \tag{4.10c}$$

where $|\cdot|$ is the cardinality of a finite countable set, i.e. the number of elements of a set. In Eq. (4.10c), $\Omega_j^i \in \Omega_{\mathcal{O}}^i$ indicates that the measurement combination Ω_j^i can fulfill the observability of reaction-rate-determining states $\boldsymbol{x_M^i}$ of critical region $\boldsymbol{\Theta^i}$. If all states in set Ω_Γ are measured, the reaction rate term of any trajectory starting from $\mathcal{P}_0$ can be

estimated by an observer. In other words, although $\boldsymbol{x}_{\boldsymbol{M}}^{\boldsymbol{i}}$ in different critical regions may be different requiring different measurements for observability, $\boldsymbol{x}_{\boldsymbol{M}}^{\boldsymbol{i}}$ is always observable if the chosen set of measurements satisfies Eq. (4.10c).

4.3.3 Extended Kalman Filter (EKF)

Using the minimum required set of measurements Ω_Γ defined in Eq. (4.10c), $\boldsymbol{x}_{\boldsymbol{M}}$ can be estimated by an observer. $\boldsymbol{x}_{\boldsymbol{M}}$ corresponds to the observable subspace of the governing equation (Eq. (4.1)) for each critical region. The state equation of the observable subspace for critical region $\boldsymbol{\Theta}^{\boldsymbol{i}}$ is given by Eq. (4.11).

$$\boldsymbol{x}_{\boldsymbol{M,k+1}}^{\boldsymbol{i}} = \boldsymbol{f}^i(\boldsymbol{x}_{\boldsymbol{M,k}}^{\boldsymbol{i}}) = \boldsymbol{B}\boldsymbol{x}_{\boldsymbol{M,k}}^{\boldsymbol{i}} + \Delta t x_{bio,k}\boldsymbol{A}_{\boldsymbol{M}}(\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}}\boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{i}}(\boldsymbol{x}_{\boldsymbol{N,k}}^{\boldsymbol{i}}) + \boldsymbol{g}^{\boldsymbol{i}}) + \boldsymbol{h}_{\boldsymbol{M}} \tag{4.11a}$$

$$\boldsymbol{y}_{\boldsymbol{k}} = \boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}\boldsymbol{x}_{\boldsymbol{M,k}}^{\boldsymbol{i}} + \boldsymbol{r}_{\boldsymbol{k}} \tag{4.11b}$$

$$\boldsymbol{r}_{\boldsymbol{k}} \sim TN(\boldsymbol{0}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}) \quad k = 0, 1, 2\cdots \tag{4.11c}$$

Where $\boldsymbol{x}_{\boldsymbol{N,k}}^{\boldsymbol{i}}$ and $\boldsymbol{x}_{\boldsymbol{M,k}}^{\boldsymbol{i}}$ are the flux-determining state vector and the reaction-rate-determining state vector for critical region $\boldsymbol{\Theta}^{\boldsymbol{i}}$ respectively; $\boldsymbol{A}_{\boldsymbol{M}}$ is the stoichiometry submatrix corresponding to $\boldsymbol{x}_{\boldsymbol{M}}$; similarly $\boldsymbol{h}_{\boldsymbol{M}}$ is sub-vector of $\boldsymbol{h}$ corresponding to $\boldsymbol{x}_{\boldsymbol{M}}$. It should be noticed that for different critical regions, $\boldsymbol{x}_{\boldsymbol{M}}$ involves different states. Accordingly, each critical region requires the use of a different EKF. Also, it should be noticed that the $\boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}$ matrices are different for each critical region but the measured variables Ω_Γ are the same since the same sensors are used for the entire fermentation.

To estimate $\boldsymbol{x}_{\boldsymbol{M}}^{\boldsymbol{i}}$, a standard EKF is used due to its effective and simple structure [103]. The estimate $\hat{\boldsymbol{x}}_{\boldsymbol{M,k}}^{\boldsymbol{i}}$ and covariance $\boldsymbol{P}_{\boldsymbol{k}}^{\boldsymbol{i}}$ of $\boldsymbol{x}_{\boldsymbol{M}}^{\boldsymbol{i}}$ for critical region $\boldsymbol{\Theta}^{\boldsymbol{i}}$ are described by Eq. (4.12a) and Eq. (4.12b) respectively.

$$\hat{\boldsymbol{x}}_{\boldsymbol{M,k}}^{\boldsymbol{i}} = \boldsymbol{f}^i(\hat{\boldsymbol{x}}_{\boldsymbol{M,k-1}}^{\boldsymbol{i}}) + \boldsymbol{K}_{\boldsymbol{k}}(\boldsymbol{y}_{\boldsymbol{k}} - \boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}\hat{\boldsymbol{x}}_{\boldsymbol{M,k}}^{\boldsymbol{i}}) \tag{4.12a}$$

$$\boldsymbol{P}_{\boldsymbol{k}}^{\boldsymbol{i}^{-1}} = \boldsymbol{\Phi}_{\boldsymbol{k-1}}^{\boldsymbol{i}}\boldsymbol{P}_{\boldsymbol{k-1}}^{\boldsymbol{i}}\boldsymbol{\Phi}_{\boldsymbol{k-1}}^{\boldsymbol{i}}{}^T + \boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}{}^T(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T)^{-1}\boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}} \tag{4.12b}$$

Where

$$\boldsymbol{K}_{\boldsymbol{k}} = \boldsymbol{\Phi}_{\boldsymbol{k-1}}^{\boldsymbol{i}}\boldsymbol{P}_{\boldsymbol{k-1}}^{\boldsymbol{i}}\boldsymbol{\Phi}_{\boldsymbol{k-1}}^{\boldsymbol{i}}{}^T\boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}{}^T(\boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}\boldsymbol{\Phi}_{\boldsymbol{k-1}}^{\boldsymbol{i}}\boldsymbol{P}_{\boldsymbol{k-1}}\boldsymbol{\Phi}_{\boldsymbol{k-1}}^{\boldsymbol{i}}{}^T\boldsymbol{C}_{\boldsymbol{M}}^{\boldsymbol{i}}{}^T + \boldsymbol{\Sigma}\boldsymbol{\Sigma}^T)^{-1} \tag{4.13a}$$

$$\boldsymbol{\Phi}_{\boldsymbol{k}}^{\boldsymbol{i}} = \frac{\partial \boldsymbol{f}^i}{\partial \boldsymbol{x}_{\boldsymbol{M}}^{\boldsymbol{i}}}(\hat{\boldsymbol{x}}_{\boldsymbol{M,k}}^{\boldsymbol{i}}) \tag{4.13b}$$

The measurement noise is assumed to be a truncated multivariate normal distribution

as Eq. (4.11c). This assumption is needed for estimating finite bounds as explained in the following section. Recall in Eq. (4.2) that $|\boldsymbol{l}| \leq 3\boldsymbol{\sigma}$ and $|\boldsymbol{u}| \leq 3\boldsymbol{\sigma}$, the lower and upper bounds are located within the range of $3\boldsymbol{\sigma}$. The covariance matrix $\boldsymbol{P_k}$ is always overestimated to ensure boundedness. Although the EKF resulting from this assumption is sub-optimal it is still sufficient to estimate $\boldsymbol{x^i_M}$.

4.3.4 Set Propagation and Error Compensation

Since the minimum set of measurements defined by Eq. (4.10) can only ensure the observability of $\boldsymbol{x_M}$, the estimation of other states needs different estimation strategies. The idea is to exploit the a priori knowledge of the initial ranges of initial conditions to estimate all states. Instead of predicting specific values of states, the set membership estimation (SME) approach is used to predict sets containing all possible states by a series of set operations. These set operations usually include linear mapping, projection, translation, Minkowski addition, intersection, union, and outer approximation. In this research, all sets and multiparametric linear programming operations are performed with the Multi-Parametric Toolbox 3.0 (https://www.mpt3.org/) [43] and MATLAB R2018a. The *E.coli* example can be found online (https://github.com/SetMembershipEstimationDFBM∼/E.coliExample). For DFBM, SME propagates the initial set $\mathcal{P}_0$ by affine mapping as Eq. (4.14). Affine mapping involves two operations: linear mapping of the previous set and translation.

$$\hat{\mathcal{X}}_{k+1} \approx \underbrace{\boldsymbol{B}\hat{\mathcal{X}}_k}_{\text{linear mapping}} + \underbrace{\Delta t \hat{x}_{bio,k} \boldsymbol{A}(\boldsymbol{H^i_N}\boldsymbol{\theta^i_N}(\hat{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{N,k}}) + \boldsymbol{g^i}) + \boldsymbol{h}}_{\text{translation}} \tag{4.14}$$

Where $\hat{\mathcal{X}}_k$ represents the set of states at time step k and $\hat{\mathcal{X}}_0 = \mathcal{P}_0$, i.e. the set of initial conditions assumed to be known. In Eq. (4.14), the translation term is approximated by using the estimate $\hat{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{M,k}}$ obtained by the EKF. In the application of EKF, the estimate $\hat{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{M,k}}$ needs several time steps to converge to the true flux-determining states $\boldsymbol{x^i_{M,k}}$. Thus the SME described by Eq. (4.14) may underestimate bounds while the EKF is converging. To mitigate this problem a correction is implemented to compensate for the estimate error as described below. Since no extra information is available, the compensation of the estimate error is based on the worst-case scenario.

The error in the estimate incurred by the observer for critical region $\boldsymbol{\Theta^i}$ is $\boldsymbol{e^i_M} = \boldsymbol{x^i_{M,k}} - \hat{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{M,k}}$. Since $\boldsymbol{x^i_{M,k}}$ always contains biomass $x^i_{bio,k}$ and $\boldsymbol{x^i_{N,k}}$, the corresponding estimate errors are defined as $\boldsymbol{e^i_{N,k}} = \boldsymbol{x^i_{N,k}} - \hat{\boldsymbol{x}}^{\boldsymbol{i}}_{\boldsymbol{N,k}}$ and $e^i_{bio} = x^i_{bio,k} - \hat{x}^i_{bio,k}$. Let's assume

that the function $\boldsymbol{\theta}$ is first-order differentiable and define Jacobian matrix $\boldsymbol{\psi}_{\boldsymbol{k}}^{\boldsymbol{i}}$.

$$\boldsymbol{\psi}_{\boldsymbol{k}}^{\boldsymbol{i}} = \frac{\partial \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{i}}}{\partial \boldsymbol{x}_{\boldsymbol{N}}^{\boldsymbol{i}}}(\hat{\boldsymbol{x}}_{\boldsymbol{N},\boldsymbol{k}}^{\boldsymbol{i}}) \tag{4.15}$$

Substituting the estimate error $\boldsymbol{e}_{\boldsymbol{k}}^{\boldsymbol{i}}$, e_{bio}^{i} and Jacobian matrix $\boldsymbol{\psi}_{\boldsymbol{k}}^{\boldsymbol{i}}$ into Eq. (4.8), a corrected state equation that accounts for the estimate error is obtained as Eq. (4.16). Eq. (4.16) uses a first-order approximation to account for the state deviation $\boldsymbol{\epsilon}_{\boldsymbol{k}}^{\boldsymbol{i}}$ caused by the estimate error $\boldsymbol{e}_{\boldsymbol{M},\boldsymbol{k}}^{\boldsymbol{i}}$ while the EKF is converging. The error compensation based on linearization provides satisfactory bounds because the error between the estimate and measured is small and decreases quickly due to the convergence of EKF.

$$\boldsymbol{x}_{\boldsymbol{k+1}} = \boldsymbol{B}\boldsymbol{x}_{\boldsymbol{k}} + \Delta t \hat{x}_{bio,k}\boldsymbol{A}(\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}}\boldsymbol{\psi}_{\boldsymbol{k}}^{\boldsymbol{i}}\hat{\boldsymbol{x}}_{\boldsymbol{N},\boldsymbol{k}}^{\boldsymbol{i}} + \boldsymbol{g}^{\boldsymbol{i}}) + \boldsymbol{h} + \boldsymbol{\epsilon}_{\boldsymbol{k}}^{\boldsymbol{i}} \tag{4.16a}$$

$$\boldsymbol{\epsilon}_{\boldsymbol{k}}^{\boldsymbol{i}} = \boldsymbol{D}_{\boldsymbol{k}}\boldsymbol{e}_{\boldsymbol{N},\boldsymbol{k}}^{\boldsymbol{i}} + e_{bio,k}\boldsymbol{M}_{\boldsymbol{k}}\boldsymbol{e}_{\boldsymbol{N},\boldsymbol{k}}^{\boldsymbol{i}} + \boldsymbol{L}_{\boldsymbol{k}}e_{bio,k} \tag{4.16b}$$

Where

$$\boldsymbol{D}_{\boldsymbol{k}} = \hat{x}_{bio,k}\Delta t\boldsymbol{A}\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}}\boldsymbol{\psi}_{\boldsymbol{k}}^{\boldsymbol{i}} + \boldsymbol{h} \tag{4.17a}$$

$$\boldsymbol{M}_{\boldsymbol{k}} = \Delta t\boldsymbol{A}\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}}\boldsymbol{\psi}_{\boldsymbol{k}}^{\boldsymbol{i}} \tag{4.17b}$$

$$\boldsymbol{L}_{\boldsymbol{k}} = \Delta t\boldsymbol{A}(\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}}\boldsymbol{\theta}_{\boldsymbol{N},\boldsymbol{k}}^{\boldsymbol{i}}(\hat{\boldsymbol{x}}_{\boldsymbol{N},\boldsymbol{k}}^{\boldsymbol{i}}) + \boldsymbol{g}^{\boldsymbol{i}}) \tag{4.17c}$$

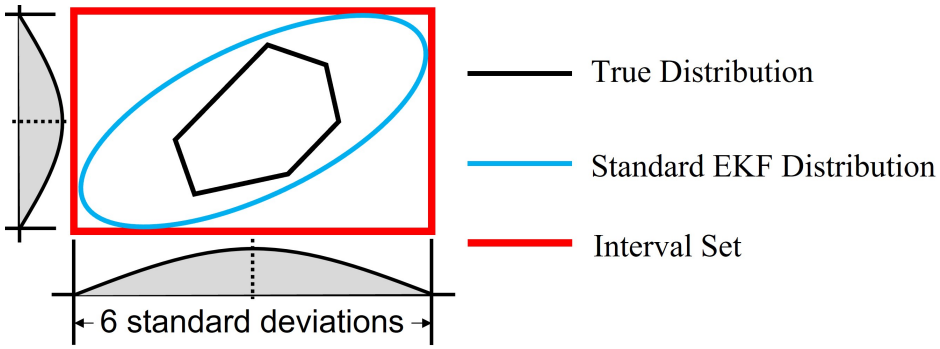


Figure 4.1: Illustration of the interval set containing the distribution of states.

To formulate an error compensation operation scheme several set operations are introduced first as follows. The n-dimensional interval set is $\mathcal{S}(\boldsymbol{p},\boldsymbol{q})$ with lower bound $\boldsymbol{p}$ and upper bound $\boldsymbol{q}$ as $\mathcal{S}(\boldsymbol{p},\boldsymbol{q}) = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{p} \leq \boldsymbol{x} \leq \boldsymbol{q}\}$. The outer approximation operation

$\mathcal{Q}(\cdot)$ of a bounded set $\mathcal{W}$ is denoted by $\mathcal{Q}(\mathcal{W})$, which involves the mapping of the set $\mathcal{W}$ to a new interval set. If the infimum and supremum are denoted by $\inf(\cdot)$ and $\sup(\cdot)$ respectively, the outer approximation of the set $\mathcal{W}$ is $\mathcal{Q}(\mathcal{W}) = \mathcal{S}(\inf(\mathcal{W}), \sup(\mathcal{W}))$. The operator $\oplus$ is the Minkowski addition of two sets. For example, for two sets α and β, $\alpha \oplus \beta = \{\boldsymbol{a} + \boldsymbol{b} : \boldsymbol{a} \in \alpha, \boldsymbol{b} \in \beta\}$.

Notice that the diagonal elements of $\boldsymbol{P_k^i}$ are the variances of each state. Then, if the standard deviation of $\boldsymbol{e_{N,k}^i}$ is $\boldsymbol{\eta_{N,k}^i}$ and of $e_{bio,k}^i$ is $\eta_{bio,k}^i$, two interval sets $\mathcal{E}_{N,k}$ and $\mathcal{E}_{bio,k}$ can be defined to bound $\boldsymbol{\eta_{N,k}^i}$ and $\eta_{bio,k}^i$ respectively based on choice of 3 standard deviation range, as $\boldsymbol{e_{N,k}^i} \in \mathcal{E}_{N,k} = \mathcal{S}(-3\boldsymbol{\eta_{N,k}^i}, 3\boldsymbol{\eta_{N,k}^i})$ and $e_{bio,k}^i \in \mathcal{E}_{bio,k} = \mathcal{S}(-3\eta_{bio,k}^i, 3\eta_{bio,k}^i)$. In Eq. (4.16b), since $|e_{bio,k}^i| < 3\eta_{bio,k}^i$, we have $e_{bio,k}\boldsymbol{M_k}\boldsymbol{e_{N,k}^i} \in 3\eta_{bio,k}^i\boldsymbol{M_k}\mathcal{E}_{N,k}$. Similarly, the other two terms in Eq. (4.16b) can be bounded as $\boldsymbol{D_k}\boldsymbol{e_{N,k}^i} \in \boldsymbol{D_k}\mathcal{E}_{N,k}$ and $\boldsymbol{L_k}e_{bio,k} \in \boldsymbol{L_k}\mathcal{E}_{bio,k}$ respectively. Therefore, the state deviation $\boldsymbol{\epsilon_k^i}$ term can be contained within the interval set $\mathcal{E}_{\epsilon,k}$ according to Eq. (4.18).

$$\boldsymbol{\epsilon_k^i} \in \mathcal{E}_{\epsilon,k} = \mathcal{Q}((\boldsymbol{D_k} + 3\eta_{bio,k}^i\boldsymbol{M_k})\mathcal{E}_{N,k}) \oplus \mathcal{Q}(\boldsymbol{L_k}\mathcal{E}_{bio,k}) \tag{4.18}$$

where the sets $\boldsymbol{D_k}\mathcal{E}_{N,k}$ and $3\eta_{bio,k}^i\boldsymbol{M_k}\mathcal{E}_{N,k}$ occurring in Eq. (4.18) are combined together. On the other hand $\boldsymbol{L_k}\mathcal{E}_{bio,k}$ originates from a different set $\mathcal{E}_{bio,k}$ and thus Minkowski addition must be used to add the different sets. However, linear mapping of interval sets can lead to irregular convex sets. In computational geometry, traditional algorithms that perform Minkowski addition for two convex irregular high-dimensional polytopes are computationally expensive [23]. On the other hand, Minkowski addition of two interval sets is computationally efficient because intervals are axis-aligned. Thus, the operator $\mathcal{Q}(\cdot)$ that converts the irregular set to an axis-aligned set is applied to speed up the computation of the Minkowski addition.

Following the above, the set of states $\hat{\mathcal{X}}_{k+1}$ is bounded by the prior estimate set $\mathcal{P}_{k+1}^-$ according to Eq. (4.19).

$$\mathcal{P}_{k+1}^- = \mathcal{Q}\{\underbrace{\boldsymbol{B}\mathcal{P}_k^+}_{\text{linear mapping}} + \underbrace{\Delta t\hat{x}_{bio,k}\boldsymbol{A}(\boldsymbol{H_N^i}\boldsymbol{\theta_N^i}(\boldsymbol{\hat{x}_{N,k}^i}) + \boldsymbol{g^i}) + \boldsymbol{h}\}}_{\text{translation}} \oplus \mathcal{E}_{\epsilon,k} \tag{4.19a}$$

$$\hat{\mathcal{X}}_{k+1} \subset \mathcal{P}_{k+1}^- \tag{4.19b}$$

Where the set of the posterior estimate is $\mathcal{P}_k^+$. $\boldsymbol{B}\mathcal{P}_k^+$ denotes the scaling of the set $\mathcal{P}_k^+$ by the diagonal matrix $\boldsymbol{B}$. Then the set $\boldsymbol{B}\mathcal{P}_k^+$ is translated by the vector in the big curly brackets. To compensate for the deviation during the convergence of EKF, the interval set $\mathcal{E}_{\epsilon,k}$ is added by Minkowski addition.

Considering the truncated measurement noise $\boldsymbol{r_k} = \boldsymbol{y_k} - \boldsymbol{C}\boldsymbol{x_k}$ is bounded by lower $\boldsymbol{l}$ and upper bounds $\boldsymbol{u}$, let define a set $\mathcal{M}_k = \{\boldsymbol{x_k} \in \mathbb{R}^{n_x} : \boldsymbol{l} < \boldsymbol{y_k} - \boldsymbol{C}\boldsymbol{x_k} < \boldsymbol{u}\}$. Then, the posterior estimate set $\mathcal{P}^+_{k+1}$ is given by Eq. (4.20). In this study, it is assumed that $\mathcal{P}^+_k$ and $\mathcal{P}^-_{k+1}$ are much smaller than the volumes of the critical regions.

$$\mathcal{P}^+_{k+1} = \mathcal{P}^-_{k+1} \bigcap \mathcal{M}_{k+1} \tag{4.20a}$$

$$\hat{\mathcal{X}}_{k+1} \subset \mathcal{P}^+_{k+1} \tag{4.20b}$$

$$\mathcal{P}^+_0 = \mathcal{P}_0 \tag{4.20c}$$

Fig. (4.2) illustrates the set propagation using intervals for an example involving two states, e.g. glucose and biomass concentrations. The initial set $\mathcal{P}_0$ contains all possible initial values of glucose and biomass. Then $\mathcal{P}^+_1$ is generated through set operations by computational geometry algorithms. Since an interval set is used, it is computationally efficient to project the set $\mathcal{P}^+_1$ onto the biomass and glucose axes to obtain the corresponding lower bounds l_{glc}, l_{bio} and upper bounds u_{glc}, u_{bio} as shown in the figure for the set $\mathcal{P}^+_1$.

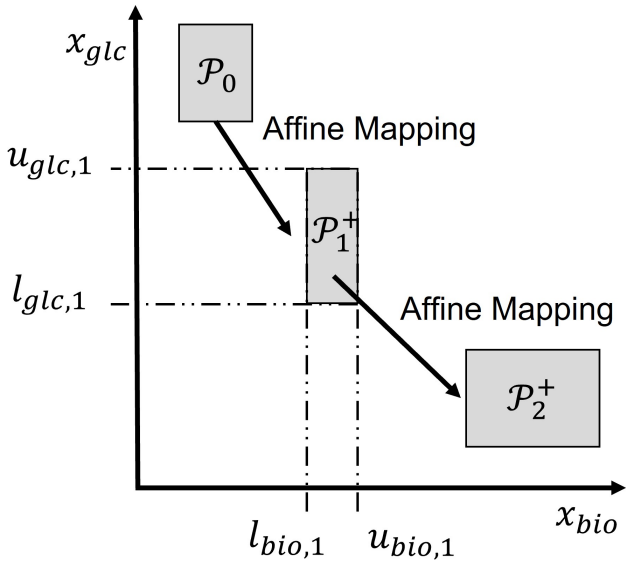


Figure 4.2: Illustration of set propagation of SME by set operations.

4.3.5 Detecting the transition between critical regions

The proposed use of multiparametric programming converts the DFBM into a variable structure system composed of subsystems where each critical region corresponds to a subsystem. Along a given time trajectory the states may transit from one critical region to another. When the states estimated by the EKF leave a critical region $\boldsymbol{\Theta^i}$ to enter into another critical region $\boldsymbol{\Theta^j}$, the estimate $\hat{\boldsymbol{x}}_{\boldsymbol{M,k}}$ and the covariance $\boldsymbol{P_k}$ must be reinitialized because $\boldsymbol{x_M}$ for different critical regions may be different, even though the measured states are the same. Moreover, a criterion is required to detect whether the states are entering into a new critical region.

When the system is traversing from one critical region to another, it needs to cross a boundary between the critical regions. Over time the states may cross over several boundaries along their trajectories and these crossings must be detected. Two neighboring critical regions share a boundary where an active constraint will become inactive or vice versa. The activation of a constraint may require the change of constraints related to $\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}$. For a given constraint, θ is usually the only function of two states at most because of commonly used Michaelis–Menten kinetics [71] or constraints to prevent the depletion of nutrients [13] and one of these two states is biomass. So two special cases should be considered as follows when the system switches from one critical region to the next:

Case i- $\boldsymbol{x_N^i}$ of the old critical region $\boldsymbol{\Theta^i}$ have one more state observable than the $\boldsymbol{x_N^j}$ of the new critical region $\boldsymbol{\Theta^j}$. For this case, the switch between critical regions is determined by Eq. (4.21). Eq. (4.21) calculates the norm of the difference between the flux estimates obtained with Eq. (4.7) in the two neighboring regions. Notice that the flux estimate of $\boldsymbol{\Theta^j}$ is based on estimate $\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}$ of the old critical region. The value of $\gamma(i,j,k)$ is used to detect the occurrence of a switch. If the system is exactly at the boundary of these two critical regions, the flux equation Eq. (4.7) for these two critical regions should result in the same flux value, and $\gamma(i,j,k)$ will be zero. A schematic example is shown in Fig. (4.3). Polygons in different colors represent different critical regions in the parameter space $\boldsymbol{\Theta}$. As the state evolves with time, the corresponding $\boldsymbol{\theta}$ changes along the dashed line in parameter space $\boldsymbol{\Theta}$. As the $\boldsymbol{\theta}$ approaches the boundary of between the critical region $\boldsymbol{\Theta^1}$ and $\boldsymbol{\Theta^2}$, $\gamma(i,j,k)$ approaches zero. Correspondingly, a value of $\gamma(i,j,k)$ smaller than a user-specified tolerance indicates a switch between critical regions thus requiring reinitialization of the EKF as follows: $\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{j}}$ is set equal to $\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}$ and $\boldsymbol{P_k^j}$ is set equal $\boldsymbol{P_k^i}$.

$$\gamma(i,j,k) = \left\| \hat{\boldsymbol{v}}_{\boldsymbol{k}}^{\boldsymbol{i}} - \hat{\boldsymbol{v}}_{\boldsymbol{k}}^{\boldsymbol{j}} \right\| = \left\| \boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}} \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{i}}(\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}) + \boldsymbol{g}^{\boldsymbol{i}} - (\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{j}} \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{j}}(\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}) - \boldsymbol{g}^{\boldsymbol{j}}) \right\| \tag{4.21}$$

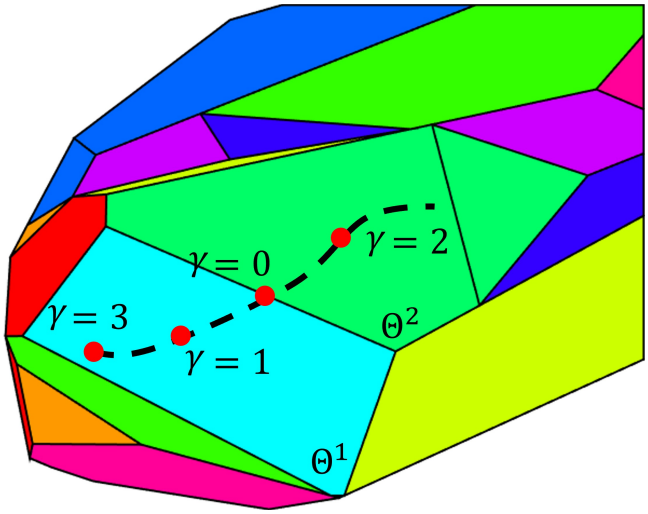


Figure 4.3: Illustration of detecting critical region switch.

Case ii- $\boldsymbol{x_N^j}$ of the new critical region $\boldsymbol{\Theta^i}$ have one more state observable than the $\boldsymbol{x_N^j}$ of the old critical region $\boldsymbol{\Theta^i}$. To reinitialize the EKF, $\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{j}}$ and $\boldsymbol{P_k^j}$ can be set to the old values except for the new observable state that is not observable in the old critical region and thus it needs to be estimated for calculating $\gamma(i,j,k)$. By projecting the set $\mathcal{P}_k^+$, the lower $l_{un,k}$ and upper bounds $u_{un,k}$ can be calculated. Since no extra information is available, the mean value of the upper bound and the lower bound is used as the nominal value of the unobservable state as per Eq. (4.22).

$$\hat{x}_{un,k}^i = \frac{1}{2}(u_{un,k} + l_{un,k}) \tag{4.22}$$

Eq. (4.23) is used to calculate $\gamma(i,j,k)$. The flux estimate for the new critical region $\boldsymbol{\Theta^j}$ is based on the nominal values of the unobservable state $\hat{x}_{un,k}^i$ combined with $\hat{x}_{N,k}^i$ of the old critical region.

$$\gamma(i,j,k) = \left\| \hat{\boldsymbol{v}}_{\boldsymbol{k}}^{\boldsymbol{i}} - \hat{\boldsymbol{v}}_{\boldsymbol{k}}^{\boldsymbol{j}} \right\| = \left\| \boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}} \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{i}}(\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}) + \boldsymbol{g}^{\boldsymbol{i}} - (\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{j}} \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{j}}(\hat{x}_{un,k}^i, \hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}) - \boldsymbol{g}^{\boldsymbol{j}}) \right\| \tag{4.23}$$

To reinitialize the EKF the estimate and covariance are used together with the estimate of the new state that is added in the new critical region. Assuming the states are close

enough to the boundary between the critical regions, then equation (4.24) holds.

$$\left\| \boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{i}} \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{i}}(\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}^{\boldsymbol{i}}) + \boldsymbol{g}^{\boldsymbol{i}} - (\boldsymbol{H}_{\boldsymbol{N}}^{\boldsymbol{j}} \boldsymbol{\theta}_{\boldsymbol{N}}^{\boldsymbol{j}}(\hat{x}_{N,k}^{j}, \hat{x}_{un,0}^{i}) - \boldsymbol{g}^{\boldsymbol{j}}) \right\| = 0 \tag{4.24}$$

The initial estimate of the new observable state $\hat{x}_{un,0}^{j}$ in the new region can be calculated by solving the equation (4.24). Since the new state is between the upper bound and lower bound by SME, the half-length between $u_{un,k}$ and $l_{un,k}$ is the worst possible deviation. Then, using a 3 standard deviation range, the initial variance $\eta_{un,k}^2$ can be estimated according to Eq. (4.25) and all other covariance terms related to the new state are assumed to be zero.

$$\eta_{un,k} = \frac{1}{3} \cdot \frac{1}{2}(u_{un,k} - l_{un,k}) \tag{4.25}$$

Bounds of states estimated by the SME are rigorously guaranteed in each critical region separately but subject to the accurate tuning of the tolerance that is used to switch between the subsystems. The tolerance of $\gamma(i, j, k)$ is the only user-specified parameter in this research. If the tolerance is too large or small, the EKF may switch the subsystem too early or too late. Accordingly, if the wrong state equations are used in estimation, the bounds on the states may be violated. To avoid such a situation, exhaustive simulations that are initialized with $\mathcal{P}_0$ are conducted to find the tolerance used to switch between critical regions. As an alternative, an overestimated covariance can also be used to reinitialize the EKF when a state enters a new critical region to avoid bound violations.

4.4 Results

4.4.1 DFBM Model of *E.coli*

A DFBM model of *E. coli* reported in [64] is used to illustrate the proposed methodology. The DFBM in batch operation includes four states glucose concentration x_{glc}, oxygen concentration x_{oxy}, acetate concentration x_{ace}, and biomass concentration x_{bio} as Eq. (4.26). Thus the state vector is $\boldsymbol{x} = \begin{bmatrix} x_{glc} & x_{oxy} & x_{ace} & x_{bio} \end{bmatrix}^T$. The substrates are glucose, oxygen,

and acetate.

$$x_{glc,k+1} = x_{glc,k} + \Delta t x_{bio,k} \boldsymbol{A_{glc}} \boldsymbol{v_k} \tag{4.26a}$$
$$x_{oxy,k+1} = (1 - k_L a \Delta t) x_{oxy,k} + \Delta t x_{bio,k} \boldsymbol{A_{oxy}} \boldsymbol{v_k} + 0.21 k_L a \Delta t \tag{4.26b}$$
$$x_{ace,k+1} = x_{ace,k} + \Delta t x_{bio,k} \boldsymbol{A_{ace}} \boldsymbol{v_k} \tag{4.26c}$$
$$x_{bio,k+1} = x_{bio,k} + \Delta t x_{bio,k} \boldsymbol{A_{bio}} \boldsymbol{v_k} \tag{4.26d}$$
$$\boldsymbol{x_0} \in \mathcal{P}_0 = \mathcal{S}(\begin{bmatrix} 0.38 & 0.1995 & 0.19 & 0.00095 \end{bmatrix}^T, \begin{bmatrix} 0.42 & 0.2205 & 0.21 & 0.00105 \end{bmatrix}^T) \tag{4.26e}$$

Where $k_L a = 4\ h^{-1}$ is the oxygen mass transfer coefficient. The initial state vector $\boldsymbol{x_0}$ is defined by the interval set $\mathcal{P}_0$ according to Eq. (4.26d). The matrix A contains the stoichiometric coefficients corresponding to four reactions according to Eq. (4.27). Each column of this matrix corresponds to one reaction and each row corresponds to one component.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A_{glc}} \\ \boldsymbol{A_{oxy}} \\ \boldsymbol{A_{ace}} \\ \boldsymbol{A_{bio}} \end{bmatrix} = \begin{bmatrix} 0 & -9.46 & -9.84 & -19.23 \\ -35 & -12.92 & -12.73 & 0 \\ -39.43 & 0 & 1.24 & 12.12 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.27}$$

The flux vector $\boldsymbol{v_k}$ is obtained by solving the following linear programming problem as Eq. (4.28):

$$\max_{\boldsymbol{v_k}} \quad \boldsymbol{A_{bio}} \boldsymbol{v_k} \tag{4.28a}$$
$$\text{subject to} \quad -\boldsymbol{A_{oxy}} \boldsymbol{v_k} \leq \text{OUR}_{max} \tag{4.28b}$$
$$\boldsymbol{A_{ace}} \boldsymbol{v_k} \leq 100 \tag{4.28c}$$
$$-\Delta t \boldsymbol{A_{glc}} \boldsymbol{v_k} \leq \frac{x_{glc,k}}{x_{bio,k}} = \theta_{1,k} \tag{4.28d}$$
$$-\Delta t \boldsymbol{A_{oxy}} \boldsymbol{v_k} \leq \frac{(1 - k_L a \Delta t) x_{oxy,k} + 0.21 k_L a \Delta t}{x_{bio,k}} = \theta_{2,k} \tag{4.28e}$$
$$-\Delta t \boldsymbol{A_{ace}} \boldsymbol{v_k} \leq \frac{x_{ace,k}}{x_{bio,k}} = \theta_{3,k} \tag{4.28f}$$
$$-\boldsymbol{A_{glc}} \boldsymbol{v_k} \leq \frac{\text{GUR}_{max} x_{glc,k}}{K_m + x_{glc,k}} = \theta_{4,k} \tag{4.28g}$$

Where $\text{OUR}_{max} = 12\text{mM/(g-dw·h)}$ is the maximum oxygen uptake rate and g-dw is grams of the dry weight of biomass; $\text{GUR}_{max} = 6.5\text{mM/(g-dw·h)}$ denotes the maximum glucose uptake rate. Eq. (4.28a) describes that the objective of the cells is to maximize the biomass growth rate. Eq. (4.28b) indicates that the oxygen consumption rate is limited by a maximum uptake limit. Eq. (4.28c) indicates that the acetate generation rate is bounded by 100mM/(g-dw·h). Eq. (4.28g) indicates that the glucose consumption rate is bounded by an upper limit. All the other constraints are positivity constraints to prevent the depletion of metabolites. To express these constraints in Eq. (4.28) compactly the constraints in (4.28) can be expressed in the form of Eq. (4.3):

$$\boldsymbol{G}\boldsymbol{v}_{\boldsymbol{k}} \leq \boldsymbol{F}\boldsymbol{\theta}_{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{k}}) + \boldsymbol{z} \tag{4.29a}$$

$$\boldsymbol{G} = \begin{bmatrix} -\boldsymbol{A}_{\boldsymbol{oxy}} \\ \boldsymbol{A}_{\boldsymbol{ace}} \\ -\Delta t\boldsymbol{A}_{\boldsymbol{glc}} \\ -\Delta t\boldsymbol{A}_{\boldsymbol{oxy}} \\ -\Delta t\boldsymbol{A}_{\boldsymbol{ace}} \\ -\boldsymbol{A}_{\boldsymbol{glc}} \end{bmatrix} \tag{4.29b}$$

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.29c}$$

$$\boldsymbol{z} = \begin{bmatrix} \text{OUR}_{max} \\ 100 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.29d}$$

4.4.2 Determination of Minimum Measurements

Due to the assumption that the initial state is contained in an interval, the problem in Eq. (4.28) can be formulated as a multiparametric linear programming (mpLP) problem. The vector $\boldsymbol{\theta}$ is composed of four parameters which are nonlinear functions of states. Using

the Multi-Parametric Toolbox 3.0 it is found that the entire parameter space $\boldsymbol{\Theta}$ can be decomposed into a maximum of 24 critical regions. For each critical region, the mpLP solver calculates the constraints that form the boundaries of the region and the equations that generate the optimal solutions. In order to reduce the computational effort, extensive simulations are conducted with randomly chosen initial values in set $\mathcal{P}_0$ to identify which critical regions are relevant for the problem. It is found from these simulations that for the chosen range of initial conditions the states only traverse through two neighboring critical regions $\boldsymbol{\Theta}^1$ and $\boldsymbol{\Theta}^2$ assuming small critical regions are ignored. According to the results of the mpLP solver, the two critical regions can be defined as Eq. (4.30a) and Eq. (4.30b). Critical regions $\boldsymbol{\Theta}^1$ and $\boldsymbol{\Theta}^2$ share a boundary defined in Eq. (4.30c). Since $\boldsymbol{\theta}$ is a function of $\boldsymbol{x}$, the critical regions are next to each other in the state space.

$$\boldsymbol{\Theta}^1 : \begin{bmatrix} -0.9988 & 0 & 0 & 0.0499 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -0.9971 & -0.0767 & 0 \\ 0 & 0 & 0 & -0.0033 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix} \boldsymbol{\theta}(\boldsymbol{x}) \leq \begin{bmatrix} 0 \\ -0.6 \\ -0.6740 \\ 0.0171 \\ 8.7864 \\ 0 \end{bmatrix} \tag{4.30a}$$

$$\boldsymbol{\Theta}^2 : \begin{bmatrix} -0.9988 & 0 & 0 & 0.0499 & 0 \\ 0 & -0.7469 & 0.6630 & 0.0510 & 0 \\ 0 & 0 & -0.0254 & -0.0053 & -0.9997 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0.9971 & 0.0767 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix} \boldsymbol{\theta}(\boldsymbol{x}) \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.6740 \\ 0 \end{bmatrix} \tag{4.30b}$$

$$\boldsymbol{\Theta}^1 \bigcap \boldsymbol{\Theta}^2 : \begin{bmatrix} 0 & 0 & 0.9971 & 0.0767 & 0 \end{bmatrix} \boldsymbol{\theta}(\boldsymbol{x}) = 0.6740 \tag{4.30c}$$

Accordingly, the mpLP solver also calculates the matrix $\boldsymbol{H}$ and $\boldsymbol{g}$ used in the flux equation Eq. (4.7) for these two critical regions. By taking advantage of the sparseness of $\boldsymbol{H}$ for these two critical regions, $\boldsymbol{\theta_N}$ can be determined. The equations to calculate fluxes for these two critical regions can be expressed as Eq. (4.31).

$$\boldsymbol{v_k^1} = \begin{bmatrix} -0.039 & 0.1057 & 0 & 0 \end{bmatrix}^T \theta_4(x_{glc,k}) + \begin{bmatrix} 0.3429 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.31a}$$

$$\boldsymbol{v_k^2} = \begin{bmatrix} 0.5072 & 0 & 0 & 0 \end{bmatrix}^T \theta_3(x_{ace,k}, x_{bio,k}) + \begin{bmatrix} 0 & 0.1057 & 0 & 0 \end{bmatrix}^T \theta_4(x_{glc,k}) \tag{4.31b}$$

Where $\boldsymbol{\theta_N}$ for critical region $\boldsymbol{\Theta}^1$ is θ_4 and $\boldsymbol{\theta_N}$ for critical region $\boldsymbol{\Theta}^2$ is θ_3 and θ_4. By substituting the flux equation Eq. (4.31) into the Eq. (4.26), the simplified state equations of *E.coli* model can be rewritten compactly as in Eq. (4.32).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B x_k} + \Delta t x_{bio,k} \boldsymbol{A v_k^1}(x_{glc,k}) + \boldsymbol{h} \qquad \boldsymbol{\theta(x_k)} \in \boldsymbol{\Theta}^1 \qquad (4.32a)$$

$$\boldsymbol{x_{k+1}} = \boldsymbol{B x_k} + \Delta t x_{bio,k} \boldsymbol{A v_k^2}(x_{ace,k}, x_{bio,k}, x_{glc,k}) + \boldsymbol{h} \qquad \boldsymbol{\theta(x_k)} \in \boldsymbol{\Theta}^2 \qquad (4.32b)$$

Following the calculations above the original *E. coli* model is simplified into an equivalent system comprised of two subsystems of interest. Eq.(4.32a) and Eq.(4.32b) describe subsystem 1 and subsystem 2 respectively. These two subsystems are continuous in the state space and they share the same boundary as per Eq. (4.30c). Once the state crosses the boundary between the two subsystems, the governing equation is switched from Eq. (4.32a) to Eq. (4.32b). Because the initial state is randomly initialized in set $\mathcal{P}_0$, $\mathcal{P}_0$ corresponds to a set in $\boldsymbol{\Theta}^1$. Thus, the state evolves within the region of subsystem 1 and gradually approximates the region of subsystem 2 governed by Eq. (4.32b) until finally crosses the boundary given by Eq. (4.30c). As only part of θ is known, a detector is used to detect the crossing of the boundary thus ensuring that the switch between the regions is done accurately.

Based on the flux equation Eq. (4.31), the reaction-rate-determining states vector $\boldsymbol{x_M^i}$ for $\boldsymbol{\Theta}^1$ are biomass and glucose and for $\boldsymbol{\Theta}^2$ are biomass, acetate, and glucose. Accordingly, the possible combinations of measurements needed for observing $\boldsymbol{x_M^1}$ of $\boldsymbol{\Theta}^1$ include $\Omega_1^1 = \{Bio\}$, $\Omega_2^1 = \{Glc\}$ and $\Omega_3^1 = \{Bio, Glc\}$. Similarly, there are 7 possible combinations of measurements for observing the vector $\boldsymbol{x_M^2}$ in $\boldsymbol{\Theta}^2$, namely $\Omega_1^2 = \{Ace\}$, $\Omega_2^2 = \{Bio\}$, $\Omega_3^2 = \{Glc\}$, $\Omega_4^2 = \{Ace, Bio\}$, $\Omega_5^2 = \{Bio, Glc\}$, $\Omega_6^2 = \{Ace, Glc\}$, and $\Omega_7^2 = \{Ace, Bio, Glc\}$. To find a combination of measurements Ω_Γ that will be suitable for both critical regions, it is necessary to perform an analysis of observability for these combinations. The Symbolic Toolbox calculation of MATLAB R2018a is used to develop an analytical equation observability rank condition and rank of $\boldsymbol{\Phi_k^i}$ of the nonlinear system according to the criterion presented in [103]. Since the symbolic expressions of the rank for each critical region for Eq. (4.11) are very complex it is very difficult to infer an analytical condition of observability for all possible values of the states. Instead, the rank values are calculated for different measurement combinations and rank of $\boldsymbol{\Phi_k^i}$ using a Monte Carlo algorithm based on 5 million samples of $\boldsymbol{\Theta}^1$ and $\boldsymbol{\Theta}^2$ respectively. According to these Monte Carlo simulations, the only measurement required for observability of the vectors $\boldsymbol{x_M^1}$ in $\boldsymbol{\Theta}^1$ and $\boldsymbol{x_M^2}$ in $\boldsymbol{\Theta}^2$ is the biomass concentration, namely $\Omega_\Gamma = \{Bio\}$.

4.4.3 EKF for the Two Subsystems and Detection of Transition between Subsystems

Based on the aforementioned observability analysis the biomass concentration is the only state that needs to be measured online as per Eq. (4.33a) for implementation of the EKF. Measurement noise is assumed as a truncated normal distribution as described by Eq. (4.33b). Since the initial $\mathcal{P}_0$ is assumed to be known, the EKF is initialized at the center of $\mathcal{P}_0$ with a variance based on 3 standard deviations and zero covariance terms. The state of the plant is initialized randomly by sampling a point within the region defined by $\mathcal{P}_0$.

$$y_k = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x_k} + r_k \tag{4.33a}$$

$$r_k \sim TN(0, 0.004^2, -0.0004, 0.0004) \quad k = 0, 1, 2 \cdots \tag{4.33b}$$

Based on the assumed $\mathcal{P}_0$, in the batch process the EKF starts in the critical region $\boldsymbol{\Theta}^1$ and later it transitions into critical region $\boldsymbol{\Theta}^2$. Thus, two EKFs are required in this case study to estimate the $\boldsymbol{x_M}$ as summarized in Table. 4.1. Based on the biomass measurement y_k, the glucose and biomass concentrations are estimated by the EKF for $\boldsymbol{\Theta}^1$ as $\hat{x}_{N,bio,k}$ and $\hat{x}_{N,glc,k}$. With the same biomass measurement, the second critical region $\boldsymbol{\Theta}^2$ has one more observable state which is the acetate concentration $\hat{x}_{N,ace,k}$.

Table 4.1: Observable and Unobservable Subspace of Two Subsystems of DFBM Model of *E.coli* .

	Subsystem of $\boldsymbol{\Theta}^1$	**Subsystem of $\boldsymbol{\Theta}^2$**
Observable Subspace ($\boldsymbol{x_M}$)	Glc, Bio	Glc, Ace, Bio
Unobservable Subspace	Ace, Oxy	Oxy
Measurement	Bio	Bio

Since acetate and oxygen are unobservable in $\boldsymbol{\Theta}^1$ they need to be estimated by bounds. To find these bounds, SME propagates the initial set $\mathcal{P}_0$ by set operations to obtain a prior estimate set $\mathcal{P}_k^-$ as Eq. (4.19). After obtaining the measurement of biomass, a posterior estimate set $\mathcal{P}_k^+$ as in Eq. (4.20) is calculated by set operations. The error due to the lack of convergence of the EKF is compensated by using Eq. (4.18). By projecting $\mathcal{P}_k^+$ onto the axis of acetate and oxygen respectively, the upper bound $\boldsymbol{u_{un,k}}$ and lower bound $\boldsymbol{l_{un,k}}$ of these two states are obtained.

Since $\boldsymbol{\Theta}^2$ has one more flux-determining state acetate that is not observable from the measurement of biomass, it must be estimated as explained in Eq. (4.22). Using the mean value of $u_{un,ace,k}$ and $l_{un,ace,k}$ the nominal values of the unobservable state $\hat{x}_{un,ace,k}$ is obtained. Using the EKF estimates of the observable flux-determining states $\hat{\boldsymbol{x}}_{\boldsymbol{N,k}}$ together with the nominal value of acetate $\hat{x}_{un,ace,k}$, the detection scheme explained in subsection (4.3.5) can be implemented. Accordingly, $\gamma(i,j,k)$ is calculated from Eq. (4.23) to determine the switch from critical region $\boldsymbol{\Theta}^1$ to critical region $\boldsymbol{\Theta}^2$. The tolerance of $\gamma(i,j,k)$ to determine the switch between the critical regions is assumed as 0.08. This tolerance is the only tuning parameter of the proposed method and it is determined by trial and error. After the switch occurs the acetate concentration is initialized by the solution of Eq. (4.24) and the variance of acetate is initialized based on Eq. (4.25). After the switch to critical region $\boldsymbol{\Theta}^2$ the EKF continues to generate estimates of glucose, acetate, and biomass concentrations in $\boldsymbol{\Theta}^2$, and the SME approach is used to propagate the set $\mathcal{P}_k^+$ as done in critical region 1. Fig. (4.4) presents the posterior estimate sets $\mathcal{P}^+$ and true plant state $\boldsymbol{x}$ at different times. Since the model is 4 dimensional, the posterior estimate sets $\mathcal{P}^+$ are projected for visualization onto two-dimensional spaces: glucose-oxygen subspace and acetate-biomass subspace. The 8 boxes denote the projected posterior estimate sets between $0h$ to $7h$ and each box represents an hour. The arrows in Fig. (4.4) indicate the direction of time evolution. The black dots denote the true plant state. Since biomass is measured, the length of the boxes along the biomass dimension is relatively smaller as compared to the other dimensions. The switch between the critical regions occurs at around $5h$.

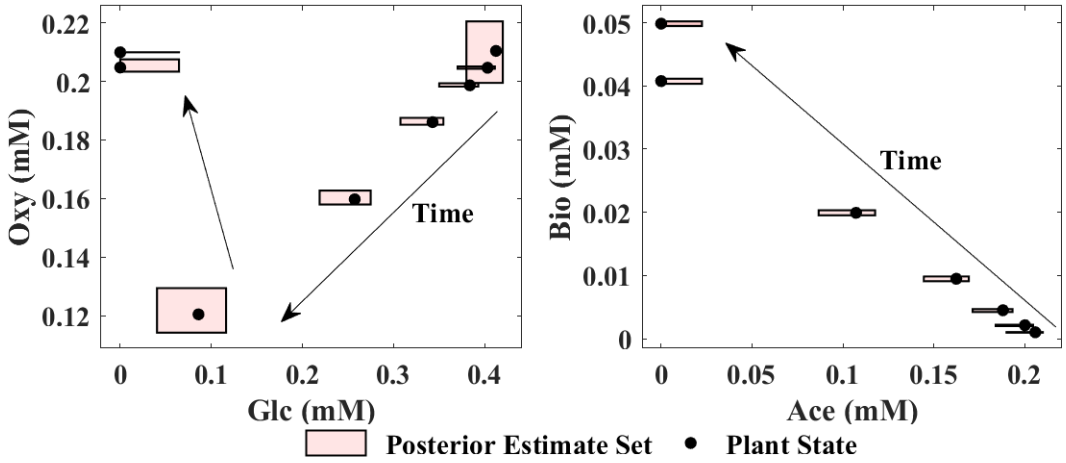


Figure 4.4: Posterior estimate sets projected onto glucose-oxygen subspace and acetate-biomass subspace at different times.

4.4.4 Set Membership Estimation

To verify the estimate and bounds generated by the proposed algorithm, we use a special Monte Carlo Algorithm (MCA) that takes biomass measurements into account. MCA randomly samples 100000 different points from $\mathcal{P}_0$ and uses them as initial states' values, and then calculates the corresponding trajectories with respect to time. Since for the measurement of biomass a truncated normal distribution measurement noise was assumed, some trajectories are not within the confidence interval of measurements. Once a trajectory is found out of the measurement range, the evolution of the trajectory is stopped and the corresponding trajectory is removed while trajectories that are still within the confidence interval of measurements are kept. Accordingly, only a part (2581) out of the trajectories starting from $\mathcal{P}_0$ are used for comparison to the bounds calculated by the proposed method. It should be noticed that the fraction of trajectories kept for comparison is small because only a very narrow set of solutions are within the measurement range from the beginning to the end. In other words, only a small part of the samples considered in the simulation is compatible with the biomass-measured trajectory that is assumed for the calculation of bounds by the set-based approach. Using parallel computation, 4 hours and 4 minutes of CPU time were required to complete all simulations. For comparison, the method proposed in this work can generate bounds with only 41 sec of CPU time without parallel computation. It should be remembered that the MCA was conducted for a specific trajectory of biomass measurements so as to enable a fair comparison with the method proposed in the

current study. While it could be argued that MCA could be used to calculate bounds for all possible biomass trajectories this will be computationally prohibitive. Thus the proposed technique is a practical and analytical approach to the online estimation problem.

In Fig. (4.5) the grey area denotes the trajectories randomly sampled and the two black lines represent the upper and lower bounds by SME. It is clear that the SME contains all the solutions generated by MCA, especially for the unobservable states. It can be observed that the switch from one critical region to the other occurs at approximately 5 h as shown in Fig. (4.2) shows. Before 5 h, the reactor has enough resources for cell growth and the limiting step is glucose uptake as Eq. (4.31a) shows. Thus, critical region $\mathbf{\Theta}^1$, corresponds to the logarithmic phase of growth where the latter is driven by glucose consumption. At about 5 h, the simultaneous depletion of acetate and glucose leads to a metabolic switch from the logarithmic phase to the stationary phase. Following this metabolic switch, the culture is also acetate limited, and thus acetate becomes a new flux-determining state. Since the oxygen feed rate is maintained constant in the model, the fact that the growth significantly decreases after the switch explains why the oxygen concentration bounces back up.

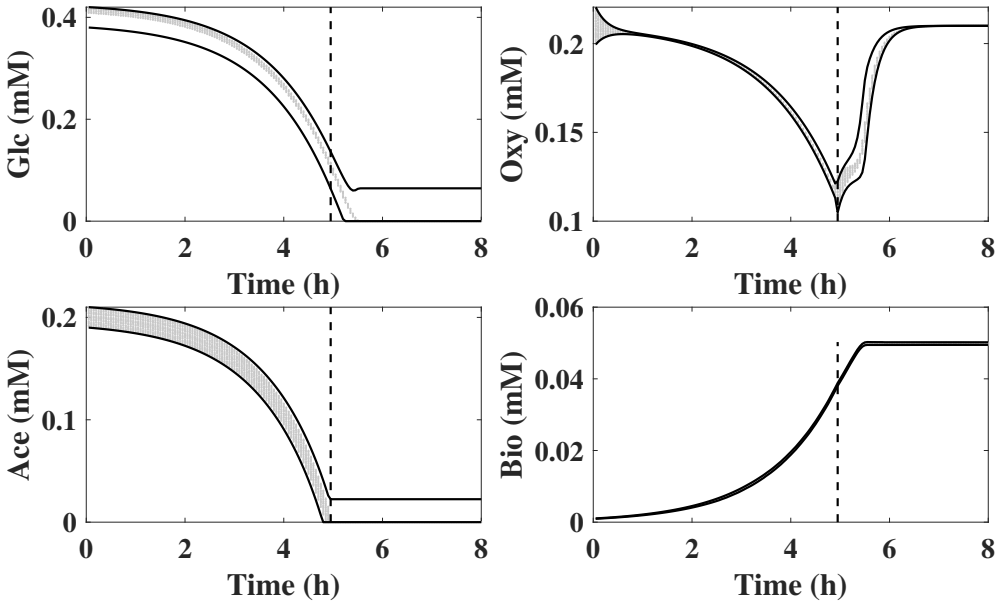


Figure 4.5: Comparison between MCA with bounds of 4 components estimated by SME in batch fermentation of *E.coli*.

To further verify the proposed scheme, similar MCA simulations were conducted with a larger initial uncertainty and measurement noise. In Fig. (4.6), the bounds of 4 component concentrations estimated by SME are shown. It is clear that the simulated trajectories contained in the grey color band generated by MCA are within the bounds calculated by the proposed methodology. From the comparison of Fig. (4.5) and Fig. (4.6), it is found that the SME approach copes with the larger noise and initial uncertainty by generating larger bounds.

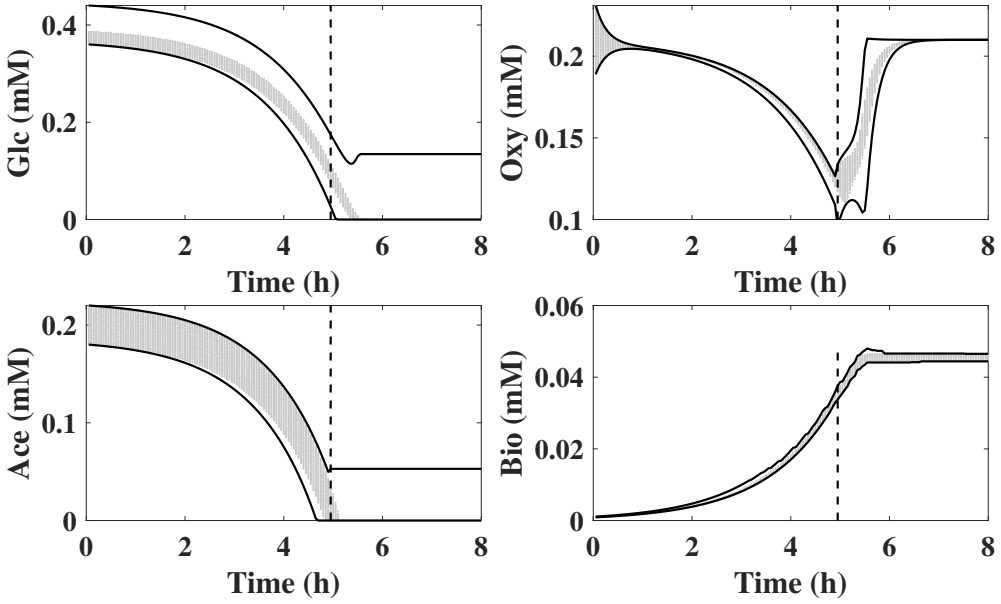


Figure 4.6: Comparison between MCA with bounds of 4 components estimated by SME with a loud noise.

4.5 Discussion

DFBM models are advantageous since they contain significant detail about cell metabolism as compared to classical unstructured models. However, due to this level of detail, DFBM contains many states thus resulting in a more difficult state estimation problem. The challenge of dealing with a large number of states is further exacerbated by the fact that online measurements of metabolites are generally difficult to obtain or not available. With limited online measurements, it is often impossible to have observability for all the states.

Noticing that the diagonal matrix $\boldsymbol{B}$ in Eq. (4.19) is a linear mapping of states, if the nonlinear term $\Delta t x_{bio,k} \boldsymbol{A v_k}$ can be estimated then it is possible to estimate the other states of the DFBM.

Multiparametric LP is introduced to convert the original system into a series of piecewise continuous subsystems based on the partitioning of the parameter space into critical regions. The availability of an explicit expression for the calculation of the LP optima for each critical region significantly simplifies the solution of the problem. Although many critical regions may be mathematically possible, industrial fermentation is operated in a narrow range of initial operating conditions such as only a few critical regions need to be considered.

Beyond their computational convenience, the critical regions identified by the Multiparametric LP approach can be interpreted as corresponding changes in cell metabolism. The relative abundance of substrates i.e. glucose, acetate, and oxygen in *E.coli* model and their consumption of biomass lead to the occurrence of different resources limitations at any given time. Within some range of concentration, the limiting substrate remains the same corresponding to a specific metabolism strategy.

In the *E. coli* example, four reactions can synthesize the biomass from glucose, acetate, and oxygen. However, since the objective is to maximize growth subject to constraints, the cell prioritizes these reactions differently at any given time due to their different efficiency for biomass synthesis. The ratio of the stoichiometry coefficients in each column of matrix $\boldsymbol{A}$ indicates the biomass yield of each substrate for each reaction. Reaction 1 is the only reaction that consumes acetate to synthesize biomass. The yield of acetate to biomass is $\frac{1}{39.43}$ for reaction 1, which is very low compared with reaction 2 and reaction 3. The biomass yield of reaction 2 and reaction 3 by glucose is $\frac{1}{9.46}$ and $\frac{1}{9.84}$ respectively. Reaction 4 is the only reaction that does not consume oxygen to generate biomass but it is very inefficient. Because the biomass yield of these reactions is different, reaction 2 is preferred over reaction 1 and reaction 3 when glucose and oxygen are abundant. When oxygen is very low, the cells switch their metabolism from aerobic to anaerobic to generate biomass through reaction 4.

To maximize the biomass growth rate, cells take advantage of reactions 1 and 2 to consume as much acetate and glucose as possible when oxygen is sufficient. However, the glucose amounts that can be consumed by the cells are limited by the glucose uptake rate, which is θ_4. Similarly, oxygen consumption is limited by a constant oxygen uptake rate as Eq. (4.28b). The oxygen is consumed first with glucose in reaction 2 to synthesize biomass and the remaining oxygen is consumed for reaction 1. Multiparametric LP captures the relative priority of different reactions towards maximization of growth and identifies the

key limited resources. In critical region $\mathbf{\Theta^1}$, glucose is the key resource that determines the flux vector according to Eq. (4.31a). As glucose and acetate are consumed by reactions 1 and 2, biomass increases exponentially and the oxygen concentration drops fast due to oxygen demands as in Figs. (4.5) and (4.6). At some point, the concentration of acetate becomes very low but acetate is necessary for reaction 2 to synthesize biomass. Acetate becomes the key limited resource and the system enters into a new critical region $\mathbf{\Theta^2}$. Then in $\mathbf{\Theta^2}$, the metabolism is limited by the available acetate and glucose, and as they deplete the growth of cells decreases and ultimately stops. Accordingly, $\mathbf{\Theta^1}$ corresponds to the logarithmic phase and $\mathbf{\Theta^2}$ to the stationary phase of growth.

The use of EKF for each subsystem is used to estimate the reaction-rate-determining states thus reducing the need for online measurements. Since biomass is highly correlated with the reaction-rate-determining states, EKF can take advantage of biomass measurement to estimate these states. Because some of these reaction-rate-determining states are common to different critical regions, only a fewer states are required to be measured or estimated, which greatly reduces the demand for online measurements of concentration. In the *E.coli* example, only biomass need to be measured. Once the biomass is measured, glucose can be estimated by the EKF in the critical region $\mathbf{\Theta^1}$, and glucose and acetate can be estimated in $\mathbf{\Theta^2}$.

By using the SME upper and lower bounds for all states can be generated including the unobservable ones such as acetate and oxygen in $\mathbf{\Theta^1}$. Using the bounds of acetate and biomass estimate it was possible to determine the switch from one critical region to another and to re-initialize the estimates and covariance matrix for the EKF after the switch.

This research is helpful in DFBM-based control in bio-processes when many components cannot be measured online. Using the upper and lower bounds calculated by SME of unobservable states and estimated by EKF of observable states, robust control methods can be applied to achieve optimal operation in the presence of uncertainty. The method developed can also be extended to monitor the bio-processes and differentiate normal and abnormal operations.

4.6 Conclusions

This research proposed a comprehensive DFBM-based approach to estimate the metabolite concentrations with a minimal number of online measurements. The main idea is to convert the DFBM model with uncertainty in initial conditions to an explicit variable structure system that can be analyzed by multiparametric linear programming. A key finding of the

proposed work is that only a subset of the states referred to as reaction-rate-determining states, is needed to calculate the flux vector. Identification of the reaction-rate-determining states for each critical region permitted the determination of the minimum set of measurements required for full-state estimation. EKFs were used to estimate the observable states and set propagation by SME was used to identify the bounds of both the observable states and unobservable states.

Chapter 5

Online Estimation Using Dynamic Flux Balance Model and Multiparametric Programming

5.1 Overview

[1]An approach is proposed for online estimation of bounds on metabolite's concentrations based on limited measurements and Dynamic Flux Balance Models (DFBM) which use linear programming (LP) to model the evolution of metabolites with time. A Weighted primal-dual method to address the multiplicity of solutions of DFBMs is combined with multiparametric nonlinear programming (mpNLP) for set membership estimation. The set membership estimation (SME) approach is used to propagate the uncertainty onto metabolites' concentrations over time. By only measuring biomass concentration and culture volume, the bounds of metabolites' concentrations can be estimated online by SME during the fermentation. The proposed algorithm is applied to batch and fed-batch fermentation of *E. coli*.

[1]Adapted from Shen, X., & Budman, H. (2022). Online estimation using dynamic flux balance model and multiparametric programming. Computers & Chemical Engineering, 164, 107872.

5.2 Introduction

Nowadays the production of bio-pharmaceuticals highly relies on process monitoring and control. While model-based control is widespread in many process industries, its application is more challenging in the pharmaceutical industry because of the lack of online measurements [4, 52]. Online measurements of crucial variables in biochemical processes, including concentrations of amino acids, vitamins, metals, and precursors, are often difficult to obtain.

Dynamic flux balance models (DFBMs) are genome-based models that have gained increasing attention for modeling biochemical processes. A DFBM is formulated as a linear programming (LP) problem based on the assumption that cells allocate available nutrients to boost a biological objective such as growth or other. A key challenge for the solution of the resulting LP is that the optimal solution is often not unique. Three main methods have been proposed to solve the multiplicity of solutions, including the lowest overall flux parsimonious enzyme usage FBA (pFBA) [60, 47, 78], hierarchical optimization (lexicographic optimization) [36, 2] and weighted primal-dual method (WPDM) [98]. WPDM converts the LP into interior-point form nonlinear programming (NLP) and uses parameters interior weights to determine which unique solution is obtained. Parameters interior weights can be obtained by fitting experimental data which makes the method data-driven and widely applicable. Because of the ability to adjust weights based on data, WPDM was also shown to provide good fitting as compared to other approaches.

In this research, we combine WPDM to address multiplicity with the set membership estimation (SME) approach to perform online estimation of the bounds of metabolites' concentrations. Since WPDM converts the LP within the DFBMs into a nonlinear programming problem, it is not possible to combine directly the SME approach with the WPDM-based solution. Instead, multiparametric nonlinear programming (mpNLP) is applied to the resulting NLP which transforms the original DFBM into a variable structure system. Then, set membership estimation is used to propagate the initial uncertainty through the variable structure system to obtain the bounds of the states over time. The proposed method has been applied to *E. coli* fermentation of batch and fed-batch process. Section 2 presents the methods. Section 3 presents the experimental results and discussions.

5.3 Methods

5.3.1 Dynamic Flux Balance Models

The DFBM is generally defined by a combination of a state space model with a static optimization problem as follows. The state space model as a function of metabolic fluxes is defined in Eq. (5.1).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B}(x_{v,k}, q_k)\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{S}\boldsymbol{v_k} + \boldsymbol{h}(x_{v,k}, q_k) \tag{5.1a}$$

$$\boldsymbol{y_k} = \boldsymbol{C}\boldsymbol{x_k} + \boldsymbol{r_k} \tag{5.1b}$$

$$\boldsymbol{r_k} \sim TN(\boldsymbol{0}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}) \quad k = 0, 1, 2\cdots \tag{5.1c}$$

$$\boldsymbol{x_0} \in \mathcal{X}_0 \tag{5.1d}$$

where subscript k indicates time step from 0, 1, $2\cdots$ and Δt is the time step size. $\boldsymbol{x_k}$ is a vector of n_x state variables at time step k, containing metabolites' concentrations, biomass concentration $x_{bio,k}$ and culture volume $x_{v,k}$. $\boldsymbol{S} \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_r}$ is a matrix containing stoichiometric coefficients of all reactions involved in the metabolic network, where n_r is the number of reactions considered in the metabolic network. q_k is the feed flow rate (for a batch process, $q_k = 0$). $\boldsymbol{v_k} \in \mathbb{R}^{n_r}$ denotes a metabolic flux vector defined below. The matrix $\boldsymbol{B} \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ and vector $\boldsymbol{h} \in \mathbb{R}^{n_x}$ are functions of feed flow rate and culture volume. $\boldsymbol{y_k}$ is the vector of n_y measured variables and C is the output state matrix. The initial state $\boldsymbol{x_0}$ is assumed to be uncertain but within a known set $\mathcal{X}_0$. While in industrial practice the initial media formulation and seed are kept the same, human factors and raw material variability result in batch-to-batch fluctuations in the values of the initial concentrations.

$\boldsymbol{r_k} \in \mathbb{R}^{n_y}$ denote a measurement noise vector which follows a truncated multivariate normal distribution (TN) [112, 11]. The probability density function p for $TN(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u})$ are defined according to Eq. (5.2).

$$p(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{l}, \boldsymbol{u}) = \frac{exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\}}{\int_{\boldsymbol{l}}^{\boldsymbol{u}} exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\}} \tag{5.2}$$

For $\boldsymbol{r_k}$, the mean vector of TN is $\boldsymbol{0} \in \mathbb{R}^{n_y}$; the covariance is $\boldsymbol{\Sigma} \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_y}$; the lower and upper bounds are $\boldsymbol{l} \in \mathbb{R}^{n_y}$ and $\boldsymbol{u} \in \mathbb{R}^{n_y}$ respectively.

The flux vector $\boldsymbol{v_k}$ is determined by a local linear programming (LP) problem according to Eq. (5.3). At each time step, $\boldsymbol{v_k}$ is solved by the LP solver and substituted into Eqs.

(5.1) to obtain the state vector at the next time step.

$$\min_{\boldsymbol{v_k}} \quad \boldsymbol{c}^T \boldsymbol{v_k} \tag{5.3a}$$

$$P \quad \text{subject to} \quad \boldsymbol{G}\boldsymbol{v_k} \leq \boldsymbol{F}\boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) + \boldsymbol{b} \tag{5.3b}$$

where the constant vector $\boldsymbol{c} \in \mathbb{R}^{n_r}$, the constant matrix $\boldsymbol{F} \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_\theta}$, the constant vector $\boldsymbol{b} \in \mathbb{R}^{n_G}$, the constant matrix $\boldsymbol{G} \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_r}$, $\boldsymbol{\theta}(\boldsymbol{x}, q_k) \in \boldsymbol{\Theta} \subseteq \mathbb{R}^{n_\theta}$. For simplicity, the subscript k will be omitted later because the same mathematical expressions apply for all time steps. n_G is the number of linear constraints. The parameter vector $\boldsymbol{\theta}$ is a nonlinear vector-valued function of state $\boldsymbol{x}$ and feed flow rate q_k. n_θ denotes the number of elements in the parameter vector $\boldsymbol{\theta}$. $\boldsymbol{\Theta}$ denotes the parameter space where the optimal solution of the LP resides. Eq. (5.3a) denotes the objective function of the LP and maximization of biomass growth rate, or equivalently minimization of its negative value, is used in the current study. Eq. (5.3b) describes metabolic constraints such as reaction rates' limits and available nutrients' limits. Here Eq. (5.3) is denoted as the P formulation of the LP. A detailed *E. coli* example is presented in the results section.

5.3.2 Weighted Primal-Dual Method

Because of the insufficient number of constraints or possible correlations between constraints and the objective function, the optimal solution of Eq. (5.3) is often not unique. Commonly used LP solvers are dual-simplex algorithms and naive interior-point algorithms. These solvers can only obtain a particular solution from all possible solutions at each time step and there is often no evidence that the solution chosen by the solvers is preferable over other solutions. When these different solvers are used, only specific state trajectories over time are obtained even though multiple trajectories can satisfy the DFBMs. To tackle the multiplicity of solutions, we used a data-driven solver referred to as the weighted primal-dual method (WPDM) in a previous study. This method calculates a particular optimal solution based on the choice of interior-point weights that determine the search direction of the interior-point algorithm toward the optimal solution. WPDM is defined in Eq. (5.4) and it is denoted as the P_w form of the LP in Eq. (5.3). For each time step k, the flux $\boldsymbol{v_k}$ can be found from the WPDM and then the solution is substituted into Eqs. (5.1) to

obtain the states at the next time interval.

$$
\begin{aligned}
& \inf_{\mu \to 0, \boldsymbol{v_k}, \boldsymbol{z_k}} && \boldsymbol{c}^T \boldsymbol{v_k} - \mu \sum_{i=1}^{n_G} w_i ln(z_{k,i}) && \text{(5.4a)} \\
P_w \qquad & \text{subject to} && \boldsymbol{G v_k} + \boldsymbol{z_k} = \boldsymbol{F \theta_k}(\boldsymbol{x_k}, q_k) + \boldsymbol{b} && \text{(5.4b)} \\
& && \boldsymbol{z_k} > \boldsymbol{0} && \text{(5.4c)}
\end{aligned}
$$

where μ is an infinitesimal constant, $\boldsymbol{w}$ is an interior-point weight vector and w_i is the *i-th* interior-point weight. $\boldsymbol{z}$ is a vector of slack variables that are added to convert the inequality (5.3b) to equality. The *i-th* interior-point weight w_i corresponds to a slack variable z_i and *i-th* constraint.

The suitability of WPDM for addressing the multiplicity of DFBMs is due to three main properties as follows: i- WPDM in Eq. (5.4) can approximate the original LP in Eq. (5.3) as μ tends to zero [98]. At the limit of μ equal to zero, the logarithmic barrier function $-\mu \sum_{i=1}^{n_G} w_i ln(z_{k,i})$ vanishes, and the objective in Eq. (5.4a) tends to Eq. (5.3a) because μ is controlled to decrease quicker than the increase of the logarithmic function. Since the slack variables are only auxiliary variables, the constraints defined by Eq. (5.4b)-(5.4c) and (5.3b) are equivalent. Therefore, when the optimal solution of the P form is unique, the WPDM will tend to this solution and corresponding objective value once μ is sufficiently small.

ii- WPDM provides a unique optimal solution [98]. WPDM is a variant of the interior-point method where the objective of the original LP is augmented with a weighted logarithmic barrier function $-\mu \sum_{i=1}^{n_G} w_i ln(z_{k,i})$ of slack variables. Since the resulting augmented objective of WPDM is strictly convex, the optimal solution is unique even though multiple optima are possible for the P form.

iii- By manipulating the interior-point weights, the obtained optimum can be directed towards a specific optimum when multiple optima coexist in the P form [98]. Even though all these optima may have the same objective function value, there is generally one particular optimal solution that will best fit the available experimental data. Therefore, the interior weights can be found by fitting experimental data. To the knowledge of the authors, WPDM is the only method that can be easily tuned to favor a solution that results in the best fitting of data. Most other methods for tackling multiplicity assume that the flux should be efficient and parsimonious [60, 47, 78]. While this assumption may be suitable for wild-type strains it may not be accurate for engineered strains that are used in pharmaceutical manufacturing processes [98].

The optimal solution $\boldsymbol{v}$ and $\boldsymbol{z}$ of the WPDM algorithm [98] can be obtained from the

K.K.T. condition according to Eq.(5.5) where for simplicity the subscript k is omitted.

$$\boldsymbol{G^T\lambda} = -\boldsymbol{c} \tag{5.5a}$$

$$\boldsymbol{Gv} + \boldsymbol{z} = \boldsymbol{F\theta} + \boldsymbol{b} \tag{5.5b}$$

$$\lambda_i z_i = \mu w_i \quad \forall i = 1, \cdots, n_G \tag{5.5c}$$

$$\boldsymbol{z} > \boldsymbol{0} \quad \boldsymbol{\lambda} > \boldsymbol{0} \tag{5.5d}$$

where λ_i are positive multipliers for *i-th* constraint. Eq. (5.5c) is a weak complementary slackness condition. The right-hand side μ is infinitesimal but not zero. To satisfy the weak complementary slackness condition, either z_i or λ_i should be infinitesimal for any constraint but they cannot be infinitesimal at the same time.

5.3.3 Multiparametric Programming of DFBMs

Although WPDM can be used to calculate individual solutions of DFBMs for a particular set of initial conditions, it is not effective for propagating uncertainty in initial conditions since this will require solving many WPDM problems which will be computationally prohibitive. To address this computational challenge a multiparametric programming approach is introduced [2].

In the current study, a special combination of multiparametric linear programming (mpLP) and multiparametric nonlinear programming (mpNLP) is used to propagate uncertainty in initial conditions onto the states. mpNLP is required since the WPDM method converts the original LP into an NLP. The application of the parametric programming approach is motivated by the fact that $\boldsymbol{\theta}$ is a vector-valued function of state $\boldsymbol{x}$ and, as the state evolves, $\boldsymbol{\theta}$ also varies. Then, $\boldsymbol{\theta}$ is regarded as a varying parameter of the optimization problem. Multiparametric programming algorithms divide the parameter space $\boldsymbol{\Theta}$ into different critical regions and find the optimal solution expression for each critical region. If $\boldsymbol{\theta}$ is within a critical region for which the solution to the corresponding LP is unique then mpLP is used to pre-solve the corresponding LP problem. On the other hand, if $\boldsymbol{\theta}$ is within a critical region for which the solution to the LP is not unique then mpNLP is used to pre-solve the corresponding WPDM problem. The detailed definition of a critical region is given in a later section.

Criterion of Uniqueness and Multiplicity

For any feasible parameter value $\boldsymbol{\theta}$ it is necessary to determine whether the corresponding parameter space region of $\boldsymbol{\theta}$ has a unique solution or not. To determine uniqueness some concepts about active constraints and inactive constraints are introduced. For any optimal solution $\boldsymbol{v}$ to LP in Eq. (5.3b), *i-th* constraint is active if $\boldsymbol{G_i v} = \boldsymbol{F_i \theta} + b_i$ holds. On the other hand, for the optimal solution $\boldsymbol{v}$ of the WPDM in Eq. (5.4), it is not possible to determine if *i-th* constraint is active based on the value of the corresponding slack variable z_i since this variable will only become infinitesimal but not exactly zero. Instead, to decide whether the constraint is active when using WPDM, the weak complementary slackness condition can be used where multiplier $\lambda_i >> 0$ if the constraint is active and $\lambda_i \to 0$ if the constraint is inactive. In practice, if the slack variable is smaller than a small value, the *i-th* constraint is regarded as an active constraint. Otherwise, the constraint is inactive.

Let's define $\mathcal{A}(\boldsymbol{\theta})$ and $\mathcal{I}(\boldsymbol{\theta})$ as the set of indices corresponding to active constraints and inactive constraints respectively. Since a constraint being active or inactive is determined by the parameter $\boldsymbol{\theta}$, sets $\mathcal{A}(\boldsymbol{\theta})$ and $\mathcal{I}(\boldsymbol{\theta})$ are functions of $\boldsymbol{\theta}$. Without loss of generality, let assume $\boldsymbol{G}_{\mathcal{A}(\boldsymbol{\theta})}$ are linearly independent active constraints (linear redundant rows can always be removed by Gaussian elimination). Let define $|\cdot|$ as the cardinality of a finite countable set, i.e. the number of elements of a set. If $|\mathcal{A}(\boldsymbol{\theta})| = n_r$ for WPDM, the optimal solution to the original LP is unique. If $|\mathcal{A}(\boldsymbol{\theta})| < n_r$ for WPDM, multiple solutions to the original LP exist. Further use and explanation of this criterion is shown in the following section.

Multiparametic Linear Programming (mpLP)

For a given $\boldsymbol{\theta}$ in LP Eq. (5.3), let define the set of indices of active constraints as $\mathcal{A}$. Without loss of generality, assume $\boldsymbol{G}_{\mathcal{A}}$ are linear independent active constraints (linear redundant rows are removed by Gaussian elimination). If $|\mathcal{A}| = n_r$, namely $\boldsymbol{G}_{\mathcal{A}}$ is full rank, the inverse of $\boldsymbol{G}_{\mathcal{A}}$ exists. For these active constraints, $\boldsymbol{G}_{\mathcal{A}}\boldsymbol{v} = \boldsymbol{F}_{\mathcal{A}}\boldsymbol{\theta} + \boldsymbol{b}_{\mathcal{A}}$ holds. Hence, for the given $\boldsymbol{\theta}$, the optimal solution $\boldsymbol{v}$ can be obtained through Eq. (5.6), which defines an affine mapping with respect to $\boldsymbol{\theta}$ [10].

$$\boldsymbol{v_{LP}} = \boldsymbol{\mathcal{F}\theta} + \boldsymbol{\beta} \tag{5.6a}$$

$$\boldsymbol{\mathcal{F}} = \boldsymbol{G}_{\mathcal{A}}^{-1}\boldsymbol{F}_{\mathcal{A}} \tag{5.6b}$$

$$\boldsymbol{\beta} = \boldsymbol{G}_{\mathcal{A}}^{-1}\boldsymbol{b}_{\mathcal{A}} \tag{5.6c}$$

Then, by substituting Eq. (5.6) into the inactive constraints $\boldsymbol{G}_{\mathcal{I}}\boldsymbol{x} \leq \boldsymbol{F}_{\mathcal{I}}\boldsymbol{\theta} + \boldsymbol{b}_{\mathcal{I}}$, a

polyhedral region $\Theta_{\mathcal{A}}$ described by Eq. (5.7) in the parameter space $\boldsymbol{\Theta}$ can be obtained.

$$\Theta_{\mathcal{A}} = \left\{ \boldsymbol{\theta} \in \Theta \mid -\boldsymbol{F}_{\mathcal{I}}\boldsymbol{\theta} < \boldsymbol{b}_{\mathcal{I}} - \boldsymbol{G}_{\mathcal{I}}(\boldsymbol{G}_{\mathcal{A}}^{-1}\boldsymbol{F}_{\mathcal{A}}\boldsymbol{\theta} + \boldsymbol{G}_{\mathcal{A}}^{-1}\boldsymbol{b}_{\mathcal{A}}) \right\} \tag{5.7}$$

The resulting polyhedral region $\Theta_{\mathcal{A}}$ defined by Eq. (5.7) is referred to as a critical region. For $\forall \boldsymbol{\theta} \in \Theta_{\mathcal{A}}$ (except the boundary of $\Theta_{\mathcal{A}}$), the set of indices of the active constraints $\mathcal{A}$ is same and Eq. (5.6) also applies for $\forall \boldsymbol{\theta} \in \Theta_{\mathcal{A}}$ (including the boundary of $\Theta_{\mathcal{A}}$). For a given $\boldsymbol{\theta}$, Eq. (5.7) can be used to verify whether $\boldsymbol{\theta}$ is inside or outside. If it is within the $\Theta_{\mathcal{A}}$, the optimal solution can be calculated directly from Eq. (5.6). Therefore, the optimization problem can be solved a priori as a function of $\boldsymbol{\theta}$. The mpLP algorithm consists in finding all these critical regions and calculating their corresponding optimal solution expressions as functions of $\boldsymbol{\theta}$ for further use in control or prediction problems.

A particular challenge arises for the case $|\mathcal{A}| < n_r$ where the rank of $\boldsymbol{G}_{\mathcal{A}}$ is deficient. For this case, the critical region is defined by Eq. (5.8), which combines active and inactive constraints to define a corresponding polyhedron.

$$\Theta_{\mathcal{A}} = \left\{ \boldsymbol{\theta} \in \Theta \mid -\boldsymbol{F}_{\mathcal{A}}\boldsymbol{\theta} = \boldsymbol{b}_{\mathcal{A}} - \boldsymbol{G}_{\mathcal{A}}\boldsymbol{v}, -\boldsymbol{F}_{\mathcal{I}}\boldsymbol{\theta} < \boldsymbol{b}_{\mathcal{I}} - \boldsymbol{G}_{\mathcal{I}}\boldsymbol{v} \right\} \tag{5.8}$$

Within this critical region, the active constraints $\boldsymbol{G}_{\mathcal{A}}\boldsymbol{v} = \boldsymbol{F}_{\mathcal{A}}\boldsymbol{\theta} + \boldsymbol{b}_{\mathcal{A}}$ will have at least one degree of freedom because of the rank deficiency and the optimal solution of $\boldsymbol{v}$ is not unique. Typical mpLP algorithms are based on simplex solvers which cannot find such solutions and critical regions. For these cases, the optimal solutions can be at interior points and not only at the vertices of the feasible decision space. However, interior-point algorithms seek an optimal solution within the feasible space and never reach the boundary exactly so that WPDM can find the optimal solution in the interior that simplex solvers cannot.

When WPDM is used for a given $\boldsymbol{\theta}$, if the original solution of LP is unique, the solution of WPDM will be within an infinitesimal neighborhood of the solution of LP at the vertex. If exact zero tolerance is used for convergence, the number of active constraints is n_r. When the original solution of the LP is not unique, WPDM converges to an interior-point optimum that is determined by the choice of the interior-point weights. Since the optimal WPDM solution may be an interior point far from the vertices, the number of active constraints is less than n_r. Accordingly, a critical region for which the solution of the LP is not unique can be identified according to Eq. (5.8). This property is used in the later algorithm to determine whether a critical region has a unique solution or not.

Multiparametric Nonlinear Programming

If $|\mathcal{A}| = n_r$, mpLP can be used to obtain a critical region and the corresponding optimal solution from Eq. (5.6) and Eq. (5.7). However, if $|\mathcal{A}| < n_r$, the critical region has multiple solutions. In this case, WPDM is required to obtain a particular solution for such a critical region with multiplicity. Since WPDM transforms the original LP into an NLP due to the presence of the logarithmic term in the objective function, mpNLP is used instead of mpLP to obtain the optimal solution expression. While different mpNLP methods have been proposed [26], multiparametric quadratic programming (mpQP) is used in this study to approximate the solution of the mpNLP in the critical region with multiplicity [50, 26]. The approximation involves dividing the critical region with multiplicity into zones and solving local mpQPs for each zone. A typical nonlinear parametric programming with linear constraints for a given polyhedral region $\mathcal{Q}$ is defined as Eq.(5.9). In this study, the region $\mathcal{Q}$ refers to a critical region with multiplicity or a subset of the critical region with multiplicity.

$$
\begin{aligned}
P_{NLP} \quad & \min_{\boldsymbol{v}} && f(\boldsymbol{v}, \boldsymbol{\theta}) && \text{(5.9a)} \\
& \text{subject to} && \boldsymbol{A}\boldsymbol{v} < \boldsymbol{K}\boldsymbol{\theta} + \boldsymbol{d} && \text{(5.9b)} \\
& && \boldsymbol{A_e}\boldsymbol{v} = \boldsymbol{K_e}\boldsymbol{\theta} + \boldsymbol{d_e} && \text{(5.9c)} \\
& && \boldsymbol{\theta} \in \mathcal{Q} && \text{(5.9d)}
\end{aligned}
$$

where objective function is $f : \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_r} \mapsto \mathbb{R}$; $\boldsymbol{A}$, $\boldsymbol{A_e}$, $\boldsymbol{K}$ and $\boldsymbol{K_e}$ are matrices at proper dimension. Here, parameter $\boldsymbol{\theta}$ denotes a point within the given region $\mathcal{Q}$.

Let $\boldsymbol{v_*}$ denotes the optimal solution of problem P_{NLP} for given parameter $\boldsymbol{\theta_*}$. In this study, the geometric center of a zone or critical region is used as the $\boldsymbol{\theta_*}$ to perform the approximation of the original mpNLP because the geometric center is easy to calculate. A local quadratic programming problem P_{QP} defined as Eq.(5.10) is employed to approximate Eq.(5.9) at $\boldsymbol{\theta_*}$ and $\boldsymbol{v_*}$. Specifically, a quadratic convex surface is used to approximate the

nonlinear convex surface in the neighborhood of the parameter $\boldsymbol{\theta_*}$ and optimal point $\boldsymbol{v_*}$.

$$
\begin{aligned}
P_{QP} \quad & \min_{\boldsymbol{v}} && \frac{1}{2}(\boldsymbol{v}-\boldsymbol{v_*})^T\boldsymbol{H}(\boldsymbol{v}-\boldsymbol{v_*}) + (\boldsymbol{L}^T + (\boldsymbol{\theta}-\boldsymbol{\theta_*})^T\boldsymbol{N})(\boldsymbol{v}-\boldsymbol{v_*}) \\
& && + g(\boldsymbol{v_*},\boldsymbol{\theta_*}) && (5.10a) \\
& \text{subject to} && \\
& && \boldsymbol{A}\boldsymbol{v} < \boldsymbol{K}\boldsymbol{\theta} + \boldsymbol{d} && (5.10b) \\
& && \boldsymbol{A_e}\boldsymbol{v} = \boldsymbol{K_e}\boldsymbol{\theta} + \boldsymbol{d_e} && (5.10c) \\
& && \boldsymbol{\theta} \in \mathcal{Q} && (5.10d)
\end{aligned}
$$

where $\boldsymbol{H} = \nabla^2_{\boldsymbol{vv}} f(\boldsymbol{v},\boldsymbol{\theta_*})$, $\boldsymbol{L} = \nabla_{\boldsymbol{v}} f(\boldsymbol{v_*},\boldsymbol{\theta_*})$, $\boldsymbol{N} = \nabla^2_{\boldsymbol{\theta v}} f(\boldsymbol{v_*},\boldsymbol{\theta_*})$, $g(\boldsymbol{v_*},\boldsymbol{\theta_*}) = f(\boldsymbol{v_*},\boldsymbol{\theta_*}) + \nabla^T_{\boldsymbol{\theta}} f(\boldsymbol{v_*},\boldsymbol{\theta_*})(\boldsymbol{v}-\boldsymbol{v_*}) + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta_*})\nabla^2_{\boldsymbol{\theta\theta}} f(\boldsymbol{v_*},\boldsymbol{\theta_*})(\boldsymbol{\theta}-\boldsymbol{\theta_*})$.

Without loss generality, let's assume $\boldsymbol{A}$ and $\boldsymbol{A_e}$ are linearly independent. Let assume $\boldsymbol{H}$ is nonsingular and the active set of Eq.(5.10b) is $\mathcal{A}$. Define $\boldsymbol{A_C} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{A}} \\ \boldsymbol{A_e} \end{bmatrix}$, $\boldsymbol{K_C} = \begin{bmatrix} \boldsymbol{K}_{\mathcal{A}} \\ \boldsymbol{K_e} \end{bmatrix}$, $\boldsymbol{d_C} = \begin{bmatrix} \boldsymbol{d}_{\mathcal{A}} \\ \boldsymbol{d_e} \end{bmatrix}$. Then, the explicit solution $\boldsymbol{v_{QP}}$ for a given optimal active set $\mathcal{A}$ is given by Eq.(5.11), which is also an affine mapping of $\boldsymbol{\theta}$ [7].

$$
\begin{aligned}
\boldsymbol{v_{QP}} &= \boldsymbol{\mathcal{F}}\boldsymbol{\theta} + \boldsymbol{\beta} && (5.11a) \\
\boldsymbol{\mathcal{F}} &= \boldsymbol{H}^{-1}\boldsymbol{A_C}^T(\boldsymbol{A_C}\boldsymbol{H}^{-1}\boldsymbol{A_C}^T)^{-1}(\boldsymbol{K_C} + \boldsymbol{A_C}\boldsymbol{H}^{-1}\boldsymbol{N^T}) - \boldsymbol{H}^{-1}\boldsymbol{N}^T && (5.11b) \\
\boldsymbol{\beta} &= \boldsymbol{H}^{-1}\boldsymbol{A_C}^T(\boldsymbol{A_C}\boldsymbol{H}^{-1}\boldsymbol{A_C}^T)^{-1}(\boldsymbol{d_C} + \boldsymbol{A_C}\boldsymbol{H}^{-1}\boldsymbol{P^T}) - \boldsymbol{H}^{-1}\boldsymbol{P}^T && (5.11c)
\end{aligned}
$$

where $\boldsymbol{P} = \boldsymbol{L} - \boldsymbol{H}\boldsymbol{v_*} - \boldsymbol{N}^T\boldsymbol{\theta_*}$.

In contrast with the optimal solution expression in Eq. (5.6), Eq.(5.11) only applies in the neighborhood of $\boldsymbol{\theta_*}$ and $\boldsymbol{v_*}$ because it is a local approximation. To improve accuracy, $\mathcal{Q}$ needs to be divided into smaller zones until convergence. In the neighborhood of $\boldsymbol{\theta_*}$, the error due to the approximation of the mpNLP is bounded by $\boldsymbol{v_{QP}}(\boldsymbol{\theta}) - \boldsymbol{v_{NLP}}(\boldsymbol{\theta}) = \mathcal{O}(\|\boldsymbol{\theta}-\boldsymbol{\theta_*}\|_2^2)$ [50]. Once a zone is small enough, the QP approximation at the geometric center of the zone can reach high accuracy. To divide the $\mathcal{Q}$ efficiently so that fewer divisions are required and high accuracy can be reached a k-d tree partition method is utilized in this investigation which has been used for mpNLP [50]. Here, the original k-d tree partition algorithm [39] has been modified for use with our WPDM. As shown schematically in Fig.(5.1) the partition strategy is simple and efficient to implement. In this investigation, critical regions and zones can be expressed as polyhedrons since all constraints are linear.

To partition a polyhedron of $\mathcal{Q}$ with n_θ dimensions through its middle point, there are n_θ options for partition of $\mathcal{Q}$ into two polyhedrons $\mathcal{Q}_1$ and $\mathcal{Q}_2$.

For a given division, the larger the distance from the geometric center $\boldsymbol{\theta}_*$, the larger the approximation error is because both the quadratic surface and the WPDM surface are strictly convex surfaces with respect to $\boldsymbol{v}$ and $\boldsymbol{\theta}$. Therefore, the deviations at the vertices between the approximated and actual values are good measures of the approximation accuracy of a particular division. Assuming that the set of vertexes of polyhedron $\mathcal{Q}$ are $\mathcal{V}(\mathcal{Q})$, the geometric center of polyhedron $\mathcal{Q}$ is $\mathcal{C}(\mathcal{Q})$, the division error ϵ_r through the r-th dimension is defined as Eq.(5.12). $|\mathcal{V}(\mathcal{Q})|$ denotes the cardinality of set $\mathcal{V}(\mathcal{Q})$, namely the number of vertices of $\mathcal{Q}$. Because different polyhedrons have different numbers of vertices, the division error needs to consider the number of vertices.

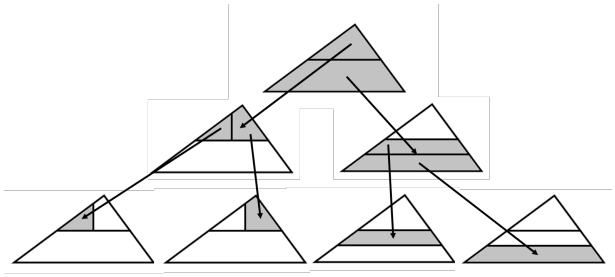


Figure 5.1: K-d tree partition of a critical region

$$\epsilon_r = \frac{1}{|\mathcal{V}(\mathcal{Q}_1)|} \sum_{\substack{\boldsymbol{\theta} \in \mathcal{V}(\mathcal{Q}_1) \\ \boldsymbol{\theta}_* = \mathcal{C}(\mathcal{Q}_1)}} \|\boldsymbol{v}_{\boldsymbol{QP}}(\boldsymbol{\theta}) - \boldsymbol{v}_{\boldsymbol{NLP}}(\boldsymbol{\theta})\| + \frac{1}{|\mathcal{V}(\mathcal{Q}_2)|} \sum_{\substack{\boldsymbol{\theta} \in \mathcal{V}(\mathcal{Q}_2) \\ \boldsymbol{\theta}_* = \mathcal{C}(\mathcal{Q}_2)}} \|\boldsymbol{v}_{\boldsymbol{QP}}(\boldsymbol{\theta}) - \boldsymbol{v}_{\boldsymbol{NLP}}(\boldsymbol{\theta})\| \tag{5.12}$$

After calculating the partition-related error for all dimensions, the dimension with the smallest error is the best dimension to divide a given polyhedron. Similarly, the approximation error of a given polyhedron $\mathcal{Q}$ is defined by Eq.(5.13).

$$\epsilon_{\mathcal{Q}} = \frac{1}{|\mathcal{V}(\mathcal{Q})|} \sum_{\substack{\boldsymbol{\theta} \in \mathcal{V}(\mathcal{Q}) \\ \boldsymbol{\theta}_* = \mathcal{C}(\mathcal{Q})}} \|\boldsymbol{v}_{\boldsymbol{QP}}(\boldsymbol{\theta}) - \boldsymbol{v}_{\boldsymbol{NLP}}(\boldsymbol{\theta})\| \tag{5.13}$$

If the approximation error $\epsilon_{\mathcal{Q}}$ is higher than the tolerance requirements specified by users for polyhedron $\mathcal{Q}$, the k-d tree partition method will continue dividing the polyhedron $\mathcal{Q}$

into smaller zones until either the requirements are satisfied or a minimal volume tolerance V_{min} is achieved.

When using the k-d tree partition algorithm, most zones are rectangular because the division of zones into sub-zones is axis-aligned and through the middle point of each dimension. Zones at the boundaries can be rectangular or general polyhedrons. There are several reasons that rectangular zones are used in this research instead of general shape polyhedrons. First, using general polyhedrons may not reduce the computation. For example, heuristic splitting proposed by Johansen et al. for mpNLP needs extra computation to define the boundaries of these general polyhedrons [39]. Second, online estimation requires fast computation, and set operations with rectangles are faster than with general shape polyhedrons. Third, when zones are not rectangular, the number of constraints defining the set of states in the later sections increases dramatically with time to over 500, which makes the calculation of set operations prohibitive and slow, such as vertexes. Many of the constraints of sets that are being increasingly added are similar and redundant but cannot be effectively removed by the algorithm because this requires precise control of numerical tolerances. In contrast, set operations with rectangular zones add much fewer constraints to the sets with small or no addition of redundant constraints.

Since the number of zones resulting from the k-tree approach may be very large, the computational expense required for further propagation of uncertainty is prohibitive. Thus, to reduce computations, regions in the parameter space that are not relevant can be trimmed as shown schematically in Fig (5.2). Such trimming is also justified because industrial fermentations are usually operated within a particular region of the parameter space. To perform trimming, Monte Carlo simulations with different initial states in set $\mathcal{X}_0$ are conducted to identify possible $\boldsymbol{\theta}$ at different times and different feeding policies. In Fig (5.2) the dots denote different $\boldsymbol{\theta}$ simulated by the Monte Carlo algorithm and rectangles denote different critical regions and zones occurring in time. Critical regions and zones that do not contain any assumed $\boldsymbol{\theta}$ are discarded to reduce the computational expense. To avoid leaving out possible $\boldsymbol{\theta}$ and to clearly identify the boundaries of critical regions, two actions are adopted in this research. First, 100000 batches simulated by Monte Carlo corresponding to about 40,100,000 $\boldsymbol{\theta}$ points, are used to identify the boundaries. Second, we check the connectivity between different CRs since $\boldsymbol{\theta}$ should be contiguous to each other.

Approximation of WPDM

While local quadratic programming can be used to approximate WPDM, this approximation has severe numerical problems in the case that the Hessian matrix $\boldsymbol{H}$ is almost singular. For this case, the quadratic programming solver cannot properly converge to

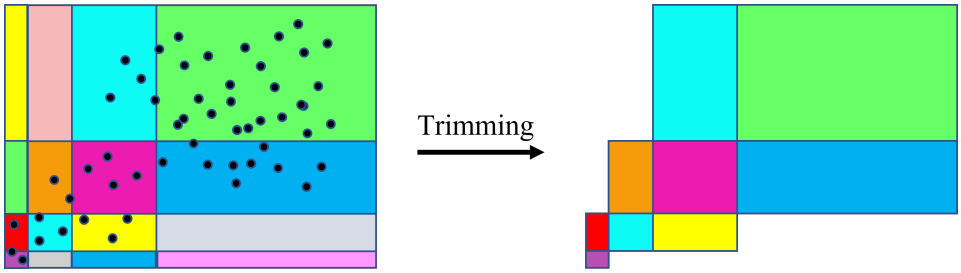


Figure 5.2: Trimming of unnecessary critical regions and zones

the optimal point $\boldsymbol{v}$. Interior-point methods are ill-defined around the optimal solution because the barrier function sharply increases to infinity towards the optimum. While the objective function increases mildly in some directions of the decision variables, the objective increases by a large amount in specific directions. Accordingly, the Hessian matrix has a large condition number and may become rank deficient thus resulting in a lack of convergence of the QP solution. A simple example is presented in Appendix A to help understand the numerical problems related to the Hessian matrix obtained from the QP approximation of the WPDM formulation.

To address the Hessian singularity an alternative formulation of the optimization P'_w is proposed. Let's assume for a given $\boldsymbol{\theta}_*$, the active and inactive constraint sets of the WPDM are $\mathcal{A}$ and $\mathcal{I}$ respectively, the optimal solution of P'_w is an approximation of problem P_w. Eq. (5.14) defines the P'_w form that is equivalent to problem P_w as shown below. Since the active and inactive constraints are known, the key idea for approximating problem P_w is that strong complementary slackness is used to replace the weak complementary slackness in WPDM, and active barrier functions are regarded as 0.

$$
\begin{aligned}
P'_w \qquad & \inf_{\boldsymbol{v},\boldsymbol{z}} && -\sum_{i\in\mathcal{I}} w_i ln(z_i) && \text{(5.14a)}\\
& \text{subject to} && \boldsymbol{G}\boldsymbol{v}+\boldsymbol{z}=\boldsymbol{F}\boldsymbol{\theta}+\boldsymbol{b} && \text{(5.14b)}\\
& && z_i = 0 \quad \forall i \in \mathcal{A} && \text{(5.14c)}\\
& && z_i > 0 \quad \forall i \in \mathcal{I} && \text{(5.14d)}
\end{aligned}
$$

Proof:

Based on knowledge of $\mathcal{A}$ and $\mathcal{I}$, Eq.(5.4) can be rearranged as follows:

$$\inf_{\mu \to 0, \boldsymbol{v}, \boldsymbol{z}} \quad \boldsymbol{c}^T \boldsymbol{v} - \mu \sum_{i \in \mathcal{A}} w_i ln(z_i) - \mu \sum_{i \in \mathcal{I}} w_i ln(z_i) \tag{5.15a}$$

$$\text{subject to} \quad \boldsymbol{G}\boldsymbol{v} + \boldsymbol{z} = \boldsymbol{F}\boldsymbol{\theta} + \boldsymbol{b} \tag{5.15b}$$

$$\boldsymbol{z} > \boldsymbol{0} \tag{5.15c}$$

As shown above, the objective can be separated into the active and inactive constraints-related terms as shown in Eq. (5.15a). Theoretically, for the active constraint, the term $-\sum_{i \in \mathcal{A}}^{m} w_i ln(z_i)$ become infinity as $z_{\mathcal{A}} \to 0$. However, in practice, the μ is controlled to decrease steadily in each iteration and faster than the increase of $-\sum_{i \in \mathcal{A}}^{m} w_i ln(z_i)$ so that $-\mu \sum_{i \in \mathcal{A}}^{m} w_i ln(z_i)$ in Eq. (5.15a) become infinitesimal and can be ignored from the objective [98].

Then, the KKT condition Eq.(5.5a) can be rearranged as in Eq.(5.16).

$$\begin{bmatrix} \boldsymbol{G}_{\mathcal{A}}^T & \boldsymbol{G}_{\mathcal{I}}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_{\mathcal{A}} \\ \boldsymbol{\lambda}_{\mathcal{I}} \end{bmatrix} = -\boldsymbol{c} \tag{5.16}$$

Using the weak complementary slackness condition in Eq. (5.5c), at the solution, $z_{\mathcal{A}} \to 0$ ($\lambda_{\mathcal{A}} > 0$) for the active constraints and $\lambda_{\mathcal{I}} \to 0$ ($z_{\mathcal{I}} > 0$) for the inactive constraints. Assuming at the solution $\boldsymbol{\lambda}_{\mathcal{I}} = 0$, Eq.(5.16) can be simplified to $\boldsymbol{G}_{\mathcal{A}}^T \boldsymbol{\lambda}_{\mathcal{A}} = -\boldsymbol{c}$. Also, $\boldsymbol{G}_{\mathcal{A}} \boldsymbol{v} = \boldsymbol{F}_{\mathcal{A}} \boldsymbol{\theta} + \boldsymbol{b}_{\mathcal{A}}$ is assumed to hold for the active constraints. Combining equations $\boldsymbol{G}_{\mathcal{A}}^T \boldsymbol{\lambda}_{\mathcal{A}} = -\boldsymbol{c}$ and $\boldsymbol{G}_{\mathcal{A}} \boldsymbol{v} = \boldsymbol{F}_{\mathcal{A}} \boldsymbol{\theta} + \boldsymbol{b}_{\mathcal{A}}$ gives $\boldsymbol{c}^T \boldsymbol{v} = -\boldsymbol{\lambda}_{\mathcal{A}}^T (\boldsymbol{F}_{\mathcal{A}} \boldsymbol{\theta} + \boldsymbol{b}_{\mathcal{A}})$ is obtained. This means that the term $\boldsymbol{c}^T \boldsymbol{v}$ is invariant with respect to the decision variables and thus it can be eliminated from the objective function (5.15a). Hence $-\mu \sum_{i \in \mathcal{I}} w_i ln(z_i)$ is the only term remaining in the objective Eq. (5.15a). Since the strong complementary condition is assumed, constraints Eq. (5.15b) become constraints of Eqs. (5.14b)-(5.14d). Thus, P'_w form is an approximation of P_w.

Fig. (5.3) summarizes the relationship between different forms of the P problem and the reason to construct different forms. NLP problem P_w is an approximation of LP problem P to tackle a multiplicity of solutions. The QP problem P_{QP} is used to approximate the NLP problem P_w for multiparametric nonlinear programming. However, the use of a logarithmic barrier function in P_w causes the Hessian matrix in QP problem P_{QP} to be almost singular resulting in convergence issues. To address the Hessian singularity a new QP problem P'_{QP} that approximates a new NLP P'_w is used instead in multiparametric nonlinear programming.

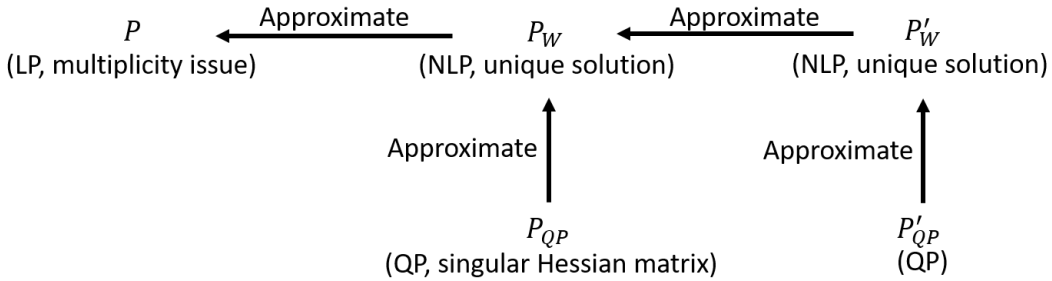


Figure 5.3: Relationship between different forms of P problem

The mpNLP algorithm itself cannot ensure the uniqueness of the optimal solution. On the other hand, the approximation of the problem by P'_{QP} and P'_w has a unique optimal solution. P'_w is strictly convex because its objective function involves a summation of logarithmic functions. The mpNLP algorithm uses local quadratic programming P'_{QP} to approximate P'_w in the neighborhood of the unique optimal solution $\boldsymbol{v}_*$ and corresponding parameter $\boldsymbol{\theta}_*$. Therefore, since the local quadratic programming cost is also strictly convex in the neighborhood of $\boldsymbol{v}_*$ and $\boldsymbol{\theta}_*$ it tends to the same unique solution $\boldsymbol{v}_*$ as the approximation is refined.

Multiparametric Programming Algorithm for DFBMs

This section summarizes the multiparametric programming based algorithms needed to find optimal solutions for different critical regions arising in the solutions of DFBMs. Set operations on polyhedrons are performed with the Multi-Parametric Toolbox 3.0 (https://www.mpt3.org/ accessed on Nov 8, 2021), including calculation of vertexes, faces, geometric center, and simplifying constraints of a polyhedron. Algorithm 1 is used to identify all critical regions occurring along the trajectories resulting from the DFBM solutions. After finding the first feasible $\boldsymbol{\theta}$, algorithm 1 solves the WPDM to determine active constraints and the first critical region. Then for each boundary of the found critical region, it finds a point outside of the current boundary corresponding to a new θ value. WPDM is then used to find the optimum at the new θ and the algorithm proceeds until all critical regions are found. If a replicate of a critical region is found, it is discarded and only new critical regions are kept.

Fig. (5.4) is presented to illustrate the idea of algorithm 1. In Fig. (5.4a), the square represents the space of $\boldsymbol{\theta}$. Based on the initial $\boldsymbol{\theta}$ assumed, the first critical region in the $\boldsymbol{\theta}$ space is obtained by solving problem P_w. In Fig. (5.4b), the blue triangle denotes the

first critical region that was found with three boundaries. For each boundary, we can find an outside point $\boldsymbol{\theta}$ as explained in algorithm 1 line 7. For example, in Fig. (5.4c), the red dot denotes an outsider point $\boldsymbol{\theta}$ of a boundary of the blue triangle. Solving problem P_w at this red dot $\boldsymbol{\theta}$ a new critical region is found denoted by the green triangle in Fig. (5.4d). Similarly, other outsider points are investigated for the other boundaries of the blue triangle, and the same procedure is repeated to obtain the other two critical regions denoted by the yellow and orange polyhedrons in Fig. (5.4e). Subsequently, For each boundary of the newly obtained critical regions, additional outsider points are explored to find new adjacent critical regions such as the one denoted by the purple triangle in Fig. (5.4f). This procedure is repeated until all critical regions within the $\boldsymbol{\theta}$ space are found. For all the critical regions that were obtained, algorithm 2 is used to check whether a given critical region has a unique solution expression or not. If the critical region under investigation has a unique solution, mpLP is used to find a solution. On the other hand, if the given critical region has multiplicity, the k-d tree partition algorithm is used to find the best direction to divide the critical regions into smaller zones. Then, for each of these smaller regions, QP approximations are made.

Algorithm 1 Multiparametric Programming for DFBMs

1: Initialize set $\mathcal{S}$ and $\mathcal{M}$ as $\varnothing$
2: Find a point $\boldsymbol{\theta}$ where problem P is feasible and use it as an initial point
3: Solve problem P_w at $\boldsymbol{\theta}$ and determine active constraint set $\mathcal{A}$
4: Call critical region algorithm
5: Let $ii = 1$, number of critical region being explored $n_{CR} = 1$
6: **while** $ii \leq n_{CR}$ **do**
7: **for each** boundary of *ii-th* critical region **do**
8: Find an outside point $\boldsymbol{\theta}$ in the neighbor of the current boundary
9: Solve problem P_w at $\boldsymbol{\theta}$ and determine active constraint set $\mathcal{A}$
10: Call critical region algorithm
11: **end for**
12: **if** new critical region is found **then**
13: $ii = ii + 1$
14: **end if**
15: **end while**

Algorithm 2 is used to determine for a given set of active constraints whether the corresponding solution is unique or not. If the solution is unique ($|\mathcal{A}| = n_r$), mpLP is used to define the critical region and its optimizer. If multiple solutions exist ($|\mathcal{A}| < n_r$), the k-d

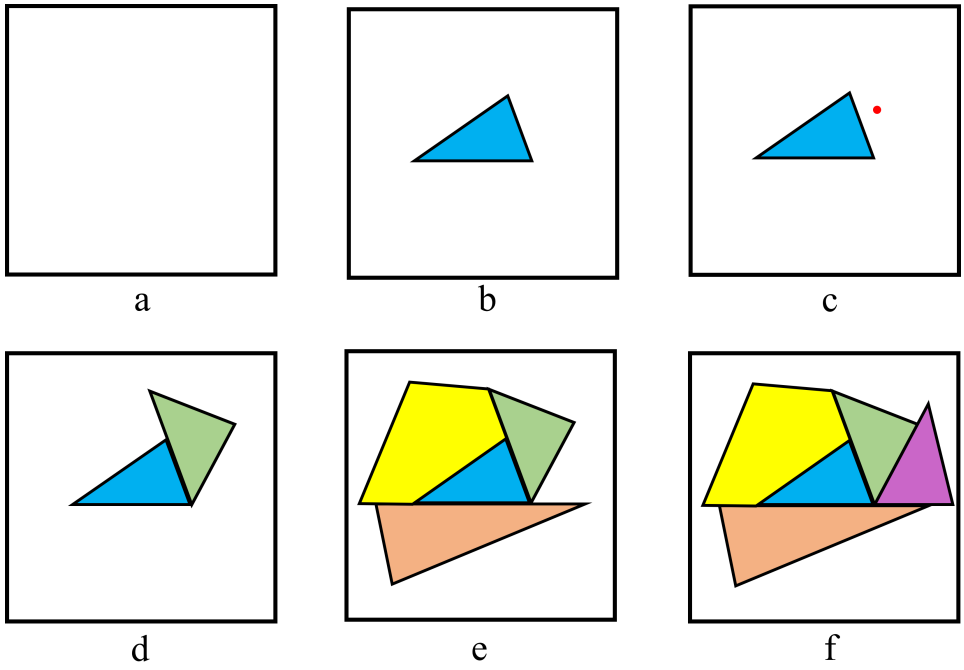


Figure 5.4: Illustration of algorithm 1

tree algorithm is applied to divide the current critical region into zones and mpNLP is used to obtain optimizing expressions for each zone. For the special case with a unique solution of $|\mathcal{A}| > n_r$, n_r constraints are used to define a new critical region and corresponding optimizer. If a duplicate critical region is found, the duplicate region is discarded.

Algorithm 3 is used to divide the critical region with multiplicity by using the k-d tree partition algorithm. If the critical region found in algorithm 1 has multiplicity, k-d tree partition algorithm is used to split the criterion region into many zones to construct the QP approximation until the accuracy is sufficient. The polyhedron that needs to be split is defined as $\mathcal{Q}$. Divide $\mathcal{Q}$ into two polyhedrons $\mathcal{Q}_1$ and $\mathcal{Q}_2$ through the middle point of each dimension. Select geometric centers of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ as $\boldsymbol{\theta}$ respectively. Solve problem P_w to determine active constraints $\mathcal{A}$. Based on active constraints $\mathcal{A}$, determine the corresponding optimizers for $\mathcal{Q}_1$ and $\mathcal{Q}_2$. Polyhedrons $\mathcal{Q}_1$ and $\mathcal{Q}_2$ need to be split until a user's defined convergence is satisfied or a minimum volume limit is achieved. Monte Carlo simulations are conducted to simulate different initial values and feeding policies with different $\boldsymbol{\theta}$. If the identified polyhedrons do not contain any simulated $\boldsymbol{\theta}$, they can be trimmed to improve computational efficiency.

Algorithm 2 Critical Region Algorithm

if $|\mathcal{A}| = n_r$ **then**
  if $\mathcal{A} \notin \mathcal{S}$ **then**
    Add $\mathcal{A}$ in set $\mathcal{S}$
    Determine the new critical region as Eq. (5.7)
    Determine the optimal solution expression as Eq. (5.6) based on problem P
  end if
else if $|\mathcal{A}| > n_r$ **then**
  Select the n_r indices of active constraints from $\mathcal{A}$ as new $\mathcal{A}$.
  The new set $\mathcal{A}$ must satisfy the following conditions:
    $\mathcal{A} \notin \mathcal{S}$
    $\forall \tilde{\mathcal{A}} \in \mathcal{M}$, $\tilde{\mathcal{A}} \nsubseteq \mathcal{A}$
  if $\mathcal{A} \neq \varnothing$ **then**
    Add $\mathcal{A}$ in set $\mathcal{S}$
    Determine the new critical region according to Eq. (5.7)
    Determine the optimal solution expression as per Eq. (5.6) based on problem P
  end if
else
  if $\mathcal{A} \notin \mathcal{M}$ **then**
    Add $\mathcal{A}$ in set $\mathcal{M}$
    Determine the new critical region as Eq. (5.8)
    Apply the k-d tree Algorithm to divide the newly found critical region into smaller regions for QP approximations
  end if
end if

Algorithm 3 k-d tree partition algorithm

Initialize set $\mathcal{B}$ as $\varnothing$ Identify the current critical region that needs to be split and add it to set $\mathcal{B}$
while if any polyhedron in set $\mathcal{B}$ need to be split **do**
    Select the first polyhedron that needs to be split as $\mathcal{Q}$
    for each dimension of polyhedron $\mathcal{Q}$ **do**
        Divide into $\mathcal{Q}_1$ and $\mathcal{Q}_2$ through its middle point along the current dimension
        Calculate $\mathcal{C}(\mathcal{Q}_1)$ and $\mathcal{C}(\mathcal{Q}_2)$ respectively
        Let $\boldsymbol{\theta}_*$ be $\mathcal{C}(\mathcal{Q}_1)$ and $\mathcal{C}(\mathcal{Q}_2)$ and solve problem P_w at $\boldsymbol{\theta}_*$ respectively to obtain $\boldsymbol{v}_*$
        Construct problem P'_w based on the active constraints $\mathcal{A}$, $\boldsymbol{\theta}_*$ and $\boldsymbol{v}_*$ for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively as Eq. (5.14)
        Use problem P'_w to approximate the solution of problem P_{NLP} and determine the corresponding optimizer $\boldsymbol{v_{QP}}$ for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively as Eq.(5.11)
        Calculate ϵ_r for the current dimension according to Eq. (5.12)
        Calculate $\epsilon_{\mathcal{Q}}$ for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ as Eq. (5.13)
        if $\epsilon_{\mathcal{Q}}$ is larger than convergence tolerance and volume is larger V_{min} **then** Mark the corresponding polyhedron that needs to be split
        end if
    end for
    Select the dimension with smallest ϵ_r to divide $\mathcal{Q}$ into $\mathcal{Q}_1$ and $\mathcal{Q}_2$
    Remove $\mathcal{Q}$ from from set $\mathcal{B}$
    Add the corresponding polyhedrons $\mathcal{Q}_1$ and $\mathcal{Q}_2$ in set $\mathcal{B}$
    Trim the irrelevant polyhedrons that do not contain any point simulated by Monte Carlo Simulation for different initial values and feeding policies
end while

Conversion of DFBM into Variable Structure System

Using the multiparametric programming approach proposed above, the feasible parameter space is ultimately divided into different critical regions. Each critical region with a unique solution has one optimal solution expression as $\boldsymbol{v_{LP}} = \boldsymbol{\mathcal{F}\theta}+\boldsymbol{\beta}$. Each zone in critical regions with multiplicity also has one optimal solution expression $\boldsymbol{v_{QP}} = \boldsymbol{\mathcal{F}\theta} + \boldsymbol{\beta}$. Both critical regions with unique solutions and zones in critical regions with multiplicity have the same type of expression $\boldsymbol{v} = \boldsymbol{\mathcal{F}\theta} + \boldsymbol{\beta}$, which is an affine mapping with respect to $\boldsymbol{\theta}$. Since the critical regions with unique solutions and zones in critical regions with multiplicity do not overlap, they can be individually denoted as regions $\{\Theta^i\}$. Accordingly, the feasible

parameter space is divided into a series of regions $\{\Theta^i\}$ with an optimal solution expression for the *i-th* region given by Eq. (5.17).

$$\boldsymbol{v}^i = \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta} + \boldsymbol{\beta}^i \quad \boldsymbol{\theta} \in \Theta^i \tag{5.17}$$

The state equation Eq. (5.1a) can be expressed as a function of the uncertain parameters by substituting Eq. (5.17) into $\boldsymbol{v}$ to obtain Eq. (5.18).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B}(x_{v,k}, q_k)\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{S}(\boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) + \boldsymbol{\beta}^i) + \boldsymbol{h}(x_{v,k}, q_k) \quad \boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) \in \Theta^i \tag{5.18}$$

Such substitution simplifies the original problem by eliminating the inner optimization problem and permitting the propagation of the uncertainty over time. It should be noticed that $\boldsymbol{\theta}$ is a function of both the states $\boldsymbol{x}$ and the feed flow rate q_k where the latter is assumed to be known at all times. The states $\boldsymbol{x}$ are described by different state equations because $\boldsymbol{\theta}$ resides within different regions at different time intervals. Thus, following the substitution of the optimizers obtained with the multi-parametric approach, the original state equations are converted into a family of state equations given by Eq. (5.18). The latter state description is referred to in the literature as a variable structure system where each state equation corresponds to a subsystem. Different regions of state space belong to different subsystems.

5.3.4 Set Membership Estimation

Unlike classical observers, set membership estimation (SME) estimates the bounds of state set under the uncertainty of initial states and parameters. In general, SME uses different shapes of sets to contain all possible states and to propagate in time the state set by set operations. Commonly used set operations include scaling, affine mapping, projection, Cartesian product, translation, Minkowski addition, intersection, union, minimum, max, and outer approximation. In this research, these set operations are performed with the Multi-Parametric Toolbox 3.0 mainly (https://www.mpt3.org/ accessed on Nov 8, 2021) [43] and Bensolve (http://bensolve.org/ accessed on Oct 28, 2021) [62]for some operations. The state equation can be interpreted as combinations of different set operations and using larger sets to overestimate the states is always a safe choice.

For the application of SME to DFMB, several notations of set operations are introduced first. The Cartesian product of sets $\mathcal{N}$ and $\mathcal{P}$ is denoted by $\mathcal{N} \times \mathcal{P} = \{(n, p) : n \in \mathcal{N}\ and\ p \in \mathcal{P}\}$. The Cartesian product can be interpreted as connecting sets along different dimensions to generate a new set. The interval set is defined as $\mathcal{R}(\boldsymbol{l}, \boldsymbol{u}) = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{l} \leq$

$\boldsymbol{x} \leq \boldsymbol{u}\}$. An Interval set is a multi-dimensional axis-aligned interval, which is easy to use for set propagation because of its simple shape. The outer approximation of the set $\mathcal{P}$ is $Out(\mathcal{P}) = \mathcal{R}(\min(\mathcal{P}), \max(\mathcal{P}))$. An Outer approximation involves the use of a large interval set for containing an irregularly shaped set. The operator $Proj_i\mathcal{P}$ projects the set $\mathcal{P}$ onto the *i-th* dimension. The Projection operator is the inverse of the Cartesian product and it is used to simplify the geometry of a set. Similarly, if $\mathcal{N}$ is a set of dimensions, $Proj_{\mathcal{N}}\mathcal{P}$ projects the set $\mathcal{P}$ onto the dimensions $\mathcal{N}$. $\mathcal{C}(\mathcal{P})$ denotes the geometric center of a set $\mathcal{P}$. While for a given convex set different types of centers can be used, the geometric center is used for simplicity.

Update States with Measurements

SME is used in this study to estimate bounds on metabolites' concentrations at different time intervals based on limited measurements. We assume that the initial state can be bounded by a polyhedral set $\mathcal{X}_0$ as Eq. (5.1d). The measurements contain noise $\boldsymbol{r}$ bounded by interval set $\mathcal{R}(\boldsymbol{l}, \boldsymbol{u})$ as Eq. (5.1c).

$$\boldsymbol{x_{k+1}} = \boldsymbol{B}(x_{v,k}, q_k)\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{S}(\boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) + \boldsymbol{\beta}^i) + \boldsymbol{h}(x_{v,k}, q_k) \qquad \boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) \in \Theta^i \tag{5.19}$$

Eq. (5.19) are the state equations describing the variable structure system obtained in the previous section. The states $\boldsymbol{x}$ in DFBMs includes metabolites' concentrations $\boldsymbol{x_{met}}$, biomass concentration x_{bio}, and volume of culture x_v. The matrix $\boldsymbol{B}$ and vector $\boldsymbol{h}$ are functions of x_v and flow rate q_k. For fed-batch operation, x_v is influenced by q_k while for batch operation, x_v is constant. Biomass concentration scales the flux term $\boldsymbol{S}(\boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) + \boldsymbol{\beta}^i)$ in Eq. (5.19). If the volume of culture x_v and biomass concentrations x_{bio} can be measured, the propagation of the state set can be simplified. Therefore, the SME proposed in this research assumes that volume of culture x_v and biomass concentrations x_{bio} are measured values denoted by $\boldsymbol{y} = [y_{bio}\ y_v]^T$. This is a reasonable assumption since biomass concentration and volume of culture are easier to measure than metabolites' concentrations $\boldsymbol{x_{met}}$. the measured values with noise are $\mathcal{Y}_k = \mathcal{R}(\boldsymbol{y_k} + \boldsymbol{l}, \boldsymbol{y_k} + \boldsymbol{u})$.

Let $\mathcal{X}_k^-$ denote the prior-estimate set containing all possible states at time step k, which is also a polyhedral set. To take advantage of measurements, the post-estimate set $\mathcal{X}_k^+$ that contain all possible state can be expressed as in Eq. (5.20).

$$\mathcal{X}_k^+ = Proj_{met}\mathcal{X}_k^- \times \mathcal{Y}_k \tag{5.20}$$

where subscript "met" is used to denote dimensions of metabolites. The resulting set $\mathcal{X}_k^+$

is the post-estimate set.

Propagation of Post-estimate Set to Parameter Set

After $\mathcal{X}_k^+$ is obtained, the set containing all possible $\boldsymbol{\theta}_k$ needs to be calculated. Since $\boldsymbol{\theta}$ is a nonlinear function of state $\boldsymbol{x}$ and feed flow rate q_k, a linearization of $\boldsymbol{\theta}$ with respect to the center of $\mathcal{X}_k^+$ is performed. Let's assume that $\boldsymbol{\theta}_k$ is first-order differentiable and the Jacobian matrix $\boldsymbol{\psi}_k$ is given by Eq. (5.21).

$$\boldsymbol{\psi}_k = \left.\frac{\partial \boldsymbol{\theta_k}}{\partial \boldsymbol{x_k}}\right|_{\boldsymbol{x_k}=C(\mathcal{X}_k^+)} \tag{5.21}$$

If the linearization point is the center of $\mathcal{X}_k^+$, the $\boldsymbol{\theta_k}$ can be expressed as Eq. (5.22). Correspondingly, the polyhedral set $\tilde{\boldsymbol{\Theta}}_k$ containing possible $\boldsymbol{\theta_k}$ can be defined as Eq. (5.23).

$$\boldsymbol{\theta}_k = \boldsymbol{\psi}_k(\boldsymbol{x_k} - \mathcal{C}(\mathcal{X}_k^+)) + \boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) \tag{5.22}$$

$$\tilde{\boldsymbol{\Theta}}_k = \boldsymbol{\psi}_k(\mathcal{X}_k^+ - \mathcal{C}(\mathcal{X}_k^+)) + \boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) \tag{5.23}$$

After the linearized $\boldsymbol{\theta}$ is substituted into the state equation of variable structure system Eq. (5.19), the new state equation is obtained as follows:

$$\begin{aligned}\boldsymbol{x_{k+1}} &= \boldsymbol{B}(x_{v,k}, q_k)\boldsymbol{x_k} + \Delta t x_{bio,k}\boldsymbol{S}(\boldsymbol{\mathcal{F}}^i(\boldsymbol{\psi}_k(\boldsymbol{x_k} - \mathcal{C}(\mathcal{X}_k^+)) + \boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k)) + \boldsymbol{\beta}^i) + \boldsymbol{h}(x_{v,k}, q_k) \\ &\qquad \boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) \in \Theta^i \quad (5.24a) \\ &= (\boldsymbol{B}(x_{v,k}, q_k) + \Delta t x_{bio,k}\boldsymbol{S}\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k)\boldsymbol{x_k} \\ &\qquad + \Delta t x_{bio,k}\boldsymbol{S}(-\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\mathcal{C}(\mathcal{X}_k^+) + \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) + \boldsymbol{\beta}^i) + \boldsymbol{h}(x_{v,k}, q_k) \\ &\qquad \boldsymbol{\theta_k}(\boldsymbol{x_k}, q_k) \in \Theta^i \quad (5.24b)\end{aligned}$$

It can be shown that Eq. (5.24b) generates tight sets since it combines terms containing $\boldsymbol{x_k}$ together.

Disassembly of $\mathcal{X}_k^+$ into different regions

The SME procedure described below can be viewed as the disassembly/assembly of a puzzle. For a given set of states, the set is initially broken into pieces for each region. For each piece, set operations are employed to estimate a corresponding new set for the next

time step. Finally, the new sets obtained from the different set operations are assembled again into a new set, and so forth.

Following the linearization in Eq. (5.23), the polyhedral set $\tilde{\boldsymbol{\Theta}}_k$ is generated that contains all possible $\boldsymbol{\theta_k}$. Since $\tilde{\boldsymbol{\Theta}}_k$ can encompass many regions $\{\Theta^i\}$, it needs to be divided into pieces and propagated separately because different $\boldsymbol{\theta}$, with their correspondingly different state equations, must be used to propagate the uncertainty over time using set operations. Let define $\tilde{\boldsymbol{\Theta}}_k^i$ as the intersection of $\tilde{\boldsymbol{\Theta}}_k$ with each region as per Eq. (5.25).

(5.25).

$$\tilde{\boldsymbol{\Theta}}_k^i = \tilde{\boldsymbol{\Theta}}_k \cap \Theta^i; \tag{5.25}$$

Then, two cases must be considered in terms of the feed flow rate: $q_k \neq 0$ (fed-batch) or $q_k = 0$ (batch operation). When the feed flow rate $q_k \neq 0$, $\boldsymbol{\psi}_k$ is full rank. The corresponding polyhedral set of states for each region i can then be expressed as in Eq. (5.26).

$$\mathcal{X}_k^{i+} = \boldsymbol{\psi}_k^{-1}(\tilde{\boldsymbol{\Theta}}_k^i - \boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k)) + \mathcal{C}(\mathcal{X}_k^+); \tag{5.26}$$

When the feed flow rate $q_k = 0$, $\boldsymbol{\psi}_k$ is rank deficient. Let's assume $\tilde{\boldsymbol{\psi}}_k$ is obtained by eliminating the linearly dependent row and column from $\boldsymbol{\psi}_k$. Then, the corresponding set for each region i can be expressed as in Eq. (5.27).

$$\mathcal{X}_k^{i+} = (\tilde{\boldsymbol{\psi}}_k^{-1}(\tilde{\boldsymbol{\Theta}}_k^i - \boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k)) + Proj_{(Met,bio)}\mathcal{C}(\mathcal{X}_k^+)) \times Proj_v\mathcal{Y}_k; \tag{5.27}$$

Eq. (5.27) indicates that the set of metabolites' concentrations and biomass can be recovered from Eq. (5.22) and using the measurement of culture volume.

Eq. (5.25) separates $\tilde{\boldsymbol{\Theta}}_k$ into pieces for each region i. Eq. (5.26) and Eq. (5.27) are used to relate the parameter region i to the corresponding state space set of equations. Now, the state equation Eq. (5.24b) can be expressed as Eq. (5.28) and $\boldsymbol{\theta}^i \in \Theta^i$ can be replaced by $\boldsymbol{x_k} \in \mathcal{X}_k^{i+}$.

$$\begin{aligned} \boldsymbol{x_{k+1}} = &(\boldsymbol{B}(x_{v,k}, q_k) + \Delta t x_{bio,k}\boldsymbol{S}\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k)\boldsymbol{x_k} \\ &+ \Delta t x_{bio,k}\boldsymbol{S}(-\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\mathcal{C}(\mathcal{X}_k^+) + \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) + \boldsymbol{\beta}^i) + \boldsymbol{h}(x_{v,k}, q_k) \\ &\boldsymbol{x_k} \in \mathcal{X}_k^{i+} \end{aligned} \tag{5.28}$$

Assembly of the sets of states from different regions into one set of states

Eq. (5.28) cannot be used for set propagation directly because $\boldsymbol{x_{k+1}}$ is a nonlinear function of $\boldsymbol{x_k}$ since $(\boldsymbol{B}(x_{v,k}, q_k) + \Delta t x_{bio,k}\boldsymbol{S}\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k)\boldsymbol{x_k}$ is a nonlinear mapping of intervals of values

of $\boldsymbol{x_k}$, $x_{v,k}$ and $x_{bio,k}$. Similarly, $\Delta t x_{bio,k}\boldsymbol{S}(-\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\mathcal{C}(\mathcal{X}_k^+) + \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) + \boldsymbol{\beta}^i)$ and $\boldsymbol{h}(x_{v,k}, q_k)$ are also defined as functions of intervals.

Assuming that the intervals of values covered by $\mathcal{X}_k^{i+}$ are small in comparison to intervals covered by $\mathcal{X}_k^+$, it is assumed that $\boldsymbol{B}(x_{v,k}, q_k)$ and $\boldsymbol{h}(x_{v,k}, q_k)$ can be approximated by substituting $\hat{x}_{v,k}^i = \mathcal{C}(Proj_v\mathcal{X}_k^{i+})$, i.e. for each region i, $\boldsymbol{B}(x_{v,k}, q_k) = \boldsymbol{B}(\hat{x}_{v,k}^i, q_k)$ and $\boldsymbol{h}(x_{v,k}, q_k) = \boldsymbol{h}(\hat{x}_{v,k}^i, q_k)$.

The uncertainty in $x_{bio,k}$ requires different treatment than the other states because $x_{bio,k}$ scales the vectors $\Delta t\boldsymbol{S}\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\boldsymbol{x_k}$ and $\Delta\boldsymbol{S}(-\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\mathcal{C}(\mathcal{X}_k^+)+\boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k)+\boldsymbol{\beta}^i)$ in Eq. (5.28). This is general for all DFBM models where all the fluxes are defined per unit biomass so that multiplying biomass is required to calculate metabolites' concentrations. Following this observation and for computational efficiency, a bounding set is obtained from an outer approximation operation with respect to biomass values. Although such an operation may be conservative it bypasses the need for nonlinear mapping scaling operations that are time-consuming.

Using the minimum and maximum of $x_{bio,k}^i$ given by $\underline{x}_{bio,k}^i$ and $\overline{x}_{bio,k}^i$, the set propagation can be approximated by affine mapping and outer approximation according to Eq. (5.29).

$$\mathcal{X}_{k+1}^i = Out(\underline{\mathcal{X}}_{k+1}^i \cup \overline{\mathcal{X}}_{k+1}^i) \tag{5.29a}$$

$$\begin{aligned}\underline{\mathcal{X}}_{k+1}^i = &(\boldsymbol{B}(\hat{x}_{v,k}^i, q_k) + \Delta t\underline{x}_{bio,k}^i\boldsymbol{S}\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k)\mathcal{X}_k^{i+} \\ &+ \Delta t\underline{x}_{bio,k}^i\boldsymbol{S}(-\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\mathcal{C}(\mathcal{X}_k^+) + \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) + \boldsymbol{\beta}^i) + \boldsymbol{h}(\hat{x}_{v,k}^i, q_k)\end{aligned} \tag{5.29b}$$

$$\begin{aligned}\overline{\mathcal{X}}_{k+1}^i = &(\boldsymbol{B}(\hat{x}_{v,k}^i, q_k) + \Delta t\overline{x}_{bio,k}^i\boldsymbol{S}\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k)\mathcal{X}_k^{i+} \\ &+ \Delta t\overline{x}_{bio,k}^i\boldsymbol{S}(-\boldsymbol{\mathcal{F}}^i\boldsymbol{\psi}_k\mathcal{C}(\mathcal{X}_k^+) + \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta_k}(\mathcal{C}(\mathcal{X}_k^+), q_k) + \boldsymbol{\beta}^i) + \boldsymbol{h}(\hat{x}_{v,k}^i, q_k)\end{aligned} \tag{5.29c}$$

Using Eq. (5.29), the set of states for each region at the next time step can be estimated. Then, an outer approximation is used to combine $\mathcal{X}_{k+1}^i$ for all regions to generate the prior-estimate set $\mathcal{X}_{k+1}^-$. Because of the outer approximation operation, the set $\mathcal{X}_{k+1}^-$ is an interval set.

$$\mathcal{X}_{k+1}^- = Out(\bigcup_i \mathcal{X}_{k+1}^i); \tag{5.30}$$

Starting from $\mathcal{X}_0^- = \mathcal{X}_0$, the initial set of states can be propagated using the procedure outlined above. Then, at any time step k, the bounds of states can be estimated by taking the maximum and minimum values of $\mathcal{X}_k^-$.

5.4 Results and Discussion

5.4.1 DFBM of E.coli

The proposed estimation approach is applied to a modified version of the DFBM model of *E. coli* reported in [64] for illustration. The DFBM includes five states glucose concentration x_{glc} in mM, oxygen concentration x_{oxy} in mM, acetate concentration x_{ace} in mM, biomass concentration x_{bio} in g-dw/L, and volume of culture x_v in L as Eq. (5.31) (g-dw refers to grams of the dry weight of biomass.). Hence, the state vector is $\boldsymbol{x} = \begin{bmatrix} x_{glc} & x_{oxy} & x_{ace} & x_{bio} & x_v \end{bmatrix}^T$. The substrates are glucose, oxygen, and acetate. Glucose is the only supplement nutrient used in fed-batch operations.

$$x_{glc,k+1} = (1 - \frac{q_k \Delta t}{x_{v,k}}) x_{glc,k} + \Delta t x_{bio,k} \boldsymbol{S_{glc}} \boldsymbol{v_k} + \frac{q_k \Delta t}{x_{v,k}} x_{glc,in} \tag{5.31a}$$

$$x_{oxy,k+1} = (1 - \frac{q_k \Delta t}{x_{v,k}} - k_L a \Delta t) x_{oxy,k} + \Delta t x_{bio,k} \boldsymbol{S_{oxy}} \boldsymbol{v_k} + 0.21 k_L a \Delta t \tag{5.31b}$$

$$x_{ace,k+1} = (1 - \frac{q_k \Delta t}{x_{v,k}}) x_{ace,k} + \Delta t x_{bio,k} \boldsymbol{S_{ace}} \boldsymbol{v_k} \tag{5.31c}$$

$$x_{bio,k+1} = (1 - \frac{q_k \Delta t}{x_{v,k}}) x_{bio,k} + \Delta t x_{bio,k} \boldsymbol{S_{bio}} \boldsymbol{v_k} \tag{5.31d}$$

$$x_{v,k+1} = x_{v,k} + q_k \Delta t \tag{5.31e}$$

$$\boldsymbol{x_0} \in \mathcal{X}_0 = \mathcal{R}(\begin{bmatrix} 0.38 & 0.1995 & 0.19 & 0.00095 & 0.285 \end{bmatrix}^T, \begin{bmatrix} 0.42 & 0.2205 & 0.21 & 0.00105 & 0.315 \end{bmatrix}^T) \tag{5.31f}$$

$$\boldsymbol{y_k} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \boldsymbol{x_k} + \boldsymbol{r_k} \tag{5.31g}$$

$$\boldsymbol{r_k} \sim TN(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \times 10^{-6} & 0 \\ 0 & 3 \times 10^{-8} \end{bmatrix}, \begin{bmatrix} -1 \times 10^{-3} \\ -0.0150 \end{bmatrix}, \begin{bmatrix} 1 \times 10^{-3} \\ 0.0150 \end{bmatrix}) \quad k = 0, 1, 2 \cdots \tag{5.31h}$$

Where $k_L a = 4\ h^{-1}$ is the oxygen mass transfer coefficient and $\Delta t = 0.025\ h$. The initial state vector $\boldsymbol{x_0}$ is assumed to be bounded by the interval set $\mathcal{X}_0$ as Eq. (5.31f). The matrix S contains the stoichiometric coefficients corresponding to four reactions according to Eq. (5.32). The last row of the stoichiometric matrix is made of zeros because the culture volume, corresponding to that row in the stoichiometric matrix, is not influenced

by reactions. Each column corresponds to one reaction and each row corresponds to one state. q_k is the feed flow rate at time step k. For batch operation, $q_k = 0$. For fed-batch operation, before the feeding is started $q_k = 0$ and $q_k = 0.02\ L/h$ of glucose from time $7.3h$ and on.

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S_{glc}} \\ \boldsymbol{S_{oxy}} \\ \boldsymbol{S_{ace}} \\ \boldsymbol{S_{bio}} \\ \boldsymbol{S_v} \end{bmatrix} = \begin{bmatrix} 0 & -9.46 & -9.84 & -19.23 \\ -35 & -12.92 & -12.73 & 0 \\ -39.43 & 0 & 1.24 & 12.12 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.32}$$

The flux vector $\boldsymbol{v_k}$ is obtained by solving the linear programming problem in Eq. (5.33):

$$\begin{aligned}
\max_{\boldsymbol{v_k}} \quad & \boldsymbol{S_{bio}}\boldsymbol{v_k} && \text{(5.33a)} \\
\text{subject to} \quad & -\boldsymbol{S_{oxy}}\boldsymbol{v_k} \leq OUR_{max} && \text{(5.33b)} \\
& \boldsymbol{S_{ace}}\boldsymbol{v_k} \leq 100 && \text{(5.33c)} \\
& -\Delta t \boldsymbol{S_{glc}}\boldsymbol{v_k} \leq (\frac{1}{x_{bio,k}} - \frac{q_k \Delta t}{x_{v,k} x_{bio,k}}) x_{glc,k} + \frac{q_k \Delta t x_{glc,in}}{x_{v,k} x_{bio,k}} = \theta_{1,k} && \text{(5.33d)} \\
& -\Delta t \boldsymbol{S_{oxy}}\boldsymbol{v_k} \leq (\frac{1}{x_{bio,k}} - \frac{k_L a \Delta t}{x_{bio,k}} - \frac{q_k \Delta t}{x_{v,k} x_{bio,k}}) x_{oxy,k} + \frac{0.21 k_L a \Delta t}{x_{bio,k}} \\
& \qquad\qquad = \theta_{2,k} && \text{(5.33e)} \\
& -\Delta t \boldsymbol{S_{ace}}\boldsymbol{v_k} \leq (\frac{1}{x_{bio,k}} - \frac{q_k \Delta t}{x_{v,k} x_{bio,k}}) x_{ace,k} = \theta_{3,k} && \text{(5.33f)} \\
& -\boldsymbol{S_{glc}}\boldsymbol{v_k} \leq \frac{GUR_{max} x_{glc,k}}{K_m + x_{glc,k}} = \theta_{4,k} && \text{(5.33g)} \\
& -\Delta t \boldsymbol{S_{bio}}\boldsymbol{v_k} \leq 1 - \frac{q_k \Delta t}{x_{v,k}} = \theta_{5,k} && \text{(5.33h)} \\
& -v_{1,k} \leq 0 && \text{(5.33i)} \\
& -v_{2,k} \leq 0 && \text{(5.33j)} \\
& -v_{3,k} \leq 0 && \text{(5.33k)} \\
& -v_{4,k} \leq 0 && \text{(5.33l)} \\
& \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \boldsymbol{v_k} \leq 0.5 && \text{(5.33m)}
\end{aligned}$$

Where $OUR_{max} = 12\text{mM}/(\text{g-dw}\cdot\text{h})$ is the maximum oxygen uptake rate; $GUR_{max} = 6.5$

mM/(g-dw·h) denotes the maximum glucose uptake rate. Eq. (5.33a) describes that the optimization objective of the cells is the maximization of the biomass growth rate. Eq. (5.33b) and Eq. (5.33c) indicate that the oxygen consumption rate and acetate generation rate are limited by maximal uptake limits respectively. Constraints Eqs. (5.33d), (5.33e), (5.33f), and (5.33h) force the concentrations to be positive. Eq. (5.33g) indicates that the glucose consumption rate is bounded by an upper limit that depends on glucose concentration. Since the RHS of these constraints are functions of the states the RHS terms are treated as time-varying parameters composing the vector $\boldsymbol{\theta}$. It should be noticed that $\theta_{5,k}$ is a function of the feed-rate q_k. As explained in the previous section, before the feeding starts or batch operation $q_k = 0$ resulting in the Jacobian matrix $\boldsymbol{\psi}_k$ to be rank deficient because $\theta_{5,k} = 1$ is a constant. Therefore, the volume cannot be estimated from $\theta_{5,k}$ during batch operation. Constraints Eq. (5.33i)-Eq. (5.33l) force the assumed directions of the reactions (fluxes).

The constraint Eq. (5.33m) was added to the original *E. coli* model. Since the coefficients of the fluxes in the growth rate (5.32) and Eq. (5.33m) are the same (all ones) this additional constraint imposes an upper bound on growth rate. The biochemical rationale for this additional constraint is that during certain periods of the culture, the growth rate may not be limited by the metabolites considered in the model, e.g. growth limitations due to depletion of growth factors that cannot be modeled or measured. While in the originally reported *E.coli* model the solution is unique, the addition of Eq. (5.33m) introduces multiplicity since the LHS coefficients of fluxes in Eq. (5.33m) are same as for $\boldsymbol{S_{bio}}$. The interior-point weights used in WPDM are $\boldsymbol{w} = \begin{bmatrix} 5 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$. The constraints

in Eq. (5.33) can be expressed in the form of Eq. (5.34):

$$\boldsymbol{G}\boldsymbol{v}_{\boldsymbol{k}} \leq \boldsymbol{F}\boldsymbol{\theta}_{\boldsymbol{k}}(\boldsymbol{x}_{\boldsymbol{k}}) + \boldsymbol{z} \tag{5.34a}$$

$$\boldsymbol{G} = \begin{bmatrix} -\boldsymbol{S_{oxy}} & & & \\ \boldsymbol{S_{ace}} & & & \\ -\Delta t\boldsymbol{S_{glc}} & & & \\ -\Delta t\boldsymbol{S_{oxy}} & & & \\ -\Delta t\boldsymbol{S_{ace}} & & & \\ -\boldsymbol{S_{glc}} & & & \\ -\Delta t\boldsymbol{S_{bio}} & & & \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad \boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.34b}$$

$$\boldsymbol{z} = \begin{bmatrix} OUR_{max} & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \end{bmatrix}^T \tag{5.34c}$$

5.4.2 Multiparametric Programming for *E.coli* Model

Monte Carlo simulations are conducted for different feeding policies and initial state from set $\mathcal{X}_0$ to determine the bounds of the parameter space $\boldsymbol{\Theta}$ that should be considered for multiparametric programming. The parameter space $\boldsymbol{\Theta}$ corresponding to this DFBM is $\boldsymbol{\Theta} = \mathcal{R}(\begin{bmatrix} 0 & 0 & 0 & 0 & 0.9 \end{bmatrix}^T, \begin{bmatrix} 500 & 250 & 250 & 6.5 & 1.005 \end{bmatrix}^T)$.

By treating both $\boldsymbol{\theta}$ and $\boldsymbol{v}$ as decision variables of Eq. (5.3), an initial feasible $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}$ is obtained as $\boldsymbol{\theta} = \begin{bmatrix} 491.24 & 241.24 & 241.24 & 6.41 & 0.919 \end{bmatrix}^T$. This $\boldsymbol{\Theta}$ is used as a starting point for the exploration of all critical regions. Then, Problem P_w is solved for the $\boldsymbol{\theta}$ found to determine active constraints. Using 10^{-6} as tolerance, if the *i-th* slack variable $z_i \leq 10^{-6}$ in P_w problem, the constraint is treated as active. Only the constraint given by Eq. (5.33m) is found active. For this DFBM, $n_r = 4$ and thus at least 4 constraints need to be active to result in a unique solution. If fewer than 4 constraints are active, the corresponding critical region will involve multiple solutions. If the only active constraint is Eq. (5.33m), the corresponding critical region is described by Eq.(5.35) and it is denoted by Θ_{12} based on the index of the active constraint in Eq. (5.33).

$$\Theta_{12}: \begin{bmatrix} 0 & -1 & 0 & -0.0331 & 0 \\ -0.7976 & -0.6032 & 0 & 0 & 0 \\ 0 & 0 & -1 & -0.1042 & 0 \\ 0 & -1 & 0 & -0.0584 & 0 \\ 0 & 0 & 0 & -100 & 0 \\ -0.9724 & 0 & -0.2333 & 0 & 0 \\ -0.9192 & -0.3938 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \boldsymbol{\theta}(\boldsymbol{x}) \leq \begin{bmatrix} -0.3179 \\ -0.1917 \\ -0.4929 \\ -0.4375 \\ -235.6 \\ -0.1150 \\ -0.1723 \\ -0.0589 \\ 100.5 \\ 650 \\ 250 \\ 250 \\ 500 \\ -90 \\ 0 \end{bmatrix} \tag{5.35}$$

The critical region Θ_{12} involves 15 constraints. Some of these constraints are from Eq. (5.8) and the rest are related to the boundaries of the parameter space $\boldsymbol{\Theta}$ defined above. The redundant constraints are eliminated by using the MPT3 toolbox. For the polyhedron described by Θ_{12}, each constraint defines the face of this polyhedron. Using the MPT3 toolbox, the Chebyshev center of each face can be easily calculated and a point in the outward direction from this face can always be found. For example, the Chebyshev center point for face defined by the first constraint of Θ_{12} is $\begin{bmatrix} 494.75 & 0.1047 & 244.75 & 6.4467 & 0.9525 \end{bmatrix}^T$. By introducing a small perturbation (10^{-5}) in the normal outward direction, a point can be found as new $\boldsymbol{\theta}$.

Then a P_w problem is solved at the new $\boldsymbol{\theta}$ to determine the active constraints of the new critical region. The corresponding active constraints are found to be $\{4,\ 6,\ 8,\ 10\}$. Since the number of active constraints is 4, these constraints define a critical region with

a unique solution and the region denoted as $\Theta_{\{4,\ 6,\ 8,\ 10\}}$ is given by Eq. (5.36).

$$\Theta_{\{4,\ 6,\ 8,\ 10\}}: \begin{bmatrix} -1.6962 & 0 & 0 & 0.0424 & 0 \\ 0 & 1.5230 & 0 & -0.052 & 0 \\ 0 & 0.7865 & 0 & 0.026 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -100 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix} \boldsymbol{\theta}(\boldsymbol{x}) \leq \begin{bmatrix} 0 \\ 0 \\ 0.25 \\ 0 \\ 0 \\ -90 \\ 500 \\ 250 \\ 650 \\ 100.5 \end{bmatrix} \tag{5.36}$$

Within critical region $\Theta_{\{4,\ 6,\ 8,\ 10\}}$, the optimal solution can be expressed as per Eq. (5.37), which is an affine mapping of $\boldsymbol{\theta}$.

$$\boldsymbol{v} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3.096 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1.523 & 0 & 0.052 & 0 \end{bmatrix} \boldsymbol{\theta}(\boldsymbol{x}) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.37}$$

Following the steps given above, a new $\boldsymbol{\theta}$ can be found for each constraint of the critical region $\Theta_{\{4,\ 6,\ 8,\ 10\}}$. After repeating the process many times and removing duplicated critical regions, all critical regions within the feasible space can be defined and the corresponding optimal solution expressions for each critical region with a unique solution are obtained.

A crucial element of the current work is to address critical regions with multiple solutions. As described in the previous section it is required to divide the critical region with multiplicity into zones and to obtain an approximate solution for each zone. For example, for the critical region Θ_{12} with multiplicity, the region is divided into two polyhedrons $\mathcal{Q}_1$ and $\mathcal{Q}_2$ through the middle point of θ_1 dimension. For Θ_{12}, the minimum and maximum θ_1 are 0.0589 and 500 respectively and thus the middle point of Θ_{12} in the θ_1 dimension is 250.03. Accordingly, constraint $\theta_1 \leq 250.03$ and $\theta_1 \geq 250.03$ are added to the original constraints in Eq. (5.35) to divide the Θ_{12} into two polyhedrons respectively. Using the MPT3 toolbox, the geometric center of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are found as $\mathcal{C}(\mathcal{Q}_1) = \begin{bmatrix} 101.92 & 101.97 & 111.17 & 4.8789 & 0.9525 \end{bmatrix}^T$ and $\mathcal{C}(\mathcal{Q}_2) = \begin{bmatrix} 375.01 & 113.74 & 113.68 & 4.5105 & 0.9525 \end{bmatrix}^T$

respectively. Let denote $\mathcal{C}(\mathcal{Q}_1)$ as $\boldsymbol{\theta_{*1}}$ and $\mathcal{C}(\mathcal{Q}_2)$ be $\boldsymbol{\theta_{*2}}$. Then, problem P_w is solved at $\boldsymbol{\theta_{*1}}$ and $\boldsymbol{\theta_{*2}}$ to obtain $\boldsymbol{v_{*1}} = \begin{bmatrix}0.1073 & 0.1699 & 0.1517 & 0.0711\end{bmatrix}^T$ and $\boldsymbol{v_{*2}} = \begin{bmatrix}0.1243 & 0.1699 & 0.1477 & 0.0581\end{bmatrix}^T$ respectively. Since $\boldsymbol{\theta_*}$ and $\boldsymbol{v_*}$ are known for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively then, problem P'_w given by Eq. (5.14) can be formulated. Instead of solving the resulting NLP, problem P'_{QP} is used to approximate P'_w at $\boldsymbol{\theta_*}$ and $\boldsymbol{v_*}$ for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ respectively. The optimal solution expressions for $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are given in Eq. (5.38), which is also affine mapping of $\boldsymbol{\theta}$.

$$\boldsymbol{v_{QP1}} = \begin{bmatrix}0 & 0 & 0 & -0.0416 & 0\\ 0 & 0 & 0 & 0.0038 & 0\\ 0 & 0 & 0 & 0.0069 & 0\\ 0 & 0 & 0 & 0.0385 & 0\end{bmatrix}\boldsymbol{\theta(x)} + \begin{bmatrix}0.3104\\ 0.1883\\ 0.1179\\ -0.1166\end{bmatrix} \tag{5.38a}$$

$$\boldsymbol{v_{QP2}} = \begin{bmatrix}0 & 0 & 0 & -0.0508 & 0\\ 0 & 0 & 0 & 0.0035 & 0\\ 0 & 0 & 0 & 0.0148 & 0\\ 0 & 0 & 0 & 0.0324 & 0\end{bmatrix}\boldsymbol{\theta(x)} + \begin{bmatrix}0.3534\\ 0.1541\\ 0.0808\\ -0.0883\end{bmatrix} \tag{5.38b}$$

Comparing the solutions of WPDM at vertices of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ with the solutions given by Eq. (5.38) at vertices, the accuracy ϵ_r can be calculated for the case that division is done through the middle point of the first dimension. After repeating the same procedures above, ϵ_r for the division at middle points along other dimensions can also be calculated. Then, the preferable way to divide the polyhedron is the one resulting in the smallest ϵ_r. $\epsilon_{\mathcal{Q}}$ is also calculated to determine whether the current polyhedron needs to be further divided to increase accuracy up to certain tolerance.

In this investigation, 9 critical regions have been found relevant along the culture from Monte Carlo simulations. Only one critical region exhibits multiplicity, which is the critical region Θ_{12} defined in Eq. (5.35). Using the k-d tree partition algorithm, Θ_{12} has been further divided into 127 polyhedrons (zones). Combining 8 critical regions with unique solutions and 127 zones in critical regions with multiplicity, the parameter space is reduced into 135 non-overlapped regions $\{\Theta^i\}$. Each optimal expression for *i-th* region Θ^i is in affine form given as $\boldsymbol{v}^i = \boldsymbol{\mathcal{F}}^i\boldsymbol{\theta} + \boldsymbol{\beta}^i$, for each $\boldsymbol{\theta}^i \in \Theta^i$. Following trimming of the parameter space by the use of Monte Carlo simulations, the entire multiparametric programming approach can be completed within 1.5 hours of CPU time.

To avoid the trimming of the parameter space will leave out some possible $\boldsymbol{\theta}$, a large number of batches (up to 100,000 batches corresponding to 40,100,000 $\boldsymbol{\theta}$ values) were simulated by a Monte Carlo algorithm. Fig. (5.5) plots $\boldsymbol{\theta}$ identified from an increasing

number of batches onto the θ_1 and θ_2 parameter subspace for the *E. coli* model. The colored rectangles are critical regions and zones used by mpNLP. As the figure shows, data of 1000 batches is enough to clearly define the boundaries of possible $\boldsymbol{\theta}$ values, and since conservative rectangular sets are used by mpNLP, all $\boldsymbol{\theta}$ are expected to be contained within the identified boundaries. Also, since the changes in the uncertain parameters are contiguous, the corresponding critical regions and zones should be also contiguous to each other. Hence, the connectivity among these rectangles is checked so that all related critical regions and zones are contiguous.

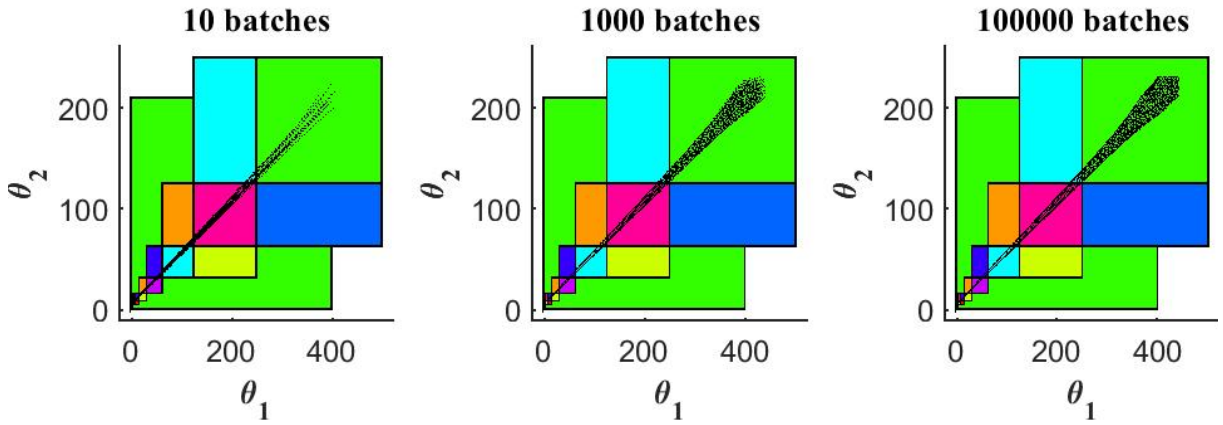


Figure 5.5: Different Number of Batches Simulated by Monte Carlo Algorithm with Parameter Space Projected to θ_1 and θ_2 for the *E.coli* model

5.4.3 Set Membership Estimation

Following the application of multiparametric programming algorithms above, the entire feasible parameter space has been reduced into 135 regions and the optimal solution for each region is an affine mapping of $\boldsymbol{\theta}$. Accordingly, the original DFBM can be converted into a variable structure system. This section shows the propagation of uncertainty from the initial state $\mathcal{X}_0$ into a set of states $\mathcal{X}_k^-$ at different time step k based on limited measurements of biomass and culture volume. Since the state equation is nonlinear, different set operations are applied to find bounds for the states at each time interval. Fig. (5.6) presents the estimate set of state by the method proposed for fed-batch operation. Because the proposed method is based on assumptions that biomass and volume of culture can be measured with bounded noises, only the estimates for the remaining three states,

i.e. glucose (Glc), oxygen (Oxy), and acetate (Ace) concentrations are shown in the figure. The 9 colored boxes in Fig. (5.6) denotes the estimated set of state projected onto glucose-oxygen-acetate subspace at $1.67h$ time interval between $0h$ to $15h$. The black dots denote the true state of the plant at the corresponding time. The estimates clearly contain the true state of the plant well. The length of boxes in different dimensions is a measure of the corresponding uncertainty. The uncertainty of acetate increases sharply during the culture as indicated by the elongation of the boxes in the acetate direction but the uncertainty shrinks towards the end of fermentation.

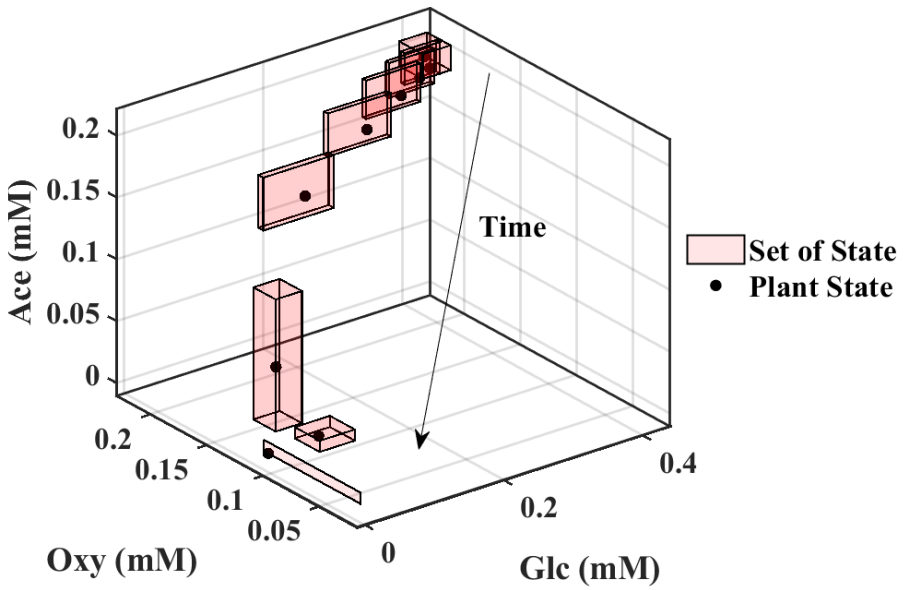


Figure 5.6: Set of state projected onto glucose-oxygen-acetate subspace for fed-batch operation

To verify the calculations by the proposed approach, Monte Carlo Simulation is conducted for comparison. For a fair comparison, a specially designed Monte Carlo Simulation must be used. In a traditional Monte Carlo Simulation of a system with uncertainty in the initial state, different initial states are sampled and propagated separately to obtain the bounds of states at different times. However, since measurements of biomass concentration and volume of culture at different times are considered, many of the trajectories simulated by the traditional Monte Carlo Simulation are outside of the confidence intervals of the biomass and volume measurements and thus they must be removed. In fact, only a small part of the simulated trajectories are compatible with the measurement data. For example, from more than 200000 simulated trajectories only 1000 trajectories were compatible

with the measurements. Of course, the number of acceptable trajectories can be further increased by simulating a larger number of initial states. However, the number of acceptable trajectories (1000) was deemed sufficient to test the method because the results did not change significantly as the number of overall simulations was further increased. The results of fed-batch operation and batch operation have been plotted in Fig. (5.7) and Fig. (5.8) respectively. The grey color areas correspond to the 1000 trajectories sampled from the initial set $\mathcal{X}_0$ compatible with the measurement $\boldsymbol{y}$ at all time steps. The sets of states are projected onto each dimension, i.e. each metabolite's concentrations, to obtain upper and lower bounds. Clearly, the bounds obtained with the proposed method contain all the trajectories compatible with the measurements without any bound violation for both cases. It can be noticed that the simulations result in a narrower grey area for batch operation (Fig. (5.8)) as compared to fed-batch (Fig. (5.7)). The difference in the Monte Carlo-based bound between batch and fed-batch is that the uncertainty at the end of the batch operation is narrower than for the end of the fed-batch operation. Thus, more trajectories must be removed in batch as compared to fed-batch because they are incompatible with the measurements. While Monte Carlo provides tighter bounds it is very time-consuming requiring one day for the required 100,000 runs as compared to 8 min with the proposed SME approach.

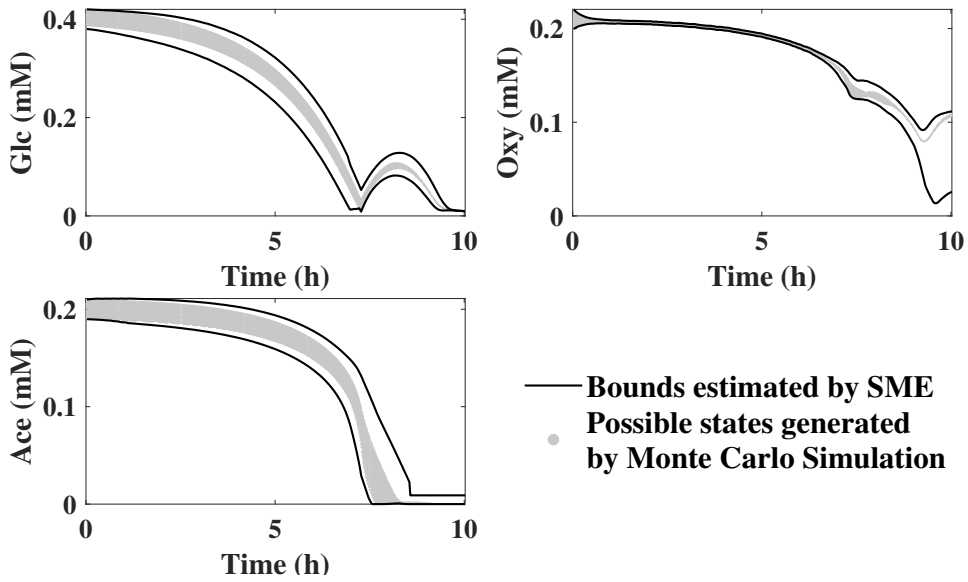


Figure 5.7: Comparison bounds estimated by SME with Monte Carlo Simulation for fed-batch operation

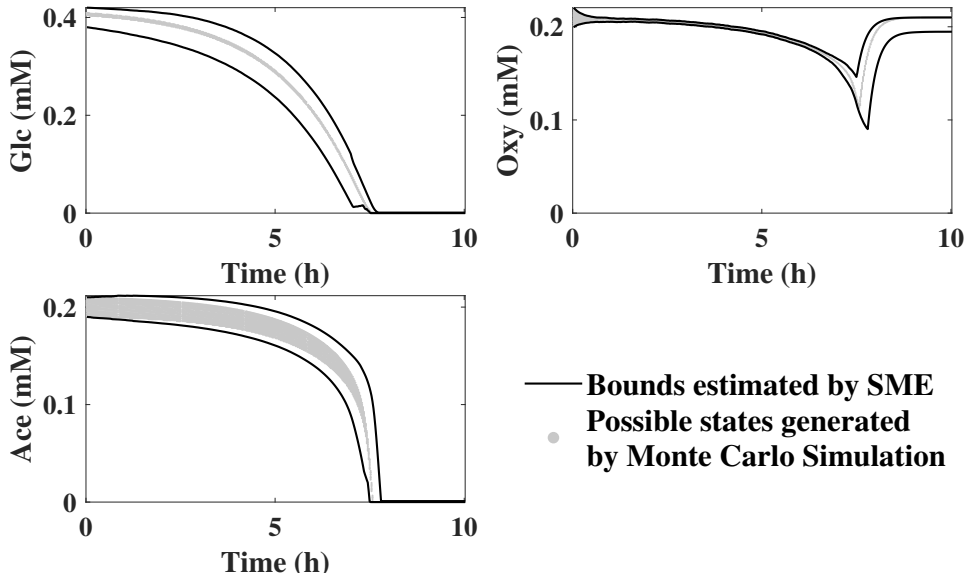


Figure 5.8: Comparison bounds estimated by SME with Monte Carlo Simulation for batch operation

5.4.4 Discussion

Applying an SME-based approach to DFBMs is challenging because of the need to solve an inner optimization. While the inner optimization is an LP problem, multiple solutions are prevalent in DFBM models. To apply SME, the multiplicity problem must be addressed. WPDM is a type of interior-point method with extra parameter interior-point weights introduced to solve the multiplicity issue. If the optimal solution of LP is unique, both WPDM and LP solvers can be used to obtain the unique solution. However, if the optimal solution of LP is not unique, WPDM can be implemented to obtain different optimal solutions. Interior-point weights can be found by fitting experimental data, which makes the method data-driven and especially suitable for addressing different engineered strains.

To bypass the need of solving the inner optimization, multiparametric programming is introduced. For a critical region with a unique solution, there is no need to apply WPDM and mpLP can be used to obtain the corresponding optimizer expression. For critical regions with multiple solutions, WPDM and mpNLP need to be combined since the former converts the problem into an NLP. To tackle the NLP problem, the critical regions with multiplicity are divided into different zones by mpNLP. For each zone, local quadratic programming is used to approximate a special P'_w optimization formulation of WPDM that addresses the singularity of the Hessian matrix.

Different metabolic reactions within a metabolic network exhibit different yields with respect to biomass growth. The constraints of the DFBM model describe limitations in terms of their availability and kinetic rate limitations. During different cell culture periods some constraints are activated indicating the cell's ability to take advantage of reactions with higher yields. Once some resources are consumed, the cell will switch to the consumption of other resources that have not been depleted. Different critical regions correspond to different abundance of resources and different abundance of resources determine different metabolism patterns.

The critical region with multiplicity corresponds to situations where cells have different ways to distribute their resources or due to constraints that are needed because of insufficient knowledge about the metabolism, i.e. unknown limited substrates. The WPDM method can be used to address multiplicity.

Once multiparametric programming is applied to the DFBMs, the explicit optimal solution expression for each region can be used to simplify the inner optimization within the state equation by converting the original problem into a variable structure system. The set of states is broken into pieces for each region. Since the new state equations are nonlinear, set propagation cannot be applied directly. Different methods are used to deal with the nonlinearity. For $\boldsymbol{x}$ to $\boldsymbol{\theta}$, first order approximation is used. For scaling of the sets, the outer approximation is used. For $\boldsymbol{B}(x_{v,k}, q_k)$ and $\boldsymbol{h}(x_{v,k}, q_k)$, the nominal value of each set for each region is used. By applying these set operations and approximations, the set of states at the new time interval can be estimated.

The bounds found by SME proposed are not strictly guaranteed. Different approximations are used in this research, including the first order approximation from $\boldsymbol{x}$ to $\boldsymbol{\theta}$, approximation of P_w by P'_w in multiparametric nonlinear programming, and approximation $\boldsymbol{B}(x_{v,k}, q_k)$ and $\boldsymbol{h}(x_{v,k}, q_k)$ by $\boldsymbol{B}(\hat{x}^i_{v,k}, q_k)$ and $\boldsymbol{h}(\hat{x}^i_{v,k}, q_k)$ respectively. However, despite these approximations, exhaustive Monte Carlo simulations for different noise and feeding policies have shown that the estimated bounds are not violated. A key reason for the lack of violations is that the outer SME approximation overestimates the size of the set thus compensating for the approximations.

5.5 Conclusions

The lack of measurements poses a challenge for the design of online estimators based on DFBM models. An online estimator of bounds on states based on limited measurements is proposed. Since multiplicity is a common occurrence in DFBM models, an interior point

method (WPDM) is used to calculate a unique solution. However, the resulting optimization becomes an NLP instead of the original LP formulation. This NLP formulation requires the use of both linear and nonlinear multiparametric programming to convert the DFBM into a variable parameter system. This system is described by state equations for each critical region defined in the uncertain parameter space. Then, a set membership estimation is applied to the variable parameter system to propagate uncertainty in initial conditions and to obtain bounds of unmeasured states. The bounds provided by the method are compared and verified with extensive Monte Carlo simulations for batch and fed-batch fermentation of *E.coli*. The proposed method is several orders of magnitude faster than the Monte Carlo-based calculations of the bounds.

Chapter 6

A Method for Tackling Multiplicity in Dynamic Flux Balance Models by an Ellipsoidal Reflection Operation

6.1 Overview

[1]Dynamic Flux Balance Models (DFBM) can describe the evolution of metabolites' concentrations with time by solving a linear programming (LP) problem at each time interval. However, since multiple solutions of the LP commonly exist, different trajectories can be obtained. An interior-point-based method (weighted primal-dual method), proposed in a previous study that picks a specific solution from multiple solutions of LP is computationally expensive for a large metabolic network. In this research, we propose an alternative computationally efficient method based on an ellipsoidal reflection mathematical operation. The method is applied to a test example and then to model *B. pertussis* fermentation resulting in excellent computational performance and accurate fitting of experimental dynamic data.

[1]Adapted from a paper submitted to Computers & Chemical Engineering, which is currently under review.

6.2 Introduction

The need to optimize bioprocesses has motivated the development of accurate models for process investigation and model-based optimization. Classic biochemical models used in earlier biotechnological research are generally macroscopic unstructured models of biomass, product, and substrates. These unstructured models ignore the specific interconnections of the metabolic network of the microorganisms. Instead, these unstructured models oversimplify the structure of metabolic networks thus preventing useful interpretation of the biochemical system [94].

Genome-based dynamic flux balance models (DFBMs) have gained increasing attention for modeling biochemical processes [64, 60, 36] since they formally consider key metabolic interconnections and can capture the detailed correlations existent between different metabolites. A DFBM is formulated as a linear programming (LP) problem based on the assumption that cells allocate available nutrients in metabolic networks to maximize a biological objective such as growth rate or others. From the solution of the LP the fluxes, i.e. the metabolic reaction rates, can be calculated and substituted into mass balances to evolve the concentrations with time. However, being the LP problem generally underdetermined due to a lack of sufficient constraints, the solution may not be unique. This results in a drastically different dynamic evolution of metabolites' concentrations based on the optimal solutions that are selected [98, 99]. This multiplicity of solutions is a critical problem especially for DFBMs because metabolites' concentrations evolve according to the time integration of the vector of fluxes multiplied by the biomass. If the solution of the fluxes chosen at different time steps is in error with respect to the data, these errors will accumulate over time thus potentially resulting in concentration profiles that will diverge significantly from the data.

Shen and Budman [98], have recently proposed the weighted primal-dual method (WPDM) [98] to address multiplicity that was found more accurate than previous techniques specifically for matching experimental data for engineered strains. WPDM is an extension of the interior-point method of LP that is based on the use of adjustable parameters (interior-point weights) in the barrier terms of the objective function. This method converts the LP into a nonlinear programming problem (NLP) to determine which solution is selected based on the chosen weights. Because the objective function of WPDM is strictly convex, the solution is unique. When multiple solutions of the LP are possible, WPDM can obtain all these solutions by varying the weights. In DFBMs, parameters interior-point weights can be obtained by fitting experimental data which makes the method data-driven and widely applicable. Because of the ability to adjust the weights based on data, WPDM was also shown to provide good fitting as compared to other approaches mentioned above [98].

On the other hand, WPDM has several limitations that hinder its application to large metabolic networks with many constraints and metabolites as follows: i-the computation expense is very high because the number of weights increases with the number of constraints thus increasing computations for fitting with experimental data, ii- commercial solvers are not available thus requiring the development of in-house software that is very slow and iii- since the logarithmic barrier function in the objective is almost infinity at the boundary, a matrix of large condition number results and the need to invert this matrix leads to numerical issues.

To overcome these drawbacks of WPDM, we propose a novel method, the ellipsoidal reflection method (ERM), since it relies on an ellipsoidal reflection operation. As in WPDM, ERM can select a specific unique solution from multiple solutions by tuning of certain parameters. On the other hand, the proposed method has several new advantages as compared to WPDM as follows: i-ERM can take advantage of fast and accurate commercial LP and QP solvers, ii-it needs fewer parameters to tune compared with WPDM because the number of parameters is not correlated to the number of constraints in the problem, iii- ERM involves a very simple 100-line code thus resulting in efficient computation, iv- the algorithmic structure and geometric meaning of ERM are far easier to understand than WPDM, and v- it is simple to initialize fitting parameters of ERM when ERM is used in the fitting with experimental data. The method is applied in the current work to a toy example and then to the DFBM of *B. pertussis* resulting in an impressive performance as compared to the previously reported WPDM.

The paper is organized as follows. Section 2 presents the background on the DFBM. Section 3 explains the multiplicity problem in DFBM models and briefly describes WPDM which is compared later to the new ERM method. Section 4 introduces the ERM and its properties. Section 5 shows the comparison between WPDM and ERM for the toy example and the B. pertussis dynamic flux model. Section 6 presents a summary and conclusions of this research.

6.3 Dynamic Flux Balance Model

DFBM combines discrete mass balance equations for relevant metabolites with a static optimization problem that solves fluxes among metabolites. The discrete mass balances as a function of the metabolic fluxes $\boldsymbol{v_k}$ are defined in Eq. (6.1).

$$\boldsymbol{\psi_{k+1}} = \boldsymbol{\psi_k} + \Delta t \psi_{bio,k} \boldsymbol{S} \boldsymbol{v_k} \tag{6.1}$$

where subscript k indicates time step from 0, 1, $2\cdots$, Δt is the time step size and $\boldsymbol{\psi_k}$ is a state vector of n_ψ state variables at time step k that includes the biomass concentration $\psi_{bio,k}$. Also, $\boldsymbol{S} \in \mathbb{R}^{n_\psi} \times \mathbb{R}^{n_r}$ is a matrix containing stoichiometric coefficients of all reactions involved in the metabolic network, where n_r is the number of reactions considered in the metabolic network.

The metabolic flux vector $\boldsymbol{v_k} \in \mathbb{R}^{n_r}$ is determined by a linear programming (LP) problem according to Eq. (6.2). At each time step, $\boldsymbol{v_k}$ is solved by the LP solver and substituted into Eq. (6.1) to obtain the state vector at the next time step by Euler-based integration as per Eq. (6.1).

$$\min_{\boldsymbol{v_k}} \quad \boldsymbol{f}^T \boldsymbol{v_k} \tag{6.2a}$$

$$\text{subject to} \quad \boldsymbol{G v_k} \leq \boldsymbol{g}(\boldsymbol{\psi_k}, \boldsymbol{\beta}) \tag{6.2b}$$

$$\boldsymbol{F v_k} = \boldsymbol{h}(\boldsymbol{\psi_k}, \boldsymbol{\beta}) \tag{6.2c}$$

where the constant vector $\boldsymbol{f} \in \mathbb{R}^{n_r}$, the constant matrix $\boldsymbol{G} \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_r}$, the constant matrix $\boldsymbol{F} \in \mathbb{R}^{n_F} \times \mathbb{R}^{n_r}$, vector-valued function $\boldsymbol{g} \in \mathbb{R}^{n_G}$ of states $\boldsymbol{\psi_k}$, vector-valued function $\boldsymbol{h} \in \mathbb{R}^{n_F}$ of states $\boldsymbol{\psi_k}$, parameters $\boldsymbol{\beta}$ controlling the constaints. n_G is the number of inequality constraints and n_F is the number of equality constraints. Eq. (6.2a) denotes the objective function of the LP. The most commonly used objective is the maximization of biomass growth rate or equivalently minimization of its negative value. Eqs. (6.2b) and (6.2c) describe balance equations and metabolic constraints such as charge balance, reaction rate bounds, and available nutrient bounds. The problem in Eq. (6.2) is based on the assumption that the cells are able to regulate the metabolic fluxes $\boldsymbol{v_k}$ to boost growth at each time step.

6.4 Linear Programming and Multiplicity of solutions

6.4.1 Linear Programming

Eq. (6.2) describes a typical LP problem and the general form of the LP problem is defined as Eq. (6.3). For comparison and simplicity, Eq. (6.3) is referred to as P problem in this

paper.

$$
\begin{aligned}
& & \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} && \text{(6.3a)}\\
P & & \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} && \text{(6.3b)}\\
& & & \boldsymbol{A}_{\boldsymbol{e}}\boldsymbol{x} = \boldsymbol{b}_{\boldsymbol{e}} && \text{(6.3c)}
\end{aligned}
$$

6.4.2 Multiplicity Issue

Since the paper focuses on the multiplicity problem, we assume without loss of generality that the LP problems discussed in this paper are feasible and bounded problems. The constraints of Eqs. (6.3b) and (6.3c) define a convex polyhedron where the gradient of objective Eq. (6.3a) is in the $\boldsymbol{c}$ direction. Along the negative gradient direction $-\boldsymbol{c}$, the objective is decreasing until it reaches the boundary of the polyhedron. Thus, the solution of LP is unique at a vertex point, or all points onto an entire face of the polyhedron are solutions. When the solution is not unique, the solutions on the entire face of the polyhedron are either at vertexes or at interior points, also referred to as basic and non-basic optima respectively. However, our goal is to identify from all the possible solutions (including all basic or non-basic optima) which solution results in the best fit to the dynamic experimental data. In our previous research [98], we found that if the optimal solutions are arbitrarily picked at different time intervals out of all possible solutions, the resulting trajectories of states $\boldsymbol{\psi}$ may largely diverge from the data. Hence, if the multiplicity problem is not addressed, the model loses predictability.

Although different solvers have been proposed to solve LP problems, they are commonly all variants of the Simplex method and interior-point method. For a minimization problem, the simplex method starts from a vertex of the polyhedron and pivots along the edges of the polyhedron to another vertex in the direction which has the smallest angle with the $-\boldsymbol{c}$ direction. After a finite number of iterations, the algorithm reaches a vertex where all edges have angles that are smaller or equal to 90° with respect to the $\boldsymbol{c}$ direction. This vertex corresponds to the optimal solution to the LP problem since it has the smallest objective value. Because the Simplex method only searches vertexes, it can only obtain basic solutions even though other solutions may exist as shown in Fig. (6.1). The dash lines are contours of the objective function. The red line sections are the optimal face of the polyhedron, where any point is an optimal solution to the LP. The star represents different solutions obtained by either the Simplex or interior point algorithms. The blue polyhedron represents the feasible space, the boundaries of which are defined by constraints.

The interior-point method (IPM) starts the search for an optimal solution from an

interior point within the polyhedron. At each iteration, the shape of the convex surface gradually becomes an hyperplane because of the linearity of the objective. Then, the minimum point of this surface gradually approaches the optimal solution along the search path. Since the IPM performs a search within the polyhedron, the resulting solution is usually an interior point of the optimal face (non-basic solutions) even though the solutions of the LP are not unique as Fig. (6.1) shows. CPLEX even includes a post-processing step to convert the interior point solution to a vertex solution when solutions are not unique [48]. It should be noticed that IPM can still provide solutions at the vertexes when multiple solutions exist, even for solvers that do not apply any post-processing step. The specific challenge tackled in the current study is to select a specific solution among all possible solutions that will satisfy an additional objective such as fitting a given set of experimental dynamic data. However, neither Simplex nor IPM is tailored to explicitly address this multiplicity-related challenge.

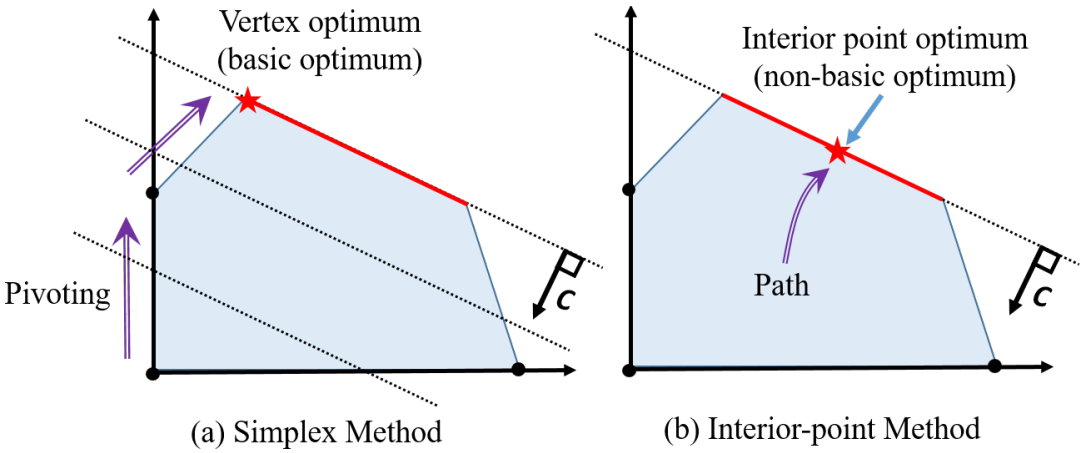


Figure 6.1: Schematic of Simplex and Interior-point methods for problems with multiple solutions

6.4.3 Weighted Primal-Dual Method

To deal with the multiplicity issue, a data-driven solver referred to in a previous study as the weighted primal-dual method (WPDM) was utilized. For completeness, a brief description of WPDM is provided here. WPDM is defined in Eq. (6.4) and it is denoted as the P_w form of the LP in Eq. (6.3). WPDM is a variant of the interior-point method, which selects a particular optimal solution based on the choice of interior-point weights

($\boldsymbol{w}$). When all weights are 1, WPDM becomes the traditional interior-point method. The weights determine the curvature and the location of the desired optimum on the face that contains the multiple optima. If the solution is unique, WPDM still can obtain the same solution as the one obtained from the P optimization problem. The numerical method for WPDM is based on a path-following approach, details of which are reported in a previous study [98].

$$
\begin{aligned}
P_w \qquad & \inf_{\tau \to 0, \boldsymbol{x}, \boldsymbol{z}} && \boldsymbol{c}^T \boldsymbol{x} - \tau \sum_{i=1}^{n_{\tilde{A}}} w_i ln(z_i) && \text{(6.4a)} \\
& \text{subject to} && \tilde{\boldsymbol{A}} \boldsymbol{x} + \boldsymbol{z} = \tilde{\boldsymbol{b}} && \text{(6.4b)} \\
& && \boldsymbol{z} > \boldsymbol{0} && \text{(6.4c)}
\end{aligned}
$$

where τ is an infinitesimal constant, $\boldsymbol{w}$ is an interior-point weight vector and w_i is the *i-th* interior-point weight. $\boldsymbol{z}$ is a vector of slack variables that are added to convert the inequality Eq. (6.3b) to equality. The *i-th* interior-point weight w_i corresponds to the slack variable z_i and *i-th* constraint. In particular, each equality constraint in Eq. (6.3c) needs to be split into two inequality constraints and two additional slack variables. For instance, $2x_1 + x_3 = 2$ needs to be split into $2x_1 + x_3 \leq 2$ and $-2x_1 - x_3 \leq -2$ first and compensated with slack variables according to $2x_1 + x_3 + z_1 = 2$ and $-2x_1 - x_3 + z_2 = -2$. Therefore, the new constraints become $\tilde{\boldsymbol{A}} \boldsymbol{x} + \boldsymbol{z} = \tilde{\boldsymbol{b}}$ and the total number of constraints is $n_{\tilde{A}}$.

The suitability of WPDM for addressing the multiplicity of DFBMs is based on the three following properties: i- WPDM in Eq. (6.4) can approximate the original LP in Eq. (6.3) as τ tends to zero [98]. ii- WPDM provides a unique optimal solution for given weights [98]. iii- By manipulating the interior-point weights, a specific optimum can be selected if multiple optima coexist according to the P problem [98]. Fitting experimental data is possible by tuning the set of weights' values. To the knowledge of the authors, WPDM is the only data-driven method to select a solution for the LP.

However, WPDM has drawbacks preventing its application in large metabolic networks as follows: i- the computation expense is proportional to the number of constraints. For example, for a metabolic network with 200 reactions and 2000 constraints, while the number of decision variables is only 200, the vector of weights $\boldsymbol{w}$ has 2000 elements. Hence, for fitting with experimental data, the number of tuning weights is 2000 which is computationally expensive. ii- commercial solvers cannot be applied directly to such problems requiring the use of an in-house code that is not optimized for computational efficiency. iii- WPDM inherits the property of the original interior-point method that the logarithmic barrier function in the objective tends to infinity at the boundary thus leading to numerical

problems in the inversion of the resulting almost singular matrix when searching for the optimal direction. iv- the trial and error for initialization of tuning weights are necessary and time-consuming.

6.5 Ellipsoidal Reflection Method

The ellipsoidal reflection method (ERM) maintains the advantages of WPDM as compared to other methods but it improves over WPDM in several aspects as follows: i- fast and accurate commercial solvers are used with ERM, namely commercial LP and QP solvers, ii- as fewer tuning parameters are required as compared with WPDM, the computation efficiency is significantly better than the latter, iii- ERM is based on a much simpler code as compared to WPDM leading to easier geometric interpretation of the result, and iv- initial tuning parameters of ERM are simple to find.

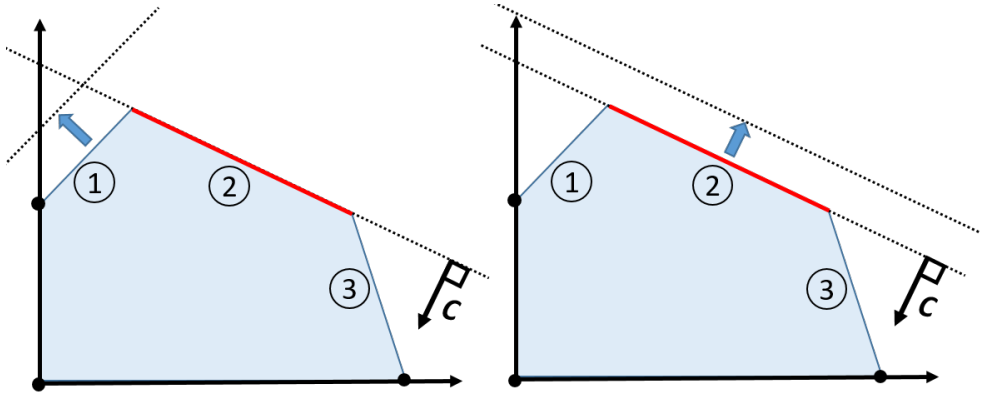
The ellipsoidal reflection method is inspired by the "rotating compass" concept. The core idea of the algorithm is to change a "pointer" vector direction to point towards different solutions. By changing the direction of a quadratic objective surface corresponding to the "pointer" vector direction, a unique solution on the optimal face that best fits the data is selected. The algorithm involves three main steps: i- identification of the optimal face, ii- translation of the optimal face, and iii- selection of the solution that points to a given direction.

6.5.1 Identification of the Optimal Face

To describe the method, some definitions are provided which are needed to describe the multiplicity issue and the method. A constraint is referred to as "binding" if equality holds. For a given solution, a binding constraint is active if the corresponding dual variable ("shadow price") is greater than 0, namely the complementary slackness condition. It should be noticed that the inference cannot be made in the reverse direction [74], i.e. the dual variable of a binding constraint can be 0 and the dual variable of an equality constraint can also be 0. The geometric explanation is presented in Fig. (6.2). For a minimization problem, the objective of the LP decreases along the inverse direction of $\boldsymbol{c}$ where the red line represents the optimal face where all points along this line are optimal solutions. As the figure shows, the relaxation of constraint 1 or constraint 3 cannot minimize the objective because it is not in the $-\boldsymbol{c}$ direction. On the other hand, the relaxation of constraint 2, which is parallel to the optimal face, can reduce the value of the objective.

Dual variables measure the marginal effect or sensitivity of the optimal value of the primal objective function with respect to an infinitesimal change on the right-hand side of the primal constraints. For example, if the optimal objective is p_* and the dual variable of *i-th* constraint is λ_i then $\lambda_i = \frac{\partial p_*}{\partial b_i}$. Since the relaxation of constraint 1 or constraint 3 cannot minimize the objective according to Fig. (6.2), the corresponding dual variables are zero. On the other hand, the dual variable of constraint 2 is greater than zero.

When the Simplex method is used for the problem described in Fig. (6.2), it may calculate either the left corner vertex defined by constraints 1 and 2, or the right corner vertex defined by constraints 2 and 3. For example, for the left corner vertex, constraint 1 is binding but the corresponding dual variable is zero. If IPM is applied and a non-vertex optimum is obtained, the only binding constraint is 2. If IPM is applied and the left corner vertex is obtained, the binding constraints are 1 and 2. The dual variable of constraint 2 is greater than zero but the dual variable of constraint 1 is a very small positive number (weakly-activated constraint) because most commercial solvers use a non-zero tolerance and IPM can only approximate the optimal solution from the interior domain but it never reaches the boundary exactly.



(a) Relaxing constraint 1 (b) Relaxing constraint 2

Figure 6.2: Relaxation of different constraints when the solution of the LP is not unique

Without loss of generality, we assume the solution of LP discussed in this paper is non-empty and bounded; and binding constraints are linearly independent. The main application of the property of active constraints discussed above is to determine whether a given LP has a unique solution or not. For an LP problem that has n decision variables, if the number of active constraints is less than n, the solution is not unique and the dimension of the optimal face containing the multiple solutions is equal to the number of degrees of freedom, i.e. number of decision variables minus the number of active constraints. Let's

define the set of indices of active constraints of inequality constraints as $\mathcal{A}$, the inactive inequality constraints are $\mathcal{N}$ and the optimal face Θ as per Eq. (6.5). The first step of ERM is to solve the LP and determine whether it has a unique solution. If the solutions are not unique, ERM calculates the optimal face Θ. It should be noticed that ERM is only applicable when the dimensionality of Θ is less than n.

$$\Theta = \left\{ \boldsymbol{x} \mid \boldsymbol{A}_{\mathcal{A}}\boldsymbol{x} = \boldsymbol{b}_{\mathcal{A}}, \boldsymbol{A}_{\mathcal{N}}\boldsymbol{x} \leq \boldsymbol{b}_{\mathcal{N}}, \boldsymbol{A}_{\boldsymbol{e}}\boldsymbol{x} = \boldsymbol{b}_{\boldsymbol{e}}, \boldsymbol{c}^T\boldsymbol{x} = p_* \right\} \tag{6.5}$$

For simplicity, Θ can be simplified as Eq. (6.6).

$$\Theta = \left\{ \boldsymbol{x} \mid \hat{\boldsymbol{A}}\boldsymbol{x} = \hat{\boldsymbol{b}}, \boldsymbol{A}_{\mathcal{N}}\boldsymbol{x} \leq \boldsymbol{b}_{\mathcal{N}} \right\} \tag{6.6a}$$

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{A}} \\ \boldsymbol{A}_{\boldsymbol{e}} \\ \boldsymbol{c}^T \end{bmatrix} \tag{6.6b}$$

$$\hat{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b}_{\mathcal{A}} \\ \boldsymbol{b}_{\boldsymbol{e}} \\ p_* \end{bmatrix} \tag{6.6c}$$

6.5.2 Selecting a particular solution

Selecting a particular solution of a QP problem

For clarity, the third step of the algorithm is described before the second step. The third step of ERM consists of selecting a particular solution from the optimal face Θ.

Inspired by the "rotating compass" concept, a pointer is used to select the solution. In ERM, the pointer is along the major axes of concentric hyper-ellipsoids, which are the contour lines of a quadratic function. By "rotating" these concentric hyper-ellipsoids together, the major axes can point towards different solutions on the optimal face Θ. The idea is shown schematically in Fig. (6.3). The optimal face Θ containing all the possible solutions is a convex polyhedron, denoted by the red line in Fig. (6.3). The green dash lines are "rotating" concentric ellipsoids corresponding to the contour lines of a quadratic function to defined below. The semi-major axes of the concentric ellipsoids are in the $\boldsymbol{r_1}$ direction. The direction $\boldsymbol{r_1}$ determines the solution, represented by the stars in Fig. (6.3), chosen among all possible solutions on Θ. Direction $\boldsymbol{r_1}$ are parameters to be tuned based on experimental data.

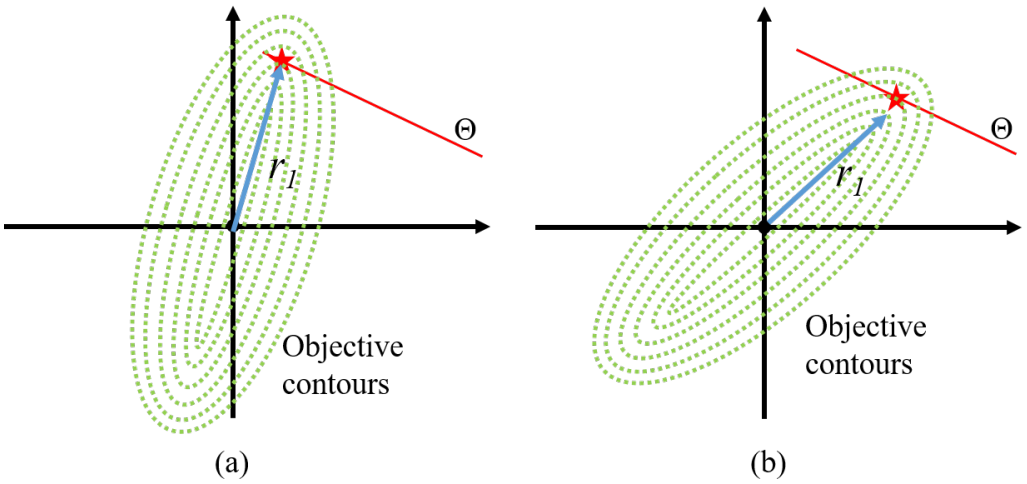


Figure 6.3: Selecting the unique solution by a given direction

The procedure described above is formulated as a QP problem by Eq. (6.7).

$$P_{QP} \quad \min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}(\boldsymbol{r_1})\boldsymbol{x} \tag{6.7a}$$

$$\text{subject to} \quad \boldsymbol{x} \in \Theta \tag{6.7b}$$

where $\boldsymbol{Q}$ is a positive definite matrix which is a function of the "rotation" direction $\boldsymbol{r_1}$. Different matrix $\boldsymbol{Q}$ will result in different "rotation" directions of the objective. Additional details about the construction of matrix $\boldsymbol{Q}$ are given in the following section.

An alternative to the QP for choosing a solution could be to elongate the $\boldsymbol{r_1}$ to intersect Θ to obtain different solutions. However, if $\boldsymbol{r_1}$ is in the wrong direction (see Fig. (6.4)), the elongation of $\boldsymbol{r_1}$ may not intersect Θ and no solution is obtained. It is numerically difficult to define the range of $\boldsymbol{r_1}$ so that the intersection of the elongation $\boldsymbol{r_1}$ with Θ is not empty. The use of the QP serves to avoid this problem. As shown in Fig. (6.4), although the semi-major axes do not point towards the optimal face Θ (denoted by the red line section), a solution (denoted by the star) can still be found. Mathematically, the existence of the unique solution is assured by the strict convex objective constructed in Eq. (6.7a), once the Θ is feasible and bounded. When the semi-major axes do not point towards the optimal face Θ, the semi-major axes are no longer the pointer, and the solution obtained is determined by the tangent point on the elliptical contours. Once the "rotation" direction $\boldsymbol{r_1}$ varies, the tangent point also changes and a different solution is obtained. In other words, ERM still works well in this special case and can provide different solutions by

varying the direction of $\boldsymbol{r_1}$. Ideally, it is preferable to have $\boldsymbol{r_1}$ in the direction of the semi-major axes because of the resulting higher sensitivity to changes in the selected direction as compared to the other axes.

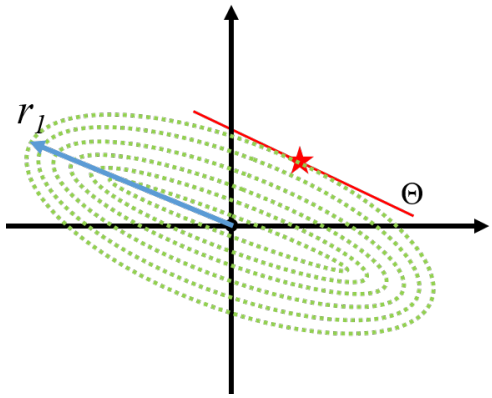


Figure 6.4: A special case for ERM that the semi-major axes do not point towards the optimal face Θ

Householder Transformation and Reflection Matrix

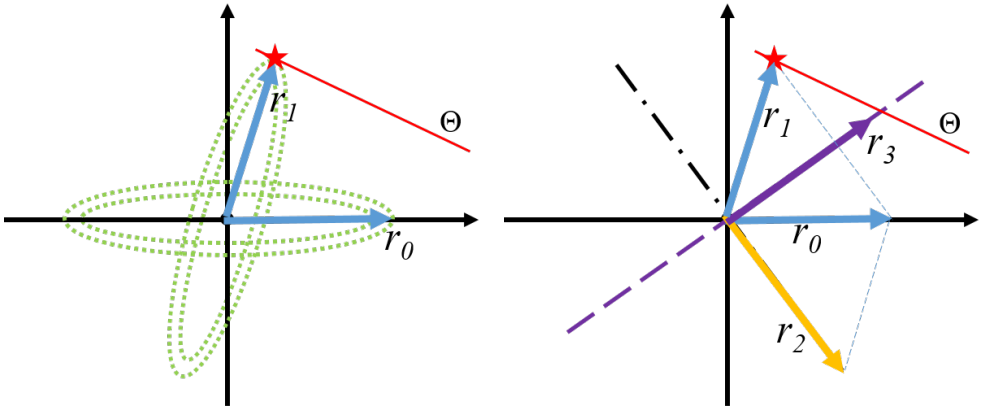
This section provides details about the construction of the matrix $\boldsymbol{Q}$ and the "rotation" direction $\boldsymbol{r_1}$. Without loss of generality, assume initially the longest semi-major axes of concentric hyper-ellipsoids are along the first coordinate and the other shorter semi-minor axes are along the remaining coordinates. Thus, the initial pointer direction is $\boldsymbol{r_0} \in \mathbb{R}^n = [1\ 0\ \cdots\ 0]^T$. Let's assume that the ideal solution is in the direction of $\boldsymbol{r_1}$ and the semi-major axes of these concentric hyper-ellipsoids are "rotated" from $\boldsymbol{r_0}$ to $\boldsymbol{r_1}$ as shown in Fig. (6.5a) to select a particular solution.

The construction of concentric hyper-ellipsoids with the longest semi-major axes aligned along the first coordinate with the remaining semi-axes aligned along the remaining coordinates, $Q_0 \in \mathbb{R}^{n\times n}$ is done according to Eq. (6.8).

$$\boldsymbol{Q_0} = \begin{bmatrix} 1e^{-6} & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \tag{6.8}$$

The matrix Q_0 is used to construct a QP as defined in Eq. (6.7). The objective of

the QP is $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q_0}\boldsymbol{x}$. The contours of $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q_0}\boldsymbol{x}$ are concentric hyper-ellipsoids at the origin with the longest semi-major axes aligned along the first coordinate. Hence, Q_0 represents the initial positions of the concentric hyper-ellipsoids before "rotation" and the semi-major axes point initially towards the $\boldsymbol{r_0}$ direction.



(a) Reflecting elliptical contours (b) Householder transformation

Figure 6.5: Reflecting elliptic contours by Householder transformation

Notice Q_0 is a positive definite matrix because all diagonal elements are positive. Positive-definite matrix assures the strict convexity of the QP and thus the solution of the QP is unique if the problem is feasible. Since all elements are on the main diagonal, these elements are also the eigenvalues of the matrix. The eigenvalues determine the lengths of the semi-axes of concentric hyper-ellipsoids and the eigenvectors determine the corresponding directions of semi-axes. The first element, e.g. $1e^{-6}$, in the matrix, determines that the first semi-axes of concentric hyper-ellipsoids are the longest ones because the lengths of semi-axes are inversely proportional to the squared root of the eigenvalues. A small value, e.g. $1e^{-6}$ in the case studies, is selected so as to force that the longest semi-axes is much larger than the length of the other semi-axes resulting in higher sensitivity with respect to rotations.

A reflection matrix $\boldsymbol{R}$ is introduced to express the "rotation" operation of these concentric hyper-ellipsoids. $\boldsymbol{R}$ is based on the Housholder transformation [32], which is used to rotates the initial vector direction $\boldsymbol{r_0}$ to another direction $\boldsymbol{r_1}$ so that $\boldsymbol{r_1} = \boldsymbol{R}\boldsymbol{r_0}$ as shown in Fig. (6.5b). Notice that although the operation is described as a "rotation", mathematically the Householder transformation is a reflection with respect to $\boldsymbol{r_3}$ in Fig. (6.5b). Thus, the contours are reflected from direction $\boldsymbol{r_0}$ to $\boldsymbol{r_1}$ about the hyperplane $\boldsymbol{r_3}$. The

direction $\boldsymbol{r_2}$ is defined as the normal vector to $\boldsymbol{r_3}$. Given direction $\boldsymbol{r_0}$ and direction $\boldsymbol{r_1}$, the reflection matrix $\boldsymbol{R}$ is defined by Eq. (6.9), which is the key matrix of the Householder transformation [32]. Since the initial direction is always fixed at $\boldsymbol{r_0}$, the reflection matrix $\boldsymbol{R}$ is only a function of $\boldsymbol{r_1}$, denoted as $\boldsymbol{R}(\boldsymbol{r_1})$.

$$\boldsymbol{r_2} = \boldsymbol{r_0} - \boldsymbol{r_1} \tag{6.9a}$$

$$\boldsymbol{R} = \boldsymbol{I} - 2\frac{\boldsymbol{r_2}\boldsymbol{r_2}^T}{\boldsymbol{r_2}^T\boldsymbol{r_2}} \tag{6.9b}$$

Before "rotation" the matrix $\boldsymbol{Q}$ is $\boldsymbol{Q_0}$, the matrix $\boldsymbol{Q}$ of the QP problem after "rotation" in Eq. (6.7a) can be defined as per Eq. (6.10).

$$\boldsymbol{Q}(\boldsymbol{r_1}) = \boldsymbol{R}^T(\boldsymbol{r_1})\boldsymbol{Q_0}\boldsymbol{R}(\boldsymbol{r_1}) \tag{6.10}$$

If different $\boldsymbol{r_1}$ are used, the resulting optimal solution is different. Accordingly, when ERM is used for fitting experimental data for a DFBM problem, the elements of $\boldsymbol{r_1}$ are the fitting parameters that determine which optimal solution is chosen among all possible solutions. For simplicity, the vector $\boldsymbol{r_1}$ is assumed as a unit vector as $\|\boldsymbol{r_1}\| = 1$. In other words, vector $\boldsymbol{r_1}$ is the direction vector of the optimal solution.

The geometric description given above can explain a key advantage of the proposed ERM method is superior to the WPDM that requires trial and error for the initial guess of the fitting parameters. To find the optimal solution that fits the data, the direction of $\boldsymbol{r_1}$ must be initialized. For this purpose, any unit vector defined by Eq. (6.11) where $\boldsymbol{x}^o$ is one possible solution of the LP can be used to initialize the vector $\boldsymbol{r_1}$.

$$\boldsymbol{r_1^o} = \frac{\boldsymbol{x}^o}{\|\boldsymbol{x}^o\|} \tag{6.11}$$

The reflection matrix instead of a rotation matrix is used in ERM because the generation of an n-dimensional reflection matrix is computationally simpler than the corresponding rotation matrix. A rotation matrix can be decomposed into the multiplication of two reflection matrices [116].

6.5.3 Translation

The second step of ERM involves a translation of the optimal face of Θ. While in most cases this translation is not needed, in a few special cases it is necessary or at least helpful in the

selection of different solutions. As shown in Fig. (6.6), for the special case that the optimal face is on a hyperplane passing through the origin then the optimal solution obtained is always the same regardless of the "rotation" direction $\boldsymbol{r_1}$. Mathematically, if the $\hat{\boldsymbol{b}} = \boldsymbol{0}$ or some elements of $\hat{\boldsymbol{b}}$ are zeros, the optimal face is either on the hyperplane passing through the origin or very close to it. In these special situations, a small translation of Θ can increase the sensitivity of the of solutions with respect to changes in model parameters. The direction of translation that will most increase sensitivity is logically along the normal to the original direction, i.e. in the $\boldsymbol{c}$ or $-\boldsymbol{c}$ directions as shown in Fig. (6.7). If the Θ is translated by $\boldsymbol{d} = \alpha \boldsymbol{c}$ and $\alpha \neq 0$, the translated Θ is denoted by Θ' and given by Eq. (6.12). In Fig. (6.7), the red dash and red lines denote the Θ and Θ' respectively. The hollow and filled stars represent the solutions obtained before and after the translation respectively.

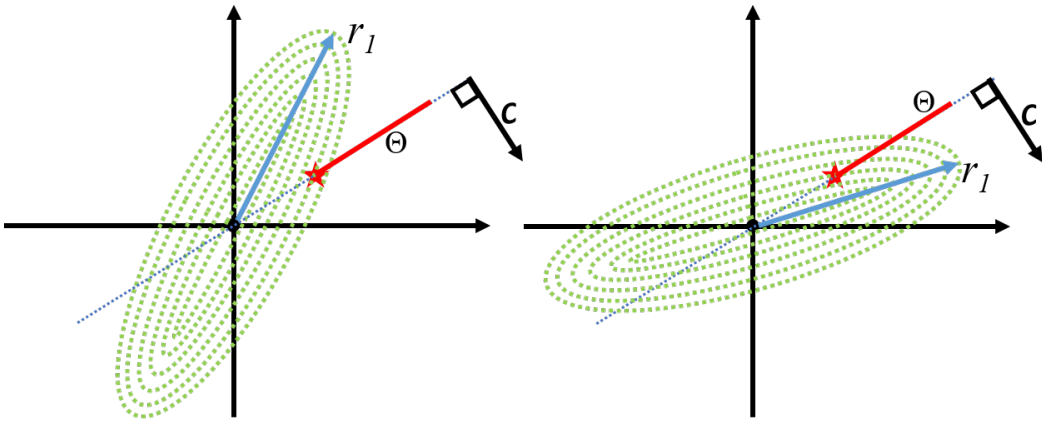


Figure 6.6: When the optimal hyperplane passes through the origin

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{d} \tag{6.12a}$$

$$\Theta' = \left\{ \boldsymbol{y} \mid \hat{\boldsymbol{A}}\boldsymbol{y} = \hat{\boldsymbol{A}}\boldsymbol{d} + \hat{\boldsymbol{b}}, \boldsymbol{A}_{\mathcal{N}}\boldsymbol{y} \leq \boldsymbol{A}_{\mathcal{N}}\boldsymbol{d} + \boldsymbol{b}_{\mathcal{N}} \right\} \tag{6.12b}$$

The QP problem corresponding to the translated optimal face Θ is defined by Eq. (6.13). After obtaining the optimal solution $\boldsymbol{y}^*$ to the translated QP problem in Eq. (6.13), the optimal solution $\boldsymbol{x}^*$ to the original LP can be recovered back from $\boldsymbol{x}^* = \boldsymbol{y}^* - \boldsymbol{d}$. This translation operation preserves all the geometric features of the original optimal face.

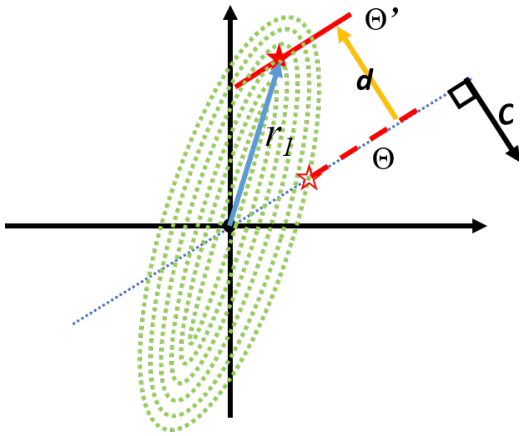


Figure 6.7: Translation of the optimal face

$$\min_{\boldsymbol{y}} \quad \frac{1}{2}\boldsymbol{y}^T\boldsymbol{Q}(\boldsymbol{r_1})\boldsymbol{y} \tag{6.13a}$$

$$P_{QP}{}' \quad \text{subject to} \quad \boldsymbol{y} \in \Theta' \tag{6.13b}$$

6.5.4 Properties of ERM

ERM has some advantageous properties as compared with the interior-point-based WPDM method. A key property is that the number of fitting parameters does not change with the number of constraints as in WPDM. When ERM is used to fit experimental data, the only fitting parameters are the components of the vector $\boldsymbol{r_1}$. Thus, the maximum number of tuning parameters is equal to n which is the dimension of $\boldsymbol{r_1}$ while the number of constraints does not influence the number of fitting parameters. WPDM is based on the constraints and the fitting parameter is $\boldsymbol{w}$. Since each constraint corresponds to an element of $\boldsymbol{w}$, the number of tuning parameters in WPDM increases with the number of constraints thus increasing the computations. Therefore, ERM is computationally more efficient than WPDM in the fitting of experimental data.

Moreover, ERM needs to solve an LP to identify the optimal face Θ and a QP to select the solution in the given direction. Since commercial solvers are available for solving LP and QP problems, the implementation of ERM is very straightforward and robust. WPDM cannot take advantage of commercial solvers and the in-house solver developed for this purpose was found to be inefficient and inaccurate for large metabolic models. Also,

ERM preserves the uniqueness and continuity properties of WPDM as described by the following theorems and remarks.

Theorem 4 (Uniqueness): A feasible QP problem defined in Eq. (6.13) has a unique solution.

Proof: According to the properties of Householder transformation, the matrix $\boldsymbol{R}(\boldsymbol{r_1})$ is invertible [32]. By definition, $\boldsymbol{Q_0}$ is positive definite. Since $\boldsymbol{Q}(\boldsymbol{r_1}) = \boldsymbol{R}^T(\boldsymbol{r_1})\boldsymbol{Q_0}\boldsymbol{R}(\boldsymbol{r_1})$ and matrix $\boldsymbol{R}(\boldsymbol{r_1})$ is invertible, the matrix $\boldsymbol{Q}(\boldsymbol{r_1})$ is positive definite [107]. For a QP problem, if the Hessian matrix $\boldsymbol{Q}(\boldsymbol{r_1})$ is a positive definite matrix, the objective function of the QP problem is strictly convex [8]. For a feasible strictly convex optimization problem, the optimal solution is unique [12].

Remark 4: The contour lines of the objective are initialized as concentric hyper-ellipsoids, which are strictly convex. The Householder transformation of the ellipsoid contour lines preserves convexity. Geometrically, the reflection operation on the ellipsoidal contour lines and the translation of the feasible set cannot change the shape of the contour lines. Since the strict convexity of the objective function is preserved, the feasible QP problem only has one solution. This property ensures the choice of one solution out of the multiple possible solutions of the original LP problem. A solution can be found provided that the feasible space is not empty.

Theorem 5 (Continuity): If the linear independence constraint qualification (LICQ) condition is satisfied, the optimal solution of the feasible QP problem defined in Eq. (6.13) is locally continuous with respect to $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}_{\mathcal{N}}$.

Proof: See appendix.

Remark 5: Under the assumption of LICQ, the optimal solution is a bounded mapping of $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}_{\mathcal{N}}$. The boundedness of the mapping ensures that infinitesimal changes in $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}_{\mathcal{N}}$ result in bounded changes in the optimal solution. Therefore, local continuity is assured.

6.6 Results and Discussions

6.6.1 Example of Simple LP Problem with Multiple Optima

A linear programming problem with multiple optima reported in the literature is used as a preliminary case study [77] to illustrate the ERM. The LP problem is defined in Eq. (6.14). The optimal face is the one with 4 vertex optima, $[4 \quad 0 \quad 2]^T$, $[0 \quad 4 \quad 4]^T$,

$[0 \quad 4 \quad 0]^T$, $[4 \quad 0 \quad 0]^T$. Any point on that face is an optimal solution.

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & -2x_1 - 2x_2 && \text{(6.14a)}\\
\text{subject to} \quad & x_1 + x_2 \leq 4 && \text{(6.14b)}\\
& x_1 + 2x_3 \leq 8 && \text{(6.14c)}\\
& x_1 \geq 0 && \text{(6.14d)}\\
& x_2 \geq 0 && \text{(6.14e)}\\
& x_3 \geq 0 && \text{(6.14f)}
\end{aligned}$$

As a first step, the LP in Eq. (6.14) is solved with IBM CPLEX. The solution obtained is $[0 \quad 4 \quad 0]^T$ and the optimal objective is -8. The dual variables (Lagrange multipliers) for constraints of Eqs. (6.14b)-(6.14f) are 2, 0, 0, 0, 0 respectively. Since the number of decision variables is 3 and only the first constraint Eq. (6.14b) is active, this LP has two degrees of freedom multiplicity. In other words, the optimal face is a two-dimensional face. According to Eq. (6.5), the optimal face Θ is defined by Eq. (6.15).

$$\Theta = \left\{ \boldsymbol{x} \in \mathbb{R}^3 \mid x_1 + x_2 = 4, x_1 + 2x_3 \leq 8, \boldsymbol{x} \geq \boldsymbol{0}, -2x_1 - 2x_2 = -8 \right\} \tag{6.15}$$

Then we proceed to construct a QP to select different solutions from Θ. Since the right-hand side of two equality constraints are 4 and -8 respectively in Eq. (6.15), the second step of translation can be skipped because Θ does not pass close to the origin.

The third step involves the "rotation" operation. For initialization, the major axes of concentric hyper-ellipsoids are aligned along the first coordinate, namely $\boldsymbol{r_0} = [1 \quad 0 \quad 0]^T$. Given an arbitrary "rotation" direction $\boldsymbol{r_1} = [0.626 \quad 0.657 \quad 0.421]^T$, $\boldsymbol{r_2}$ can be calculated as per Eq. (6.9a) to obtain $\boldsymbol{r_2} = [0.374 \quad -0.657 \quad -0.421]^T$. Then, the reflection matrix $\boldsymbol{R}$ can be calculated according to Eq. (6.9b). The resulting $\boldsymbol{R}$ is given in Eq. (6.16). Notice that the reflection matrix $\boldsymbol{R}$ is orthogonal with a -1 determinant indicating that the Householder reflection flips the contour lines from one side to the opposite side. However, since the hyper-ellipsoids are symmetric, such flipping is equivalent to a 180 degrees rotation and it does not influence the results.

$$\boldsymbol{R} = \begin{bmatrix} 0.626 & 0.657 & 0.421 \\ 0.657 & -0.153 & -0.739 \\ 0.421 & -0.739 & 0.527 \end{bmatrix} \tag{6.16}$$

Before the "rotation" is implemented, a matrix $\boldsymbol{Q_0}$ is defined according to Eq. (6.17).

$$\boldsymbol{Q_0} = \begin{bmatrix} 1e^{-6} & & \\ & 1 & \\ & & 1 \end{bmatrix} \tag{6.17}$$

Based on the reflection matrix $\boldsymbol{R}$ calculated, the matrix $\boldsymbol{Q}$ can be obtained as Eq. (6.18) from $\boldsymbol{Q} = \boldsymbol{R}^T\boldsymbol{Q_0}\boldsymbol{R}$. The eigenvalues value of $\boldsymbol{Q}$ are $1e^{-6}$, 1 and 1, which are the diagonal elements of the matrix $\boldsymbol{Q_0}$. In other words, the reflection operation preserves the convexity of the objective.

$$\boldsymbol{Q} = \begin{bmatrix} 0.608 & -0.411 & -0.263 \\ -0.411 & 0.569 & -0.276 \\ -0.263 & -0.276 & 0.823 \end{bmatrix} \tag{6.18}$$

After the matrix $\boldsymbol{Q}$ and Θ are obtained, the QP in Eq. (6.13) can be solved by a QP solver. Since no translation is applied, $\boldsymbol{d} = \boldsymbol{0}$. The solution selected from the QP is $\boldsymbol{x}^* = [1.952 \quad 2.048 \quad 1.312]^T$ and the optimal objective value is -8. It can be verified that $\boldsymbol{r_1} = \frac{\boldsymbol{x}^*}{\|\boldsymbol{x}^*\|}$ holds indicating that the semi-major axes of concentric ellipsoids in the $\boldsymbol{r_1}$ direction point towards the optimal solution $\boldsymbol{x}^*$ on the optimal face Θ.

In the example presented above, the direction $\boldsymbol{r_1}$ is arbitrarily selected. If different $\boldsymbol{r_1}$ are used to construct the QP, different solutions will be obtained. Here, Tab. (6.1) shows that by increasing the first element of the $\boldsymbol{r_1}$ an optimal solution with a larger first element is obtained.

Table 6.1: Different $\boldsymbol{r_1}$ can select different optimal solutions from Θ

$\boldsymbol{r_1}$			Optimal Solution			Optimal Objective
$[0.626$	0.657	$0.421]^T$	$[1.952$	2.048	$1.312]^T$	-8
$[0.749$	0.558	$0.357]^T$	$[2.293$	1.707	$1.093]^T$	-8
$[0.952$	0.258	$0.165]^T$	$[3.147$	0.853	$0.547]^T$	-8
$[0.985$	0.145	$0.093]^T$	$[3.488$	0.512	$0.328]^T$	-8
$[1.000$	0.020	$0.013]^T$	$[3.922$	0.078	$0.050]^T$	-8

The example in Eq. (6.14) is also geometrically illustrated in Fig. (6.8). The polyhedron shown in Fig. (6.8) describes the feasible space defined by the constraints in Eq. (6.14) with respect to the three decision variables x_1, x_2 and x_3. The blue surface denotes

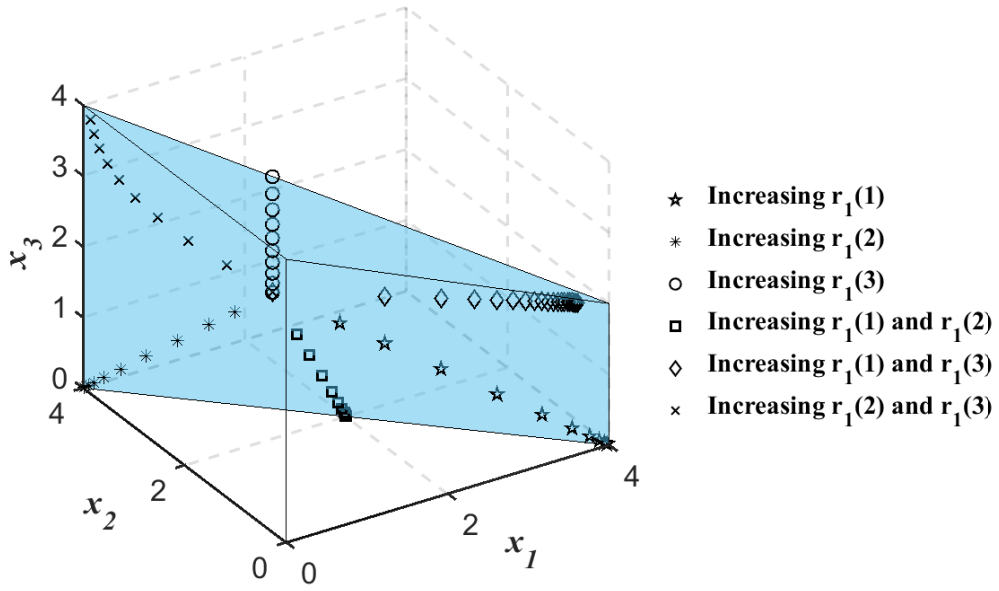


Figure 6.8: Control of elements of $\boldsymbol{r_1}$ to select different optima

the optimal face Θ in Eq. (6.15). On the blue surface, any point is a solution and the objective is -8.

For ERM, since the $\|\boldsymbol{r_1}\| = 1$, increasing one element of $\boldsymbol{r_1}$ means decreasing other elements to keep $\|\boldsymbol{r_1}\| = 1$. Increasing a particular element of $\boldsymbol{r_1}$ of ERM leads to an increase in the corresponding element of a solution $\boldsymbol{x}$ until the selected optimal solution reaches the boundary of the optimal face. In Fig. (6.8), symbols in the figure denote different trajectories of solutions selected by manipulating elements of $\boldsymbol{r_1}$. For instance, the square denotes the trajectory of the solutions by increasing the first and second elements and decreasing the third elements of $\boldsymbol{r_1}$ in ERM. As the first and second elements of $\boldsymbol{r_1}$ increase, the first and second elements of the selected solution also rise.

6.6.2 Comparison of Computational Expense

A comparative simulation study is conducted to illustrate the influence of the number of constraints on the computational expense of ERM versus the interior point WPDM method. An LP example referred to as "lp_agg2" from a repository of models[21] was used. The original example "lp_agg2" contains 516 constraints and 758 decision variables. Test examples were created by randomly selecting a different number of constraints and keeping the same objective as the original example. In this research, test examples with

50, 100, 200, 300, 400, and 500 constraints were created from the original example. To ensure that these test examples have multiple solutions, one inequality constraint with the same coefficients as the objective of the original example is added. Then, these examples were classified according to the number of constraints, and the average computation time was calculated for each case.

Fig. (6.9) presents the average computation time of these created test examples. Since many of the reported DFBMs have large constraints the effect of the number of constraints on computations is critical. For both, ERM and WPDM, the average computation time correlates with the number of constraints. However, the computation time of WPDM increases faster than the ERM as the number of constraints increases. For WPDM, the fitting parameter is $\boldsymbol{w}$ and the number of elements of $\boldsymbol{w}$ increases as the number of constraints increases. For ERM, regardless of the number of constraints, the number of elements of the vector $\boldsymbol{r_1}$ used to select an optimal solution remains the same. Thus, ERM provides a significant advantage of computational time when the number of constraints is large. Furthermore, this computational advantage is expected to be critical for problems involving online estimation or adaptation based on metabolic flux models.

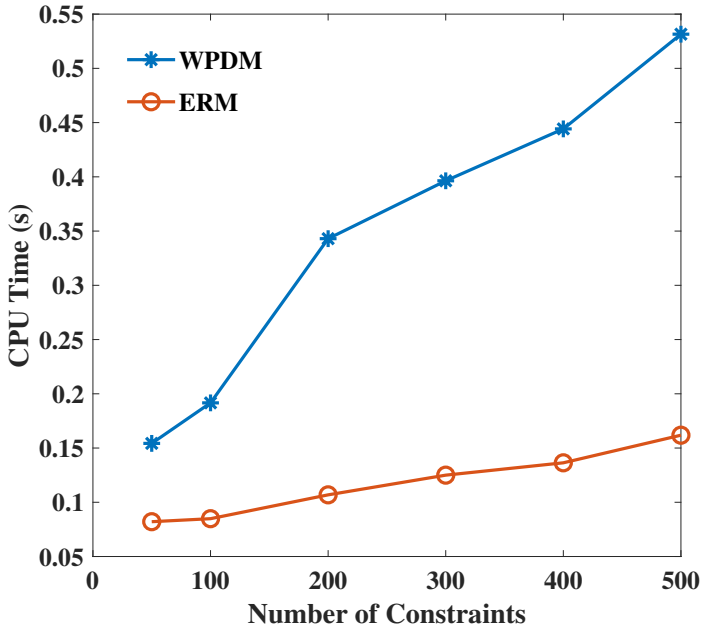


Figure 6.9: Comparison of average computation time of WPDM and ERM for different numbers of constraints

6.6.3 Example of *B. Pertussis* Model

Model Calibration

The objective in this example is to compare the performance of our newly proposed method, i.e. ERM, to our previously proposed approach, i.e. WPDM, for the problem of calibrating a DFBA model. To fairly compare ERM with WPDM when multiplicity occurs, this research follows a bi-level optimization strategy as done with WPDM and it applies both methods with the same experimental data used in [98]. Two groups of replicates' experimental data of *B. pertussis* reported in the paper of [13] were used to calibrate the dynamic flux balance model, referred to experiments 1 and 2 in the following sections.

When ERM is used to solve the LP in Eq. (6.2), the flux $\boldsymbol{v_k}$ can be express as a function of $\boldsymbol{\beta}$, $\boldsymbol{\psi_k}$, $\boldsymbol{r_1}$. In DFBM, $\boldsymbol{\beta}$ are the parameters of the LP, e.g. parameters in kinetic rate constraints expressions, and $\boldsymbol{\psi_k}$ is the state at each time step, i.e. the concentration of a metabolite or biomass. The parameters $\boldsymbol{r_1}$ determine which solution is selected from the multiple optima. Therefore, the flux will be a function $\boldsymbol{v_k} = ERM(\boldsymbol{r_1}, \boldsymbol{\beta}, \boldsymbol{\psi_k})$. During the fitting with experimental data, $\boldsymbol{r_1}$ and $\boldsymbol{\beta}$ serve as tuning parameters for fitting the model predictions to data.

Hence, the evolution of the state is determined by parameters $\boldsymbol{\beta}$ and $\boldsymbol{r_1}$ together. Model calibration consists of tuning these parameters by optimization to fit the experimental data. The problem of fitting the model to experimental data can be expressed as a bi-level optimization problem as follows:

$$\min_{\boldsymbol{\beta}, \boldsymbol{r_1}} \quad \sum_k \sum_m (\psi_{m,k} - \psi_{m,k}^{exp})^2 \tag{6.19a}$$

$$\text{subject to} \quad \boldsymbol{\psi_{k+1}} = \boldsymbol{\psi_k} + \Delta t \psi_{bio,k} \boldsymbol{S} \boldsymbol{v_k} \tag{6.19b}$$

$$\boldsymbol{v_k} = ERM(\boldsymbol{r_1}, \boldsymbol{\beta}, \boldsymbol{\psi_k}) \tag{6.19c}$$

$$\boldsymbol{\psi_0} \text{ is initial states of DFBM} \tag{6.19d}$$

$$k = 0 \cdots \tag{6.19e}$$

where $\psi_{m,k}$ is the value of the m-th state of $\boldsymbol{\psi_k}$ calculated by the model and $\psi_{m,k}^{exp}$ is the value of the state interpolated from cubic splines fitted to the experimental data. Then, the fitting procedure is based on the minimization of the summation of squared errors of measured metabolites with respect to the tuning parameters $\boldsymbol{\beta}$, $\boldsymbol{r_1}$.

The problem formulated in Eq. (6.19) is a bi-level optimization. The outer level involves the minimization of the sum of square errors between model predictions. This

outer optimization is done with respect to model parameters $\boldsymbol{\beta}$ and the "rotation" vector $\boldsymbol{r_1}$ as shown in the previous example where changes of $\boldsymbol{r_1}$ will serve to select a particular solution that results in best fit of experimental data. The inner optimization assumes the $\boldsymbol{\beta}$ and $\boldsymbol{r_1}$ are given so that the LP problem is solved by ERM over the duration of the entire batch culture. On the other hand, for fitting the DFBM with the WPDM method, the tuning parameters are $\boldsymbol{\beta}$ and the interior points weights $\boldsymbol{w}$.

Although all the parameters of $\boldsymbol{\beta}$ and parameters $\boldsymbol{w}$ of WPDM or $\boldsymbol{r_1}$ of ERM could be manipulated to fit the experimental data, this is very impractical because only a small subset of the parameters has a significant impact on the solution [44, 69]. Thus, sensitivity analysis is used to determine the top 5 elements of $\boldsymbol{\beta}$ and the top 5 elements of $\boldsymbol{w}$ of WPDM that have the largest effect on the solution. For ERM we also conduct a sensitivity analysis and select the same 5 elements of $\boldsymbol{\beta}$ that were chosen for WPDM together with the top 5 elements of $\boldsymbol{r_1}$ of ERM that have the most impact on the fluxes.

To simplify the notation, δ is used to represent all parameters ($\boldsymbol{\beta}$ for DFBM, $\boldsymbol{r_1}$ for ERM, $\boldsymbol{w}$ for WDPM) that can be tuned to fit experimental data. The sensitivity S^{δ_i} is the sensitivity of parameter δ_i defined as Eq. (6.20).

$$S^{\delta_i} = \sum_m \sum_k \left| S^{\delta_i}_{\psi_{m,k}} \right| \tag{6.20}$$

where $S^{\delta_i}_{\psi_{m,k}}$ is the sensitivity of m-th state ψ_m to the i-th parameter δ_i at sampling time interval k. The biomass and the volume of culture are treated as states. The calculations of sensitivity of the states with respect to tuning parameters are calculated with Eq. (6.21).

$$S^{\theta_i}_{\psi_{m,k}} = \frac{\partial \psi_{m,k}}{\partial \theta_i} \frac{\theta_i}{\overline{\psi}_m} \tag{6.21}$$

where $\overline{\psi}_m$ is the average of the m-th state over all sampling time intervals.

Comparison of ERM with WPDM

The fitting parameters of WPDM and ERM are elements of weights $\boldsymbol{w}$ and "rotation" direction $\boldsymbol{r_1}$ respectively. After sensitivity analysis, 5 elements of $\boldsymbol{w}$ and 5 elements of $\boldsymbol{r_1}$ together with 5 elements of model parameters $\boldsymbol{\beta}$ are used to fit two groups of experimental data for WPDM and ERM respectively. In the previous research [98], WPDM has shown that 5 elements of model parameters $\boldsymbol{\beta}$ and 5 elements of $\boldsymbol{w}$ can fit the same experimental data much better than other solvers, such as the dual-simplex methods and the naive

interior point method of IBM CPLEX and MATLAB. For the same experimental data, WPDM was also shown to be superior to other methods that were specifically proposed to address multiplicity, including the minimization of total flux, the minimization of the number of active reactions, and hierarchical optimization (lexicographic optimization).

Table. (6.2) is the fitting result of the sum of squared errors for different methods. Figs. (6.10) and (6.11) shows plots of the model predictions and data for different amino acids and biomass evolutions with time respectively. For confidentiality reasons, all concentrations are normalized so that concentrations are dimensionless. As shown in these figures, the results of ERM and WPDM are very similar. Both methods can fit some amino acids well but slightly worse in other amino acids. WPDM fits well the threonine and ERM fits well with lysine. As SSE in shows Table. (6.2), the total SSEs for ERM and WPDM are very close as well but ERM has relatively smaller errors than WPDM.

While ERM and WPDM show similar fitting performances for the multiplicity issue, the two methods are significantly different in terms of computation time. For a fair comparison, tests are conducted on the same computer and the same version of MATLAB and repeated 100 times to calculate the average computation time. On average, an LP with 50 decision variables and 220 constraints requires $0.0676s$ by WDPM and $0.0116s$ by ERM. Thus, ERM is 5.8 times faster than WPDM. One main reason for this computational efficiency is that ERM can take advantage of commercial solvers to reach fast speed and high accuracy. If more constraints are used in the DFBM, ERM still has the same number of parameters but WPDM will have more parameters to fit, which greatly increases the computation. The efficiency of ERM makes it possible to apply in the future to larger metabolic networks.

6.7 Conclusions

A novel ERM method is developed to address the multiplicity of LP problems. The method is successfully applied to a toy example and for calibrating the DFBM of *B. pertussis* during fermentation. The method scales well with the dimensions of the metabolic network because the number of calibrating parameters does not increase with the number of constraints. Since ERM is low-code and based on commercial LP and QP solvers, it is computationally efficient and suitable for metabolic flux modeling of large metabolic networks for which multiple solutions exist. The computational advantages are expected to be particularly important for online DFBA-based estimation or optimization problems.

Table 6.2: Summation of Squared Errors (SSE) of Fitting by ERM and WPDM

	SSE of ERM	**SSE of WPDM**
Biomass	3073	3333
Ala	0.126	0.180
Arg	1.030	0.754
Asp	3.522	1.746
Glu	395.6	407.6
Gly	14.36	14.30
His	1.638	1.712
Ile	0.559	0.616
Leu	2.322	1.782
Lys	15.56	31.14
Met	6.708	6.831
Phe	8.274	8.284
Pro	0.210	1.344
Ser	1.428	0.194
Thr	18.80	2.381
Tyr	0.078	0.075
Val	5.96	5.55
Total	3549	3818

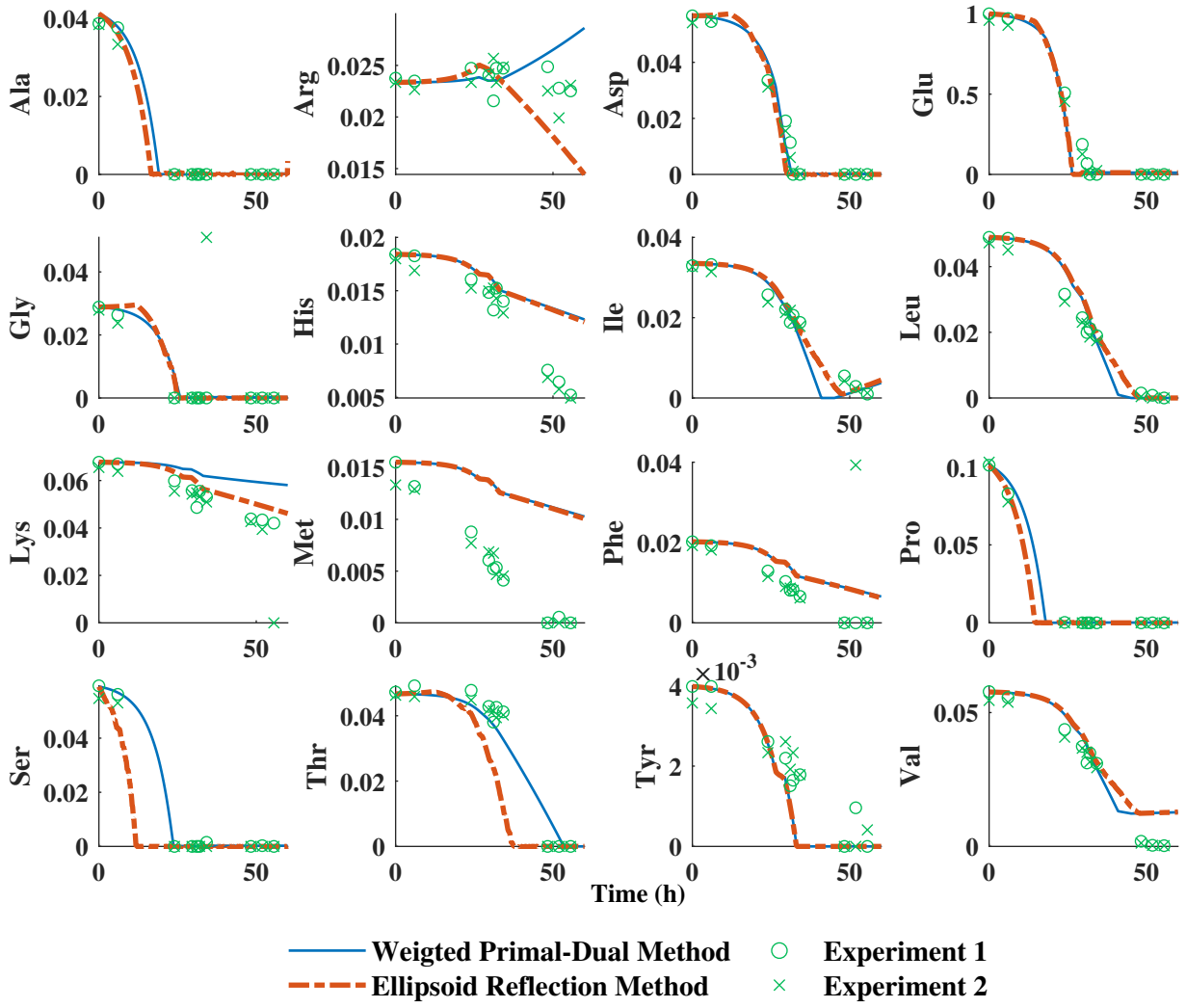


Figure 6.10: Evolution of key metabolite concentrations with time by WPDM and ERM

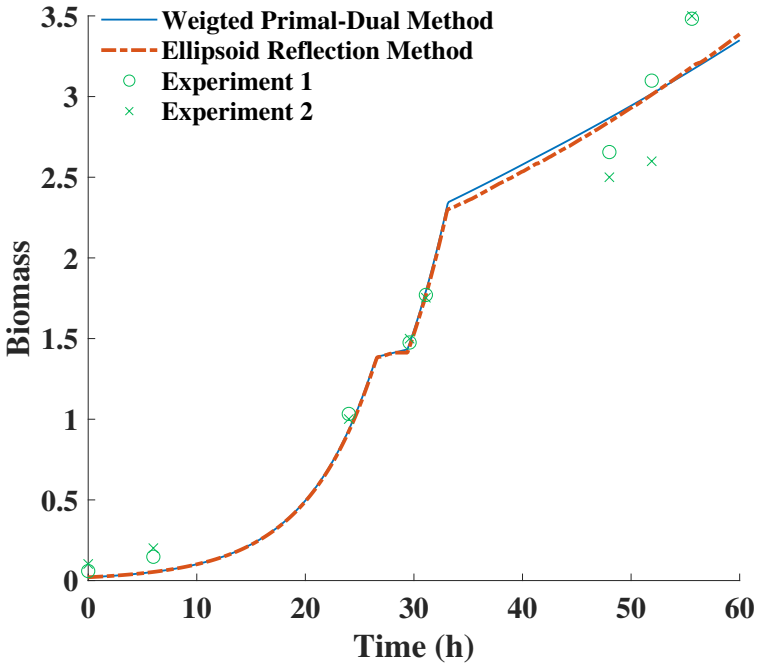


Figure 6.11: Evolution of biomass concentrations with time by WPDM and ERM

Acknowledgements

This work was supported NSERC Discovery Grants Program under grant 50503-10882, Sanofi-Pasteur and Mitacs.

Chapter 7

Setting up an Experimental Platform for Online Estimation Based on Dynamic Flux Balance Models

7.1 Overview

In the previous chapters, the weighted primal-dual method (WPDM) and the ellipsoidal reflection method (ERM) have been proposed to solve the multiplicity issue of dynamic flux balance models (DFBM). Based on the improved DFBM, different methods of set membership estimation were designed to estimate the state of metabolism in the bioreactor with limited measurements. To test these proposed ideas, an experimental platform was required for future applications. Using available equipment in our laboratory, a batch culture of *B. pertusis* was conducted, analyzed, and modeled with DFBM.

7.2 Introduction

This thesis focuses on the use of DFBM for process modeling and monitoring. However, applying DFBM has two major problems. DFBM contains a linear programming (LP) problem at each time step and the LP needs to be solved to proceed to the next time step. The multiplicity of solutions of the LP poses a challenge for predicting the time evolution of metabolites along a batch. In previous research, two data-driven methods WPDM and ERM have been proposed to select the optimal solution that best fits experimental data.

Compared to methods that assume the metabolism of cells is efficient, these two methods fit experimental data better because they can select specific solutions.

A second challenge for the use of DFBM for process monitoring is the lack of online measurements. DFBM considers more metabolites than the classical unstructured models traditionally used in bioprocess monitoring. Online measurements of most metabolites are impossible to measure in real-time since they require time-consuming techniques such as HPLC. The lack of online measurements hinders the ability to design an observer to estimate unmeasured metabolites because their observability is not satisfied. To overcome this scarcity of measurements, different set membership estimations for DFBM are constructed in previous chapters.

While set membership estimation works well in simulations, it was important to further prove its applicability in an experimental setting. In view that part of this research was funded by a two-term MITACS scholarship, I developed a bench scale platform for batch culturing of *B. pertussis* which is currently used by Sanofi for the manufacturing of the whooping cough vaccine. The platform is a scale-down model (2L) of the industrial 20L -2000 L reactors used by Sanofi.

This chapter includes three sections: materials and methods, calibration of dynamic flux balance modeling, and results and discussion.

7.3 Materials and Methods

7.3.1 Setting up of Equipment

The entire platform can be classified into different parts, bioreactor, control console, probes, pump, and camera as shown in Fig. (7.1). The bioreactor (Applikon, Netherlands) is operated in a batch fermentation mode. The control console is composed of two parts, a controller (ADI 1030; Applikon, Netherlands) and a console (ADI 1025; Applikon, Delft, Netherlands). The controller receives the signals from probes, displays process variables on the screen, and controls the key process parameters. The console has two mounted peristaltic pumps controlled by the controller and one rotary knob to tune the speed of agitation. Probes measure the dissolved oxygen (DO), pH, and temperature and send these signals to the controller. The camera (C920S; Logitech, Switzerland) is used to capture the experimental data on the screen of the controller by a program in Python, and the photos captured are stored on the laptop (Dell, USA). The pump (MasterFlex®, Germany) is used for inoculation and sampling.

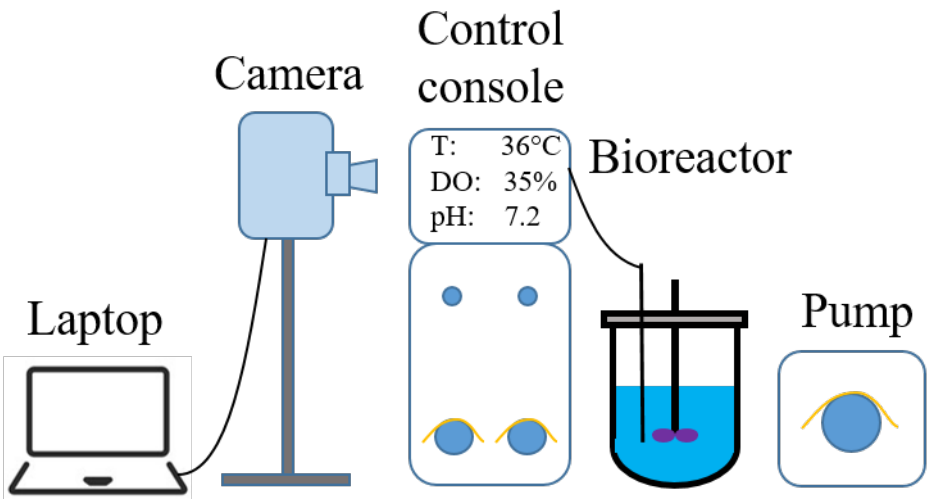


Figure 7.1: Setting up of Equipment

7.3.2 Culture Conditions and Operations

The culture conditions of *B. pertussis* are adapted from a scaled-down model developed by Sanofi. This scaled-down model is used to analyze industrial fermentation. This research adapted similar conditions to the scaled-down model for simulating the industrial setting. A *B. pertussis* strain from Sanofi was used in all experiments. The batch media is a modified version of Stainer-Scholte media supplemented by casamino acids (Bacto™ Casamino Acids; Thermo Fisher Scientific, USA). The media was prepared, then sterilized in an autoclave for 40 mins, and stored at 5°C for the culture in the shake flask. Growth factors include vitamins and salts combined together with media to promote better growth. Growth factors were prepared, then sterilized through a filter and stored at 5°C for culture in the shake flask and bioreactor.

A cryovial containing 1 mL seed stored at −80°C melt was inoculated in a sterile 500 mL flask containing 60 mL media, 57.8 mL sterile ultrapure water, and 1.2 mL growth factors. The shake flask is controlled at 200 rpm and 36°C in an incubator (NU-5510/E; NuAire, USA). The culture of the shake flask was conducted for 38 h until the OD600 (optical density measurements at 600 nm) was within 4 to 6 for inoculation to the bioreactor. The initial OD600 in the bioreactor was controlled at 0.3 by the amount of inoculation. Finally, the culture volume after inoculation was approximately 1.6 L.

DO probe (InPro®6800; Mettler Toledo, OH, USA) was calibrated at its zero value by nitrogen and at 100% value by air after the readings of DO were stable. A pH probe

(Applikon, Netherlands) was calibrated by standard buffers (Thermo Fisher Scientific, USA) at pH values of 4.01 and 7 respectively. The temperature was controlled in a closed-loop by a PI controller at a target of 36°C by a heating jacket. The pH was controlled in a closed-loop by a PID controller at a target of 7.2 by adding 2.5 M sterile phosphoric acid. 1.5% sterile antifoam (Antifoam 1520; Dow Corning, USA) was added manually to minimize the generation of foam. The dissolved oxygen was controlled in a closed loop by a PID controller at a target of 35% by manipulating the aeration rate. The agitation rate was manually controlled and increased from 200 rpm initially to 625 rpm at 29.5 hours. Once the agitation could not satisfy the DO target, the agitation rate was manually increased. During the first half of fermentation, the agitation rate increased roughly by 50 rpm every two hours. During the second half of the fermentation, the agitation rate was increased slower than for the first half. Sampling was conducted at 0 h, 4 h, 8 h, 16 h, 24 h, and at the end of the fermentation. Each time a 2 mL sample of culture was collected and stored in two 1.5 mL centrifuge tubes separately. The fermentation was stopped at about 29.5 h when a peak of DO was observed corresponding to the complete depletion of glutamate.

7.3.3 Analysis of Culture

Determination of Contamination

To prevent contamination, tryptic soy agar (Millipore Sigma, USA) and Bordet-Gengou agar (Thermo Fisher Scientific, USA) plates are used to confirm the absence of contamination. *B. pertussis* can grow on Bordet-Gengou agar but cannot grow on tryptic soy agar [31].

Determination of Biomass

Samples from the shake flask or from the bioreactor were diluted by 0.9% saline water (Intermountain Life Sciences, USA) and mixed well. The biomass concentration was determined by optical density measurements at 600 nm using a spectrophotometer [13].

Determination of Metabolites

AccQ.Tag^{TM} method with a pre-column is used to quantify the concentrations of amino acids in the culture. AccQ.Tag^{TM} method uses overdosed AQC
(6-aminoquinolyl-N-hydroxysuccinimidyl carbomate) to derivatize the amino acids. The

is procedure generates highly stable fluorescent derivatives which can be measured by fluorescence detectors to quantify the concentrations. The pre-column is used to separate these amino acid derivatives.

Samples collected from the bioreactor were centrifuged 3 mins at 10000g to remove the cells and then filtered by 0.2 μm MCE filter (Thermo Fisher Scientific, USA) to collect the supernatant. The samples of supernatant were stored at -20°C for later analysis. Frozen samples of supernatant were thawed at room temperature and then diluted with ultrapure water for HPLC. Further details of this analysis are provided below.

Following the Waters AccQ.Fluor^{TM} Reagent Kit instructions sheet, Waters AccQ.Fluor^{TM} Reagent Kit was used to derivatize the supernatant. 1 ml of reagent diluent was transferred to a reagent powder container and vortexed for 10 seconds to reconstitute the AccQ.Fluor reagent. The container of AccQ.Fluor reagent was closed, sealed, put in a 55°C water bath, and vortexed until the reagent powder was fully dissolved. 10 μL of the sample and 70 μL of AccQ.Fluor Borate Buffer was vortexed in a centrifuge tube. Then, 20 μL reconstituted AccQ.Fluor reagent was added into the centrifuge tube, vortexed, and incubated for 1 min at room temperature. The derivatized sample was transferred to a clean autosampler vial with a 250 μL limited volume insert and closed with a silicone-lined septum cap. This autosampler vial was then placed in the water bath at 55°C for 10 minutes.

The Waters 1525 chromatography system (Waters Corporation, MA, USA) with a fluorescence detector (W2475; Waters Corporation, MA, USA) was used to analyze amino acids. The AQC pre-column (Waters AccQ.TagTM Amino Acid Analysis Column C18, dimensions 4 μm, 3.9 mm $\times$ 150 mm) was preheated to 38°C. For each analysis, 5 μL of derivatized sample in the autosampler vial was injected into the column. The fluorescence detector was set to an excitation wavelength of 248 nm and an emission wavelength of 395 nm.

The eluent A was an acetate-phosphate buffer solution prepared by mixing 100 mL AccQ.TagTM Eluent A concentrate (Waters, MA.USA) with 1 L of ultrapure water. The eluent B was HPLC-grade acetonitrile. The flow rate of each eluent at different times for the HPLC run was adapted from the work of Cohen [19] and are presented in Tab. (7.1). The HPLC analysis for each sample runs for 45 min. Before each run starts and in between samples run, the column was equilibrated for 15 min using 99% of eluent A and 1% of eluent B for 15 minutes at 1 mL/min.

Table 7.1: Gradient table of eluents for HPLC analysis

Time (min)	Flow rate (mL/min)	Eluent A Buffer (%)	Eluent B ACN (%)	Gradient Curve
0	1	99	1	-
0.5	1	98	2	6
18	1	95	5	6
19	1	91	9	6
30	1	80	20	6

7.4 Dynamic Flux Balance Model

Dynamic Flux Balance Model with the Ellipsoidal Reflection Method

A DFBM model was used for model fitting with the data generated in the experiments. The DFBM is presented in detail in Eq. (6.2) and Eq. (6.1) of Chapter 6.

Model Calibration

The DFBM model previously developed [13] was used to calibrate for new experimental data. The DFBM model was calibrated using the ERM method presented in Chapter 6 of this thesis. Given the DFBM with ERM, the evolution of the state is determined by parameters $\boldsymbol{S}$, $\boldsymbol{\beta}$, and $\boldsymbol{r_1}$ together. Model calibration involves tuning these parameters to optimize the fit to the experimental data. The fitting of experimental data can be expressed as an optimization problem.

$$\min_{\boldsymbol{S_{bio}},\boldsymbol{r_1},\boldsymbol{\beta}} \quad \sum_k \sum_m (\psi_{m,k} - \psi_{m,k}^{exp})^2 \tag{7.1a}$$

$$\text{subject to} \quad \boldsymbol{\psi_{k+1}} = \boldsymbol{\psi_k} + \Delta t \psi_{bio,k} \boldsymbol{S v_k} \tag{7.1b}$$

$$\boldsymbol{v_k} = ERM(\boldsymbol{r_1}, \boldsymbol{\beta}, \boldsymbol{\psi_k}) \tag{7.1c}$$

$$\boldsymbol{\psi_0} \text{ is initial states of DFBM} \tag{7.1d}$$

$$k = 0 \cdots \tag{7.1e}$$

where $\psi_{m,k}$ and $\psi_{m,k}^{exp}$ are m-th state of $\boldsymbol{\psi_k}$ calculated by the model and from cubic spline interpolation of experimental data respectively. $\boldsymbol{S_{bio}}$ is the column of matrix $\boldsymbol{S}$ related to the biomass synthesis. The goal is to minimize the sum of squared errors between predicted

and measured metabolites' concentrations at all time steps by tuning parameters $\boldsymbol{\beta}$, $\boldsymbol{r_1}$, and $\boldsymbol{S_{bio}}$. Notice that in this research and in most DFBM studies in general, the stoichiometry coefficient matrix $\boldsymbol{S}$ is not considered a tuning parameter and instead is determined from biochemical information sources. However, in our case $\boldsymbol{S_{bio}}$ is tuned because the biomass composition can change due to the differences in the culture conditions. In Chapter 6, the fitting parameters only include $\boldsymbol{\beta}$ and $\boldsymbol{r_1}$. Since the culture conditions of new experimental data are not the same as the old experiments when the model was developed, it was hypothesized that the composition of biomass can also change so it is added as a fitting parameter. Once the biomass composition changes, the yields of substrates with respect to biomass may vary so that $\boldsymbol{S_{bio}}$ may also change.

Since the total number of possible tuning parameters is very large, sensitivity analysis was done to select the most sensitive parameters to reduce the computational expense. For simplicity, δ is used to represent the tuning parameters vector ($\boldsymbol{S_{bio}}$, $\boldsymbol{\beta}$, and $\boldsymbol{r_1}$) that can be varied to fit the experimental data. The sensitivity L^{δ_i} is the sensitivity of i-th parameter of δ defined as in Eq. (7.2).

$$L^{\delta_i} = \sum_m \sum_k \left| L^{\delta_i}_{\psi_{m,k}} \right| \tag{7.2}$$

where $L^{\delta_i}_{\psi_{m,k}}$ is the sensitivity of m-th state ψ_m with respect to the i-th parameter δ_i at sampling time interval k. The calculations of sensitivity of the states are done according to Eq. (7.3).

$$L^{\delta_i}_{\psi_{m,k}} = \frac{\partial \psi_{m,k}}{\partial \delta_i} \frac{\delta_i}{\overline{\psi}_m} \tag{7.3}$$

where $\overline{\psi}_m$ is the average of the m-th state over all sampling time intervals. According to the sensitivity analysis, parameters with the largest L values are selected to fit experimental data.

7.5 Results and Discussion

7.5.1 Determination of Contamination

The experimental data presented in this chapter is referred to as batch F06. To assure that no contamination occurred during batch F06, sterile centrifuge tubes and sterile syringes were used to extract samples from the bioreactor. When the DO peak (over DO setting point 35 % for 5 mins) was observed at about 29.5 h, an event that was used as an indication

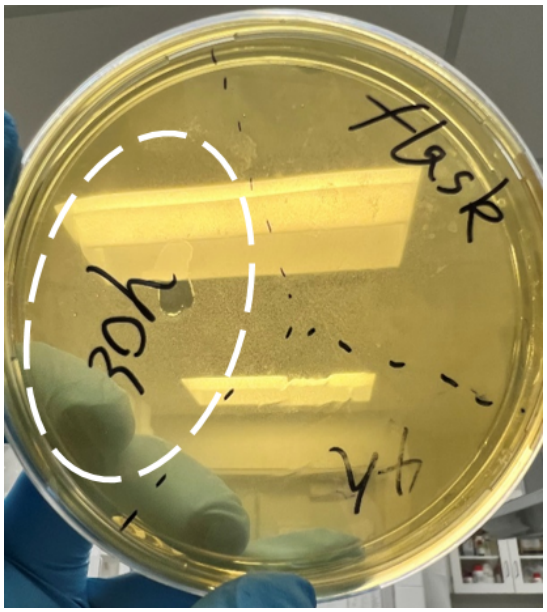
of complete depletion of glutamate, the last sample was collected. The sample at 29.5 h was inoculated on a tryptic soy agar plate and a Bordet-Gengou agar respectively. Two plates were cultivated at 35°C for 3 days in the incubator. Fig. (7.2a) and (7.2a) show that the bacteria could not grow on the tryptic soy agar but could grow on the Bordet-Gengou agar and formed the colony in brown color. The colony formed on the Bordet-Gengou agar was picked and further inoculated on a new tryptic soy agar, and incubated at 35°C for 3 days in the incubator. Fig. (7.2c) shows that no colony was formed on the tryptic soy agar (the marks are the scratches made by the inoculating loop). These experiments confirmed that batch F06 succeeded and no contamination occurred.

7.5.2 Determination of Biomass

OD 600 was used in this research to quantify biomass concentration. F06 refers to a batch of experiment conducted as per the description above. FER 1120 refers to typical experimental data from a 2 L benchtop bioreactor. The data was extracted from figures shown in a Sanofi report [92], which was used for comparison with F06. For confidential reasons, OD 600 data from F06 and FER 1120 were normalized with initial OD 600 respectively. The initial OD 600 of FER 1120 and F06 were different. F06 was conducted with a working volume of 1.6 L whereas FER 1120 was conducted with a working volume of 2L. As shown in Fig. (7.3), it was evident that the biomass observed in F06 was much higher than the FER 1120 after 24 h.

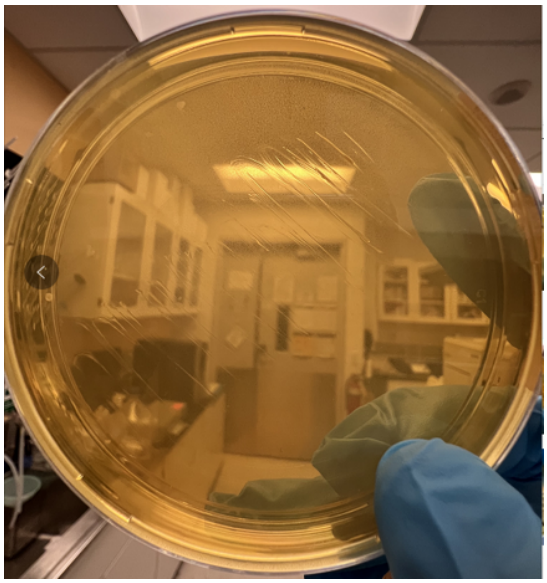
7.5.3 Determination of Metabolites

The calibration of the model relies on extracellular concentrations of amino acids. Most amino acids in the media were from casaaminoacids, which are obtained from the hydrolysis of milk and are a key part of the growth media used by Sanofi. The main carbon source for *B. pertussis* is glutamate, which is consumed in the TCA cycle to generate ATP and maintain cell growth and functions. In fact, it has been reported that *B. pertussis* can grow with a basic media containing glutamate and proline. Other amino acids are used for the synthesis of biomass and antigens. Different dilution factors were used to measure the amino acids by HPLC and replicate measurements were conducted to verify accuracy. For confidential reasons, the measured concentrations were normalized by the initial value of glutamate concentration. The concentrations measured of batch F06 are presented in Tab. (7.2).



(a) Sample at 29.5 h was inoculated on the tryptic soy agar

(b) Sample at 29.5 h was inoculated on the Bordet-Gengou agar



(c) The colony picked from the Bordet-Gengou agar was inoculated onto the tryptic soy agar

Figure 7.2: Sample inoculated on the Bordet-Gengou agar and tryptic soy agar

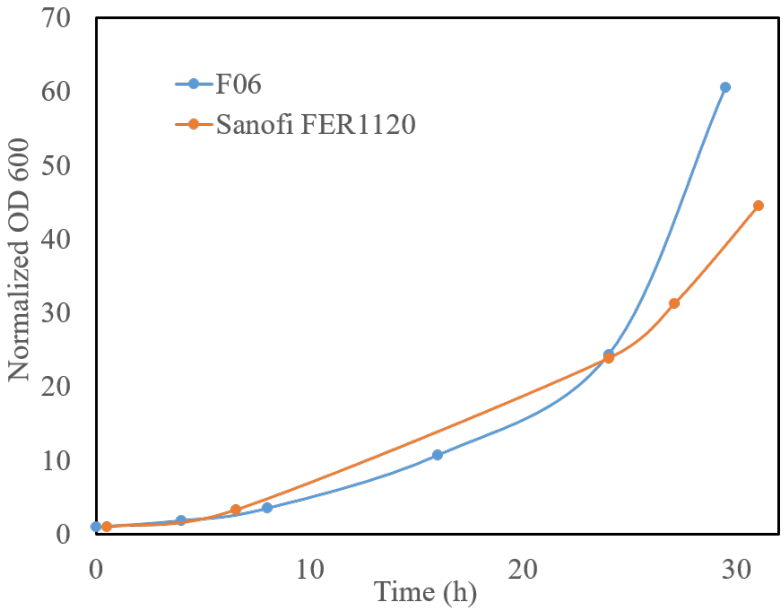


Figure 7.3: Comparison of normalized biomass of batch F06 with batch FER 1120 from Sanofi

Table 7.2: Normalized amino acid concentrations in the culture by HPLC analysis

Time (h)	0	4	8	16	24	29.5
Asp	0.0718	0.0617	0.0606	0.0449	0.0412	0.0285
Ser	0.0462	0.0361	0.0333	0.0174	0.0003	0.0001
Glu	1.0000	0.8619	0.8401	0.5953	0.4652	0.2424
Gly	0.0213	0.0148	0.0128	0.0058	0.0000	0.0000
His	0.0127	0.0079	0.0079	0.0054	0.0084	0.0088
Arg	0.0137	0.0126	0.0122	0.0091	0.0105	0.0117
Thr	0.0225	0.0201	0.0194	0.0146	0.0178	0.0169
Ala	0.0438	0.0343	0.0308	0.0139	0.0017	0.0015
Pro	0.2352	0.2078	0.1871	0.1111	0.0996	0.1094
Val	0.0491	0.0418	0.0406	0.0294	0.0301	0.0280
Met	0.0129	0.0109	0.0109	0.0073	0.0073	0.0057
Lys	0.0707	0.0607	0.0586	0.0471	0.0508	0.0529
Ile	0.0280	0.0233	0.0226	0.0167	0.0178	0.0168
Leu	0.0415	0.0346	0.0339	0.0231	0.0224	0.0190
Phe	0.0172	0.0141	0.0134	0.0094	0.0093	0.0081

7.5.4 Model Calibration

The DFBM model is calibrated with respect to a set of tuning parameters that include $\boldsymbol{\theta}$, $\boldsymbol{r}$, and $\boldsymbol{S}$. Only the elements related to the reaction of biomass synthesis in the matrix $\boldsymbol{S}$ were considered for tuning. For all other reactions in the metabolic networks, the stoichiometric coefficients are fixed and determined by biochemical information about the microorganism. The rationale for tuning the coefficients related to biomass is that its composition can vary for different culture conditions.

Since tuning all the possible parameters was computationally expensive sensitivity analysis with respect to the parameters was conducted to identify the ones that most affect the fitting to data. The sensitivity L for each parameter is calculated as Eq. (7.2). In total, 30 parameters with the highest sensitivity were used to fit the experimental data of batch F06.

The results of this fitting procedure are presented in Fig. (7.4), which is based on normalized experimental. The states that are well-fitted include Ala, Asp, Glu, Gly, Pro, Ser, and OD600 (biomass). States that could not be fitted by DFBM are Arg, His, Ile, Leu, Lys, Met, Phe, Thr, and Val. An explanation for the inability of the model to fit part of the data is discussed in the following section.

7.5.5 Difference between batch FER 1120 and F06

As shown in Fig. (7.3), there is a significant difference in OD 600 between FER 1120 and F06. FER 1120 is a representative fermentation of 2L at Sanofi and its culture conditions and operating conditions have been used as a scale-down model to analyze fermentations in the industrial scale 2000L bioreactors. Following discussion with researchers and engineers from Sanofi, several possible causes for the differences between our experiments and those conducted by Sanofi were identified as follows:

i- there was a difference in the time between the end of the flask and the inoculation into the bioreactor. At Sanofi, the inoculation from a shake flask into the bioreactor is not always conducted immediately after the culture reaches the late logarithmic phase. To preserve the culture in the late logarithmic phase, the temperature, mixing, and aeration are reduced. Later, the preserved culture is used to inoculate the bioreactor. However, from our experiments, it was observed that this delay period is harmful to the cell and could be one important reason for a long lag phase and slower growth observed at the Sanofi experiment as compared to ours. For example, experimental data collected from a shake flask experiment at our lab is presented in Fig. (7.5). This flask culture was interrupted

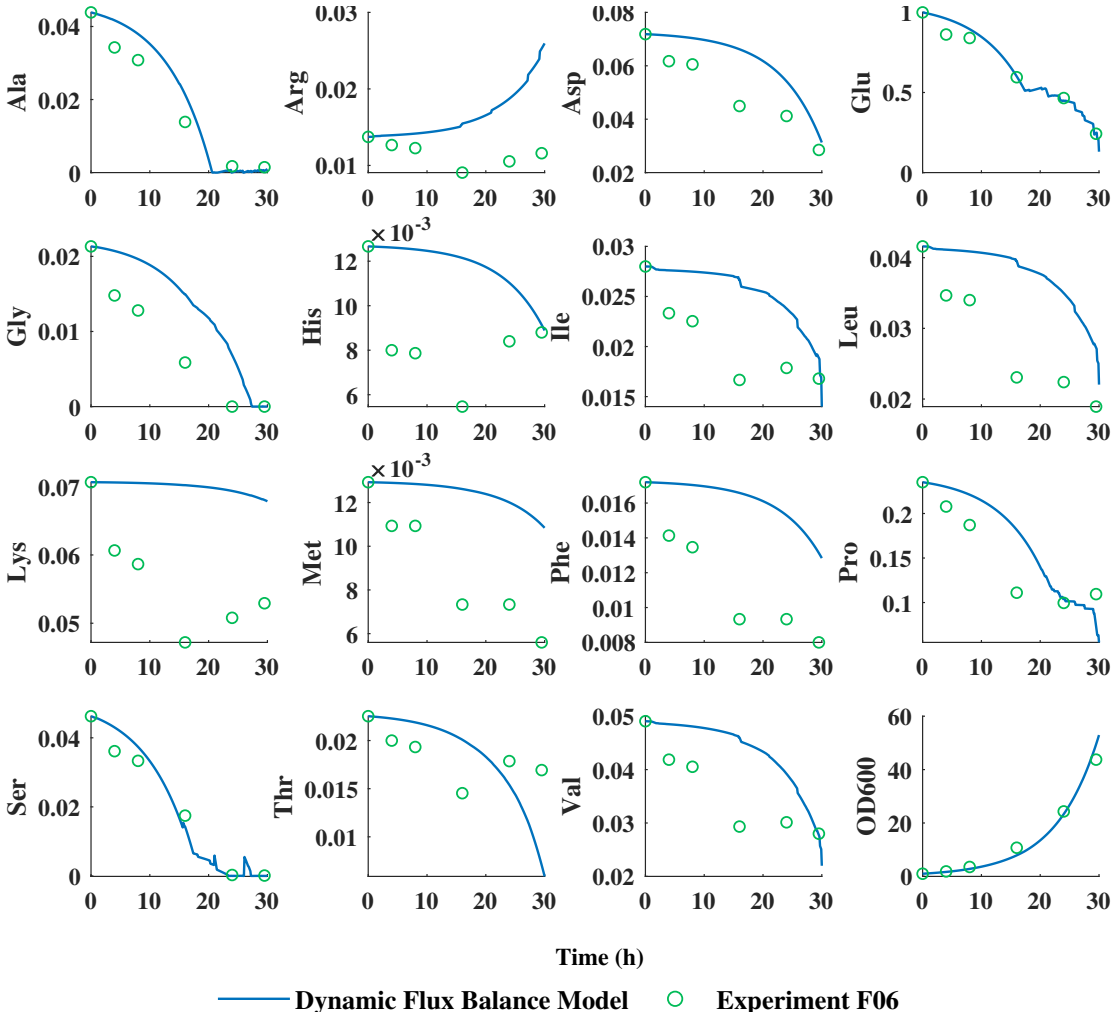


Figure 7.4: Evolution of key metabolite and biomass concentrations with time fitted by DFBM. All data are normalized and dimensionless

at the late logarithmic phase by stopping rotation and by cooling down the temperature for about 10 mins. After this short period, the flask was put back into the incubator with the normal rotation speed and temperature. As the figure shows, the OD 600 drops significantly and needs time to recover from the sudden interruption. Similar experiments [87] have reported that the transcription of virulence genes is regulated by temperature. Thus, if the inoculation of the bioreactor can not be conducted immediately after the end of the flask culture, the drastic temporary decrease in dissolved oxygen and temperature can significantly influence cell growth in the following bioreactor culture. This may explain why F06 which was inoculated after a very short delay from the mature culture in the shake flask reached a higher OD600 as compared to the bioreactor experiment at Sanofi.

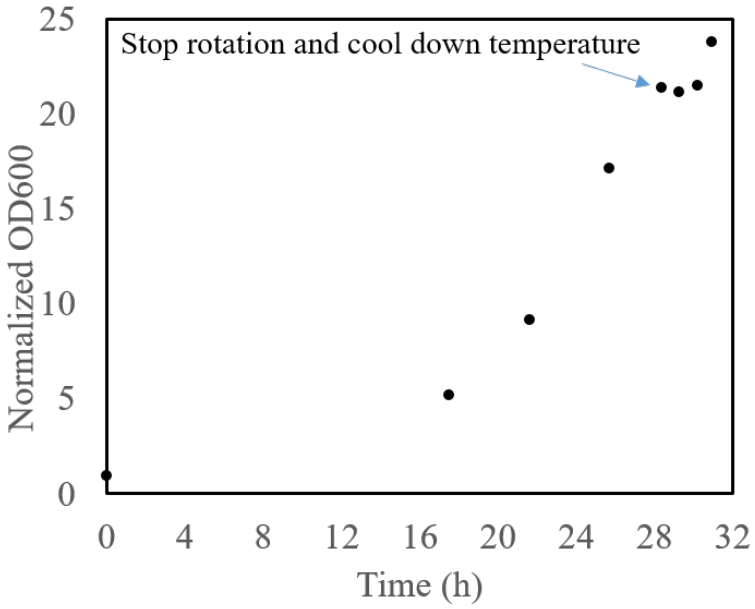


Figure 7.5: Shake flask interrupted by stopping rotation and cooling down the temperature

ii- There are differences with respect to the DO control in the experiments. The DO control strategy of FER 1120 was presented in Fig. (7.6) [92]. Essentially, this control strategy is based on the split control idea where two manipulated variables, agitation and aeration rates, are used to control one controlled variable, i.e. DO. The agitation rate is manipulated to increase the DO when the DO demands of cells are low. Once the agitation rate reaches a saturation limit, since the DO demands of cells are high the agitation rate remains at its maximum value and the aeration rate is increased to satisfy the oxygen demand. However, because of hardware limitations, the bioreactor in our laboratory could

not be operated like the one at Sanofi. As described before, the agitation is manually controlled and aeration is controlled by a PID control with a DO probe. The difference in DO control can also influence cell growth.

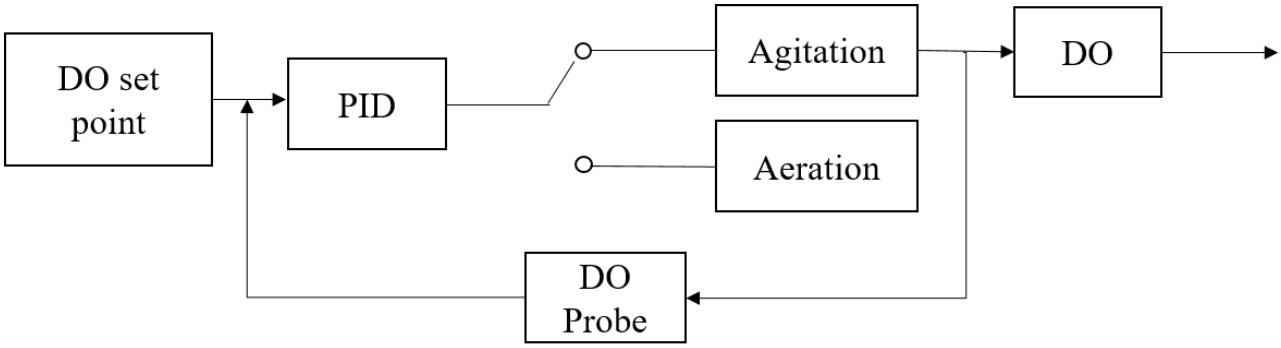


Figure 7.6: DO control strategy of FER 1120

iii- There were differences in the antifoam addition strategy. The antifoam in FER 1120 at Sanofi was added at a constant rate. The addition rate was so high at Sanofi that no foam was observed. However, as Fig. (7.7) shows, foam formed on the inner surface of the bioreactor in batch F06. Since the antifoam was added manually, the antifoam added in our bioreactor was less as compared to FER 1120. Antifoam is known for its adverse effect on cell growth thus possibly explaining the higher OD 600 observed in batch F06.

iv- Glutamate concentrations in F06 were higher than glutamate concentrations of FER 1120. In F06, the glutamate concentrations in the media are from glutamate and casamino acids. On the other hand, according to Sanofi report [92], the presence of glutamate in casamino acids is generally ignored in the calculation of the initial required amount. Since glutamate is the main carbon source in the media, higher glutamate concentrations can cause a difference in metabolism. Also, different brands or batches of casamino acids can also contain differing glutamate concentrations.

7.5.6 Analysis of lack of fitting

Compared to fitting in previous chapters, the fitting of DFBM is less accurate. Different methods, weights, and algorithms were tried to improve the fitting quality. Some amino acids including His, Ile, Leu, Lys, Phe, Val, were difficult to fit even when the parameters were assumed to be time-varying. Thus, the lack of fitting can only be explained by a significant model structure error.

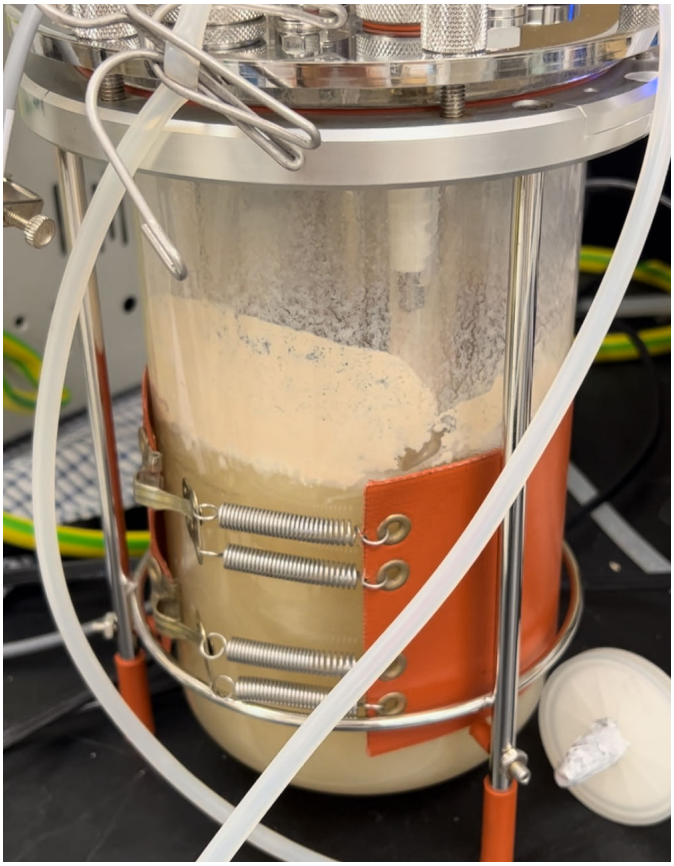


Figure 7.7: Foaming during the fermentation of batch F06

One important source of model structure error is related to metabolic differences occurring for the two types of experiments used for model calibration. Although the culture conditions and operating conditions are quite similar between the experiments, differences were identified in the previous section that was due to limitations of equipment and funding. As shown in the previous section, the growth in F06 is faster than FER 1120, which means that the metabolism is more efficient in F06. It should be remembered that the model structure of the DFBM was originally determined for the experiments conducted at Sanofi. The metabolic difference influences metabolic regulation and biomass compositions. For example, current constraints like $\boldsymbol{g}(\boldsymbol{\psi_k}, \boldsymbol{\beta})$ and $\boldsymbol{h}(\boldsymbol{\psi_k}, \boldsymbol{\beta})$ cannot capture the metabolic regulation occurring in the current experiments as compared to the ones done at Sanofi. Thus, a different set of kinetic constraints may be necessary to explain the new experiments conducted at our laboratory that resulted in significantly higher growth. Even the calibration of stoichiometric coefficients related to biomass synthesis was not sufficient for improving the fitting. It is possible that the biomass equation should include a contribution of ATP related to the fast synthesis of nucleotides that is not present in the current composition considered for biomass.

Measurement error may be another important reason for the observed lack of fitting. In particular, the HPLC results are not suspected to be in error due to column aging and column preservation. Even though the column had been flushed with ethanol for about 24 hours before the experiments started, peak distortion was observed in all results for different dilution factors and different replicate injections. Ideally, the peaks along the elution time spectrum should be highly symmetric and narrowly distributed. The peaks obtained have tails and are widely distributed as shown in Fig. (7.8). It should be noticed from Fig. (7.8), that the distribution of small peaks is wide and tailing possibly indicating that the column is aging or the column is not well preserved.

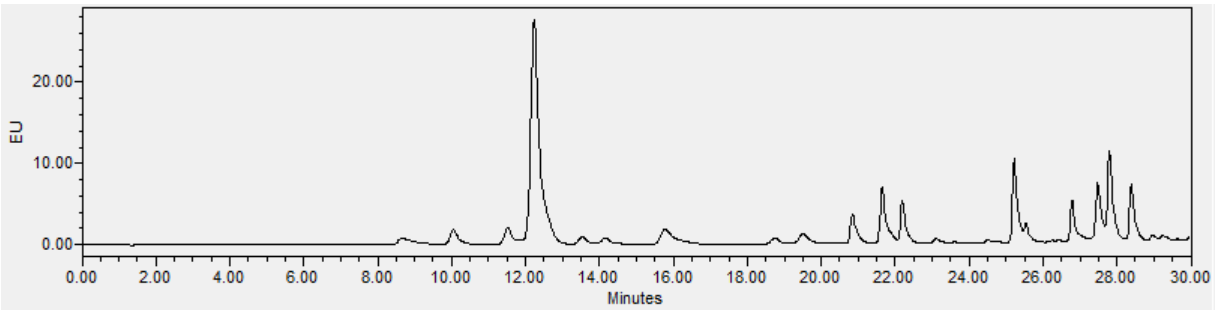


Figure 7.8: HPLC analysis of amino acids in the initial culture sample

7.6 Conclusions

To test fitting techniques and online estimation algorithms developed in this thesis an experimental bioreactor platform of *B. pertussis* has been set up and successfully run. Methods to quantify biomass and amino acid concentrations were tested. However, due to limitations of equipment and funding, different metabolic pattern was observed in the batch culture and consequently, the model could not accurately capture the dynamic behavior of metabolites. Current hardware improvements, including the use of a newer bioreactor, are currently underway. These improvements are expected to lead to more comparable experimental results to the ones conducted by Sanofi.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

DFBM is a type of unsegregated structural model that can capture the dynamics of the metabolites. Research in this thesis is modifications and applications of the DFBM. The main contribution of the thesis can be summarized in four parts, methods to solve the multiplicity issue, methods of set membership estimations, methods to simplify the DFBM and the development of an experimental platform for culturing *B. pertussis*.

8.1.1 Methods to Solve The Multiplicity Issue

The findings of this part of the research can be summarized as follows:

1. A key challenge of DFBM models is that the LP that is solved at each time step may not be unique. This situation is referred to as the multiplicity issue. In this research, we found that the solution that is randomly selected by different commercial solvers can result in different trajectories of the states which may not fit the experiments. If the multiplicity issue cannot be solved, it is very hard to use the DFBM model for applications of DFBM to state estimation, control, and optimization problems.

   Two new methods are proposed in this thesis for addressing multiplicity referred to as the Weighted Primal-Dual Method (WPDM) and the Ellipsoidal Reflection Method (ERM). Both methods can select a specific solution from all possible multiple optima by manipulating internal parameters. WPDM is a type of weighted interior-point

method that can select a specific solution by weights. Since each weight corresponds to a constraint, the number of parameters of WPDM increases with the number of constraints. This leads to a drawback in the fitting of model predictions to experimental data because more tuning parameters means more computation in the fitting process. ERM is a newly proposed method, which solves the LP to find the optimal face and constructs a QP to select a solution by a "rotation" operation of the quadratic objective surface. More importantly, the number of tuning parameters is independent of the number of constraints which is generally large in DFBM models. Thus, ERM greatly reduces significantly the computation expense for large metabolic networks which may contain many constraints. On the other hand, WPDM is still useful since it is used as the basis of the set-based estimation method proposed in Chapter 5. In the latter case, WPDM was combined with multiparametric nonlinear programming (mpNLP) for set membership estimation based on an approximation of the barrier function used in the interior point approach.

2. The applicability of DFBM largely depends on the ability to fit available data. Cells from the same species may exhibit different metabolic patterns when exposed to different experimental conditions. If a specific DFBM model lacks a sufficient number of constraints multiple optima may occur but only one of these will best fit the experimental data. Typical optimization solvers such as the Simplex or Interior-point method are not tailored to select the specific optimal solution that correctly captures the metabolism of a cell when different optimal solutions are possible.

   Although different methods have been proposed to solve the multiplicity issue, most of them are based on the assumption that cell behavior follows an efficient metabolism. However, this may not be true for engineered strains used for specific industrial purposes. WPDM and ERM are designed to be data-driven methods to capture metabolic differences among multiple optima. These methods do not require the assumption of prior knowledge of the strain. The optimal solution selected is controlled by the parameters of the methods. For WPDM, the parameters are the interior-point weights for each constraint. For ERM, the parameters are the reflection direction. Since these methods are data-driven, the subtle differences in the genome for the same species are captured by the manipulation of fitting parameters. By minimizing the sum squared errors between the model and experimental data, optimal fitting parameters can be found by optimization. Thus, our methodology is particularly suitable for describing engineered strains such as the ones used in the industry.

3. The uniqueness of the optimal solution by WPDM and ERM is mathematically guaranteed provided that the feasible space is not empty. To the author's knowledge,

earlier methods reported for tackling the multiplicity of DFBM cannot guarantee uniqueness. Because the objective functions of WPDM and ERM are designed to be strictly convex functions, the solution is shown to be unique. The logarithmic functions in WPDM and the positive definite matrix in the quadratic objectives play an important role in the uniqueness of the optimal solution.

4. Before the introduction of WPDM and ERM into DFBM, it is noticed that the fluxes solved by the Simplex method between two consecutive time steps can be drastically different even though the differences between values of concentrations (states) may be very small. The reason behind this is that the pivoting sequence of the Simplex methods along the vertices is not fixed and thus the differences in resulting fluxes obtained for two consecutive time steps can be large. To address this issue, the continuity property was explicitly considered for our proposed WPDM and ERM methods so that small changes in concentrations of metabolites will only result in correspondingly small changes in the fluxes. The continuity property is mathematically proven for both algorithms.

8.1.2 Methods of Set Membership Estimations

Two methods for DFBM model-based estimation are proposed. Both methods are based on a set membership estimation approach. The first set membership estimation method, presented in Chapter 4, is based on the assumption that DFBM always has a unique solution for each time step. The findings of the first set membership estimation method can be summarized as follows:

1. The governing equations of DFBM have two parts: state equations describing mass balances and a linear programming (LP) optimization. The LP needs to needs to be solved at each time step to calculate the fluxes and which are then used to calculate the metabolites' concentrations (states) at the next time step. Solving the LP at each time step is challenging for real-time estimation due to computational expense. One important finding of this research is that DFBM can be re-formulated by a multiparametric linear programming approach as a variable structure system thus eliminating the need to solve the LP in real-time at each time step.

   By using multiparametric LP, the original DFBM system is split into a series of subsystems, each corresponding to specific active constraints of the original LP. Each subsystem has its own explicit state equation that can be rapidly solved without

optimization. This is particularly important for real-time applications, such as state estimation and optimal control.

2. The identification of subsystems (or critical regions) provides an additional biological understanding of the occurrence of different metabolic patterns. For example, the active constraints can serve to identify the key metabolites that limit the growth of cells under different conditions. Different subsystems represent different ways in which cells relocate resources when different metabolites available are limited. Moreover, the critical regions may serve to identify metabolic switches, optimal media design for particular modes of operation (Batch/fed-batch), and optimal feeding strategies.

3. Lack of informative online measurements introduces a major observability challenge for model estimation of bio-processes. When using DFBM for model estimation, biomass is found to be the most informative state because it appears in all mass balances multiplying specific consumption/production rates of metabolites. Thus, for each subsystem (critical region) identified by the multiparametric LP approach, biomass growth provides information on key limiting resources consumed. By examining of increase in biomass, the dynamic of other correlating metabolites can be easily estimated for a particular subsystem. In this research, the construction of EKFs is based on the idea that biomass is the most informative state. Once the biomass is measured, other states can be estimated easily as shown in Chapter 3.

4. It is shown that multiple EKFs can be designed for the variable structure system. Each subsystem (critical region) obtained from the multi-parametric LP needs one EKF. A monitoring algorithm is designed to identify the state switch from one subsystem to another subsystem.

The second set membership estimation method in Chapter 5 is a general method and explicitly addresses the multiplicity of solutions. The findings of this method can be summarized as follows:

1. When the multiplicity of the linear programming is considered, the DFBM solution is based on the combination of a nonlinear programming (WPDM) approach and the state equations. In this case, multiparametric nonlinear programming is applied to the nonlinear programming (WPDM) formulation to convert the original system into a variable structure system.

2. Multiparametric nonlinear programming is known to be computationally expensive since the entire parameter space needs to be divided into zones that are sufficiently

small to obtain accurate solutions. As a result of this partition into small zones, the number of divisions required is huge which greatly increases the computation. On the other hand, the objective of this research is to estimate the state of industrial bioreactors, which usually are operated within a narrow operating region. Hence, several zones of the parameter space are irrelevant to state estimation. Then, a pragmatic approach was followed where irrelevant parameter zones were pruned thus reducing the computations.

3. Direct application of multiparametric nonlinear programming in our problem resulted in numerical issues that are related to the sensitivity of the logarithmic function in the barrier term of the interior point cost as the solution approaches the boundary of the constraint. An approximation of WPDM was constructed to bypass the numerical issues.

8.1.3 Setting up A Platform for Culturing *B. pertussis*

The contribution of this part research can be summarized as follows:

1. An experimental platform to cultivate the *B. pertussis* was developed to validate the ideas of model fitting and set membership estimation proposed in the research. Considering that *B. pertussis* is pathogenic, the platform needs to be carefully designed and developed for future applications. This involved shake flask experiments, culturing in the bioreactor, incubation of plates to corroborate the purity of the culture and measurement protocols of biomass and amino acid concentrations in the culture.

2. Comparison of the bioreactor experiments of *B. pertussis* in Sanofi and our lab, showed significant differences in the evolution of the culture. From the examination of the protocols of culturing and measurements and discussion with engineers from Sanofi, possible reasons for the observed differences were identified. Culture conditions are the most possible reasons, including different time delays in the inoculation to the bioreactor after harvesting from the shake flask, different control strategies of DO control, antifoam feeding policy, and the initial glutamate concentration.

8.2 Future Work

During my research, several possible future topics are identified. Computational efficiency is a must for real-time applications of the proposed set estimation algorithms.

1. The current WPDM algorithm is not fast enough and the results are not stable because of matrix invertibility problems. On the other hand, commercial interior-point solvers deal efficiently with matrix inversions. Since WPDM is also a type of interior-point method, there is a potential to modify commercial solvers to tackle WPDM. Although we proposed an alternative ERM method as an alternative of WPDM for fast model fitting, WPDM is still needed for the set estimation approach in the presence of multiplicity.

2. Current mpNLP is very slow and computationally expensive for large problems. Applying mpNLP in DFBM with large metabolic networks is still prohibitive. The development of a fast mpNLP is important not only for WPDM but also for other potential applications such as nonlinear control problems.

3. Alternatively, instead of the development of mpNLP that is based on WPDM, it may be possible to construct an mpQP approach based on the ERM algorithm that is solved by a combination of commercial LP and QP solvers. The mpQP combined ERM can be used to replace mpNLP with WPDM in set membership estimation. The number of critical regions of mpQP with ERM is expected to be much lower if mpNLP with WPDM is used. This will greatly reduce the computational expense thus making it possible to tackle large metabolic networks like the ones describing mammalian cells.

4. Set membership estimation should be tested by experimentally culturing of *B. pertussis* of bioreactors.

5. Economic model predictive control (EMPC) was a key initial research goal of the current research. However, the lack of sensors motivated to shift the focus of the work to model-based estimation. Therefore, methods of set membership estimation were developed. Future research should be conducted to develop an EMPC based on DFBM and the proposed set membership estimation techniques.

6. The current experimental platform is limited to testing different control strategies. These limitations were related to the available hardware and budget. In the future, software like LabView can be installed to control and automatically sample the bioreactor. The SME developed can be deployed on LabView to evaluate the performance of online state estimation algorithms.

7. DFBM model with SME can be combined with different spectroscopy like near-infrared spectroscopy, fluorescence spectroscopy, and Raman spectroscopy to estimate the metabolite concentrations. Since spectroscopy can provide more informa-

tion, the addition of sensors has the potential to improve the estimation performance. For example, our group is currently testing an online spectro-fluorometer that is able to provide online estimates of glutamate, biomass, and antigen concentrations. The addition of such a sensor in the bioreactor will permit online estimation of limiting nutrients and maximization of the amount of final product (antigen).

References

[1] Ilan Adler and Renato DC Monteiro. Limiting behavior of the affine scaling continuous trajectories for linear programming problems. *Mathematical Programming*, 50(1-3):29–51, 1991.

[2] Amir Akbari and Paul I Barton. An improved multi-parametric programming algorithm for flux balance analysis of metabolic networks. *Journal of Optimization Theory and Applications*, 178(2):502–537, 2018.

[3] Teodoro Alamo, José Manuel Bravo, and Eduardo F Camacho. Guaranteed state estimation by zonotopes. *Automatica*, 41(6):1035–1043, 2005.

[4] Jarinah Mohd Ali, N Ha Hoang, Mohamed Azlan Hussain, and Denis Dochain. Review and classification of recent observers applied in chemical process systems. *Computers & Chemical Engineering*, 76:27–41, 2015.

[5] Vincent Andrieu, Gildas Besançon, and Ulysse Serres. Observability necessary conditions for the existence of observers. In *52nd IEEE Conference on Decision and Control*, pages 4442–4447. IEEE, 2013.

[6] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[7] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[8] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[9] Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*, volume 78. Springer, 2008.

[10] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. Geometric algorithm for multiparametric linear programming. *Journal of optimization theory and applications*, 118(3):515–540, 2003.

[11] Zdravko I Botev. The normal law under linear restrictions: simulation and estimation via minimax tilting. *arXiv preprint arXiv:1603.04166*, 2016.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Hector Budman, Nilesh Patel, Melih Tamer, and Walid Al-Gherwi. A dynamic metabolic flux balance based model of fed-batch fermentation of bordetella pertussis. *Biotechnology progress*, 29(2):520–531, 2013.

[14] Anthony P Burgard, Priti Pharkya, and Costas D Maranas. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and bioengineering*, 84(6):647–657, 2003.

[15] CI Byrnes and CF Martin. Global observability and detectability: An overview. In *Modelling and Adaptive Control: Proceedings of the IIASA Conference Sopron, Hungary, July 1986*, pages 71–89. Springer, 2006.

[16] Albert E Cervera, Nanna Petersen, Anna Eliasson Lantz, Anders Larsen, and Krist V Gernaey. Application of near-infrared spectroscopy for monitoring and control of cell culture and fermentation. *Biotechnology progress*, 25(6):1561–1581, 2009.

[17] Luigi Chisci, Andrea Garulli, and Giovanni Zappa. Recursive state bounding by parallelotopes. *Automatica*, 32(7):1049–1055, 1996.

[18] Anupam Chowdhury, Ali R Zomorrodi, and Costas D Maranas. k-optforce: integrating kinetics with flux balance analysis for strain design. *PLoS computational biology*, 10(2):e1003487, 2014.

[19] Steven A Cohen. Amino acid analysis using precolumn derivatization with 6-aminoquinolyl-n-hydroxysuccinimidyl carbamate. *Amino acid analysis protocols*, pages 39–47, 2000.

[20] Zsolt Darvay. A weighted-path-following method for linear optimization. *Studia Universitatis Babes-Bolyai, Series Informatica*, 47(1):3–12, 2002.

[21] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[22] Jean-Pierre Dedieu and Mike Shub. Newton flow and interior point methods in linear programming. *International Journal of Bifurcation and Chaos*, 15(03):827–839, 2005.

[23] Vincent Delos and Denis Teissandier. Minkowski sum of hv-polytopes in rn. *arXiv preprint arXiv:1412.2562*, 2014.

[24] Maarten R Dobbelaere, Pieter P Plehiers, Ruben Van de Vijver, Christian V Stevens, and Kevin M Van Geem. Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats. *Engineering*, 7(9):1201–1211, 2021.

[25] Denis Dochain. State and parameter estimation in chemical and biochemical processes: a tutorial. *Journal of process control*, 13(8):801–818, 2003.

[26] Luis F Domínguez, Diogo A Narciso, and Efstratios N Pistikopoulos. Recent advances in multiparametric nonlinear programming. *Computers & Chemical Engineering*, 34(5):707–716, 2010.

[27] Jeremy S Edwards and Bernhard O Palsson. Metabolic flux balance analysis and the in silico analysis of escherichia coli k-12 gene deletions. *BMC bioinformatics*, 1(1):1, 2000.

[28] Denis Efimov, Wilfrid Perruquetti, Tarek Raïssi, and Ali Zolghadri. On interval observer design for time-invariant discrete-time systems. In *2013 European Control Conference (ECC)*, pages 2651–2656. IEEE, 2013.

[29] Michael C Ferris, Olvi L Mangasarian, and Stephen J Wright. *Linear programming with MATLAB*, volume 7. SIAM, 2007.

[30] Anthony V Fiacco and Garth P McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*, volume 4. Siam, 1990.

[31] LH Field and CD Parker. Differences observed between fresh isolates of bordetella pertussis and their laboratory-passaged derivatives. In *International Symposium on Pertussis. US Department of Health, Education, and Welfare, Washington, DC*, pages 124–132, 1979.

[32] William Ford. *Numerical linear algebra with applications: Using MATLAB*. Academic Press, 2014.

[33] Carlos Eduardo García Sánchez and Rodrigo Gonzalo Torres Sáez. Comparison and analysis of objective functions in flux balance analysis. *Biotechnology progress*, 30(5):985–991, 2014.

[34] Atefeh Ghorbaniaghdam, Jingkui Chen, Olivier Henry, and Mario Jolicoeur. Analyzing clonal variation of monoclonal antibody-producing cho cell lines using an in silico metabolomic platform. *PloS one*, 9(3):e90832, 2014.

[35] Jose A Gomez, Kai Höffner, and Paul I Barton. Dfbalab: a fast and reliable matlab code for dynamic flux balance analysis. *BMC bioinformatics*, 15(1):1–10, 2014.

[36] Jose A Gomez, Kai Höffner, and Paul I Barton. From sugars to biodiesel using microalgae and yeast. *Green Chemistry*, 18(2):461–475, 2016.

[37] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[38] Jean-Luc Gouzé, Alain Rapaport, and Mohamed Zakaria Hadj-Sadok. Interval observers for uncertain biological systems. *Ecological modelling*, 133(1-2):45–56, 2000.

[39] Alexandra Grancharova and Tor Arne Johansen. *Explicit nonlinear model predictive control: Theory and applications*, volume 429. Springer Science & Business Media, 2012.

[40] Osman Güler. Limiting behavior of weighted central paths in linear programming. *Mathematical Programming*, 65(1):347–363, 1994.

[41] Henri Haimi, Michela Mulas, Francesco Corona, and Riku Vahala. Data-derived soft-sensors for biological wastewater treatment plants: An overview. *Environmental Modelling & Software*, 47:88–107, 2013.

[42] Stuart M Harwood, Kai Höffner, and Paul I Barton. Efficient solution of ordinary differential equations with a parametric lexicographic linear program embedded. *Numerische Mathematik*, 133(4):623–653, 2016.

[43] Martin Herceg, Michal Kvasnica, Colin N Jones, and Manfred Morari. Multi-parametric toolbox 3.0. In *2013 European control conference (ECC)*, pages 502–510. IEEE, 2013.

[44] Rubin Hille, Jasdeep Mandur, and Hector M Budman. Robust batch-to-batch optimization in the presence of model-plant mismatch and input uncertainty. *AIChE Journal*, 63(7):2660–2670, 2017.

[45] Jared L Hjersted, Michael A Henson, and Radhakrishnan Mahadevan. Genome-scale analysis of saccharomyces cerevisiae metabolism and ethanol production in fed-batch culture. *Biotechnology and bioengineering*, 97(5):1190–1204, 2007.

[46] Kai Höffner, Stuart M Harwood, and Paul I Barton. A reliable simulator for dynamic flux balance analysis. *Biotechnology and bioengineering*, 110(3):792–802, 2013.

[47] Hermann Georg Holzhütter. The principle of flux minimization and its application to estimate stationary fluxes in metabolic networks. *European journal of biochemistry*, 271(14):2905–2922, 2004.

[48] IBM. *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*. IBM ILOG CPLEX Division, 2016.

[49] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Eric Walter. Interval analysis. In *Applied interval analysis*, pages 11–43. Springer, 2001.

[50] Tor A Johansen. On multi-parametric nonlinear programming and explicit nonlinear model predictive control. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 2768–2773. IEEE, 2002.

[51] Colin N Jones, Eric C Kerrigan, and Jan M Maciejowski. Lexicographic perturbation for multiparametric linear programming with applications to control. *Automatica*, 43(10):1808–1816, 2007.

[52] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. Data-driven soft sensors in the process industry. *Computers & chemical engineering*, 33(4):795–814, 2009.

[53] Alex Kalos, Arthur Kordon, Guido Smits, and Sofka Werkmeister. Hybrid model development methodology for industrial soft sensors. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 6, pages 5417–5422. IEEE, 2003.

[54] Pezhman Kazemi, Jean-Philippe Steyer, Christophe Bengoa, Josep Font, and Jaume Giralt. Robust data-driven soft sensors for online monitoring of volatile fatty acids in anaerobic digestion processes. *Processes*, 8(1):67, 2020.

[55] Steven M Kelk, Brett G Olivier, Leen Stougie, and Frank J Bruggeman. Optimal flux spaces of genome-scale stoichiometric models are determined by a few subnetworks. *Scientific reports*, 2:580, 2012.

[56] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[57] Masakazu Kojima, Nimrod Megiddo, and Shinji Mizuno. A primal—dual infeasible-interior-point algorithm for linear programming. *Mathematical programming*, 61(1-3):263–280, 1993.

[58] Leslie Lamport. *LaTeX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[59] Sangbum Lee, Chan Phalakornkule, Michael M Domach, and Ignacio E Grossmann. Recursive milp model for finding all the alternate optima in lp models for metabolic networks. *Computers & Chemical Engineering*, 24(2-7):711–716, 2000.

[60] Nathan E Lewis, Kim K Hixson, Tom M Conrad, Joshua A Lerman, Pep Charusanti, Ashoka D Polpitiya, Joshua N Adkins, Gunnar Schramm, Samuel O Purvine, Daniel Lopez-Ferrer, et al. Omic data from evolved e. coli are consistent with computed optimal growth from genome-scale models. *Molecular systems biology*, 6(1):390, 2010.

[61] Henry C Lim and Hwa Sung Shin. *Fed-batch cultures: principles and applications of semi-batch bioreactors*. Cambridge University Press, 2013.

[62] Andreas Löhne and Benjamin Weißing. The vector linear program solver bensolve–notes on theoretical background. *European Journal of Operational Research*, 260(3):807–813, 2017.

[63] R Mahadevan and CH Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic engineering*, 5(4):264–276, 2003.

[64] Radhakrishnan Mahadevan, Jeremy S Edwards, and Francis J Doyle III. Dynamic flux balance analysis of diauxic growth in escherichia coli. *Biophysical journal*, 83(3):1331–1340, 2002.

[65] Kyoko Makino and Martin Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 6:239–316, 2003.

[66] DG Maksarov and JP Norton. Computationally efficient algorithms for state estimation with ellipsoidal approximations. *International Journal of Adaptive Control and Signal Processing*, 16(6):411–434, 2002.

[67] Olvi L Mangasarian and RR Meyer. Nonlinear perturbation of linear programs. *SIAM Journal on Control and Optimization*, 17(6):745–752, 1979.

[68] Costas D Maranas and Ali R Zomorrodi. *Optimization methods in metabolic networks*. John Wiley & Sons, 2016.

[69] Ricardo Martinez Villegas, Hector Budman, and Ali Elkamel. Identification of dynamic metabolic flux balance models based on parametric sensitivity analysis. *Industrial & Engineering Chemistry Research*, 56(8):1911–1919, 2017.

[70] Frédéric Mazenc, Thach Ngoc Dinh, and Silviu-Iulian Niculescu. Robust interval observers and stabilization design for discrete-time systems with input and output. *Automatica*, 49(11):3490–3497, 2013.

[71] Adam L Meadows, Rahi Karnik, Harry Lam, Sean Forestell, and Brad Snedecor. Application of dynamic flux balance analysis to an industrial escherichia coli fermentation. *Metabolic engineering*, 12(2):150–160, 2010.

[72] Nimrod Megiddo. Pathways to the optimal set in linear programming. In *Progress in mathematical programming*, pages 131–158. Springer, 1989.

[73] Yanmei Meng, Qiliang Lan, Johnny Qin, Shuangshuang Yu, Haifeng Pang, and Kangyuan Zheng. Data-driven soft sensor modeling based on twin support vector regression for cane sugar crystallization. *Journal of food engineering*, 241:159–165, 2019.

[74] H Ronald Miller. *Optimization: Foundations and applications*. John Wiley & Sons, 2011.

[75] Jérôme Morchain, Jean-Christophe Gabelle, and Arnaud Cockx. A coupled population balance model and cfd approach for the simulation of mixing issues in lab-scale and industrial bioreactors. *AIChE Journal*, 60(1):27–40, 2014.

[76] Ehsan Motamedian. A new algorithm to find all alternate optimal flux distributions of a metabolic network. *Computers & Chemical Engineering*, 73:64–69, 2015.

[77] Ehsan Motamedian and Fereshteh Naeimpoor. Lamos: A linear algorithm to identify the origin of multiple optimal flux distributions in metabolic networks. *Computers & Chemical Engineering*, 117:372–377, 2018.

[78] Ettore Murabito, Evangelos Simeonidis, Kieran Smallbone, and Jonathan Swinton. Capturing the essence of a metabolic network: a flux balance analysis approach. *Journal of theoretical biology*, 260(3):445–452, 2009.

[79] Mari M Nakamura, Sin-Yee Liew, Craig A Cummings, Mary M Brinig, Christine Dieterich, and David A Relman. Growth phase-and nutrient limitation-associated transcript abundance regulation in bordetella pertussis. *Infection and immunity*, 74(10):5537–5548, 2006.

[80] Komi Nambou, Xingxing Jian, Xinkai Zhang, Liujing Wei, Jiajia Lou, Catherine Madzak, and Qiang Hua. Flux balance analysis inspired bioprocess upgrading for lycopene production by a metabolically engineered strain of yarrowia lipolytica. *Metabolites*, 5(4):794–813, 2015.

[81] Ali Nikdel, Richard D Braatz, and Hector M Budman. A systematic approach for finding the objective function and active constraints for dynamic flux balance analysis. *Bioprocess and biosystems engineering*, 41(5):641–655, 2018.

[82] Ali Nikdel and Hector Budman. Identification of active constraints in dynamic flux balance analysis. *Biotechnology progress*, 33(1):26–36, 2017.

[83] Elad Noor, Avi Flamholz, Arren Bar-Even, Dan Davidi, Ron Milo, and Wolfram Liebermeister. The protein cost of metabolic fluxes: prediction from enzymatic rate laws and cost minimization. *PLoS computational biology*, 12(11):e1005167, 2016.

[84] Richard Oberdieck, Nikolaos A Diangelakis, Ioana Nascu, Maria M Papathanasiou, Muxin Sun, Styliani Avraamidou, and Efstratios N Pistikopoulos. On multi-parametric programming and its applications in process systems engineering. *Chemical engineering research and design*, 116:61–82, 2016.

[85] Kaveh Ohadi, Raymond L Legge, and Hector M Budman. Intrinsic fluorescence-based at situ soft sensor for monitoring monoclonal antibody aggregation. *Biotechnology progress*, 31(5):1423–1432, 2015.

[86] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245, 2010.

[87] Anna Prugnola, Beatrice Aricò, Riccardo Manetti, Rino Rappuoli, and Vincenzo Scarlato. Response of the bvg regulon of bordetella pertussis to different temperatures and short-term temperature shifts. *Microbiology*, 141(10):2529–2534, 1995.

[88] Klaus Röbenack and Kurt J Reinschke. An efficient method to compute lie derivatives and the observability matrix for nonlinear systems. In *Proc. Int. Symposium on Nonlinear Theory and its Applications (NOLTA)*, volume 2, pages 625–628, 2000.

[89] Philipp Rumschinski, Steffen Borchers, Sandro Bosio, Robert Weismantel, and Rolf Findeisen. Set-base dynamical parameter estimation and model invalidation for biochemical reaction networks. *BMC systems biology*, 4(1):1–14, 2010.

[90] Bryan Rynne and Martin A Youngson. *Linear functional analysis*. Springer Science & Business Media, 2007.

[91] Ali Mohammad Sahlodin and Benoit Chachuat. Convex/concave relaxations of parametric odes using taylor models. *Computers & Chemical Engineering*, 35(5):844–857, 2011.

[92] Sanofi. Report on 2000-l fermentor in building 89 (b89) 2-l small-scale model development, Feb 2019.

[93] Robert Schuetz, Nicola Zamboni, Mattia Zampieri, Matthias Heinemann, and Uwe Sauer. Multidimensional optimality of microbial metabolism. *Science*, 336(6081):601–604, 2012.

[94] Karl Schügerl and K-H Bellgardt. *Bioreaction engineering: modeling and control*. Springer Science & Business Media, 2012.

[95] Fred Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.

[96] Onur Şeref, J Paul Brooks, Bernice Huang, and Stephen S Fong. Enumeration and cartesian product decomposition of alternate optimal fluxes in cellular metabolism. *INFORMS Journal on Computing*, 29(2):197–210, 2017.

[97] Chao Shang, Fan Yang, Dexian Huang, and Wenxiang Lyu. Data-driven soft sensor development based on deep learning technique. *Journal of Process Control*, 24(3):223–233, 2014.

[98] Xin Shen and Hector Budman. A method for tackling primal multiplicity of solutions of dynamic flux balance models. *Computers & Chemical Engineering*, 143:107070, 2020.

[99] Xin Shen and Hector Budman. Online estimation using dynamic flux balance model and multiparametric programming. *Computers & Chemical Engineering*, page 107872, 2022.

[100] Hanif D Sherali. Equivalent weights for lexicographic multi-objective programs: Characterizations and computations. *European Journal of Operational Research*, 11(4):367–379, 1982.

[101] George F Simmons. *Introduction to topology and modern analysis*, volume 44. Tokyo, 1963.

[102] Kieran Smallbone and Evangelos Simeonidis. Flux balance analysis: a geometric perspective. *Journal of theoretical biology*, 258(2):311–315, 2009.

[103] Yongkyu Song and Jessy W Grizzle. The extended kalman filter as a local asymptotic observer for nonlinear discrete-time systems. In *1992 American control conference*, pages 3365–3369. IEEE, 1992.

[104] J Spjøtvold, P Tøndel, and TA Johansen. Continuous selection and unique polyhedral representation of solutions to convex parametric quadratic programs. *Journal of Optimization Theory and Applications*, 134(2):177–189, 2007.

[105] Peter F Stanbury, Allan Whitaker, and Stephen J Hall. *Principles of fermentation technology*. Elsevier, 2013.

[106] Mahshad Valipour and Luis A Ricardez-Sandoval. A robust moving horizon estimation under unknown distributions of process or measurement noises. *Computers & Chemical Engineering*, 157:107620, 2022.

[107] Adriaan Van den Bos. *Parameter estimation for scientists and engineers*. John Wiley & Sons, 2007.

[108] Amit Varma and Bernhard O Palsson. Metabolic flux balancing: basic concepts, scientific and practical use. *Bio/technology*, 12(10):994, 1994.

[109] Amit Varma and Bernhard O Palsson. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type escherichia coli w3110. *Applied and environmental microbiology*, 60(10):3724–3731, 1994.

[110] Alejandro F Villaverde et al. Observability and structural identifiability of nonlinear biological systems. *Complexity*, 2019, 2019.

[111] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[112] Stefan Wilhelm and BG Manjunath. tmvtnorm: A package for the truncated multivariate normal distribution. *sigma*, 2(2):1–25, 2010.

[113] Eliana Zamprogna, Massimiliano Barolo, and Dale E Seborg. Optimal selection of soft sensor inputs for batch distillation columns using principal component analysis. *Journal of process control*, 15(1):39–52, 2005.

[114] Jürgen Zanghellini, David E Ruckerbauer, Michael Hanscho, and Christian Jungreuthmayer. Elementary flux modes in a nutshell: properties, calculation and applications. *Biotechnology journal*, 8(9):1009–1016, 2013.

[115] Qi Zhao, Arion Stettner, Ed Reznik, Daniel Segrè, and Ioannis Ch Paschalidis. Learning cellular objectives from fluxes by inverse optimization. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 1271–1276. IEEE, 2015.

[116] Ognyan Ivanov Zhelezov. N-dimensional rotation matrix generation algorithm. *American Journal of Computational and Applied Mathematics*, 7(2):51–57, 2017.

Appendices

Appendix A

Matlab Codes and Proof

A.1 Proof Related to Weighted Primal-Dual Method

Lemma 2 Let X and Y be normed linear spaces and let $T : X \to Y$ be a linear tansformation. T is continuous if there exists a positive real number r such that $\|T(x)\| \leq r\|x\|$ for all $x \in X$ [90].

Theorem 6 Assume the problem of Eq.(3.6) is not empty, the optimal solution is locally continuous with respect to $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{w}$.

Proof:

Define a function $F(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{z}) : \mathbb{R}^{n_\lambda} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_\lambda} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z}$ as Eq.(A.1).

$$F(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} \boldsymbol{A^T\lambda} + \boldsymbol{c} \\ \boldsymbol{Ax} + \boldsymbol{z} - \boldsymbol{b} \\ \boldsymbol{\Lambda z e} - \mu \boldsymbol{w} \end{bmatrix} \tag{A.1}$$

Let $\boldsymbol{b} = \boldsymbol{b_0}, \boldsymbol{c} = \boldsymbol{c_0}, \boldsymbol{w} = \boldsymbol{w_0}$, let assume the unique solution to Eq.(A.1) exists and can be expressed as $[\boldsymbol{\lambda_0}, \boldsymbol{x_0}, \boldsymbol{z_0}]^{\boldsymbol{T}}$. Therefore, we have Eq.(A.2)

$$F(\boldsymbol{\lambda_0}, \boldsymbol{x_0}, \boldsymbol{z_0}) = \boldsymbol{0} \tag{A.2}$$

Apply Taylor expansion to $F(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{z})$ at the optimal solution $[\boldsymbol{\lambda_0}, \boldsymbol{x_0}, \boldsymbol{z_0}]^{\boldsymbol{T}}$:

$$F(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{z}) = F(\boldsymbol{\lambda_0}, \boldsymbol{x_0}, \boldsymbol{z_0}) + J(F) \begin{bmatrix} \boldsymbol{\Delta\lambda} \\ \boldsymbol{\Delta x} \\ \boldsymbol{\Delta z} \end{bmatrix} = 0 \tag{A.3a}$$

$$\begin{bmatrix} \boldsymbol{\Delta\lambda} \\ \boldsymbol{\Delta x} \\ \boldsymbol{\Delta z} \end{bmatrix} = -J^{-1}(F) F(\boldsymbol{\lambda_0}, \boldsymbol{x_0}, \boldsymbol{z_0}) \tag{A.3b}$$

where $J(\cdot)$ is Jacobian matrix, $\boldsymbol{\Delta\lambda}, \boldsymbol{\Delta x}, \boldsymbol{\Delta z}$ are deviations from the optimal solution. Substitute parameters into Eq.(A.3):

$$\begin{bmatrix} \boldsymbol{\Delta\lambda} \\ \boldsymbol{\Delta x} \\ \boldsymbol{\Delta z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A^T} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} & \boldsymbol{I} \\ \boldsymbol{Z_0} & \boldsymbol{0} & \boldsymbol{\Lambda_0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{-c_0 - A^T\lambda} \\ \boldsymbol{b_0 - z_0 - Ax_0} \\ \boldsymbol{\mu w_0 - \Lambda_0 z_0 e} \end{bmatrix} \tag{A.4}$$

Note matrix A is an extended matrix of the original linear programming problem that includes columns of slack variables. So that full rank of matrix A is assured and the Jacobian matrix exists at the optimal solution and is nonsingular whenever $\boldsymbol{\lambda_0} > \boldsymbol{0}$ and $\boldsymbol{z_0} > \boldsymbol{0}$. Define the function G: $\mathbb{R}^{n_\lambda} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_\lambda} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_z}$ as Eq.(A.5) mapping the perturbation vector space into the deviation vector space.

$$G(\boldsymbol{\epsilon}) = \begin{bmatrix} \boldsymbol{A^T} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} & \boldsymbol{I} \\ \boldsymbol{Z_0} & \boldsymbol{0} & \boldsymbol{\Lambda_0} \end{bmatrix}^{-1} \boldsymbol{\epsilon} \tag{A.5}$$

Let $\boldsymbol{K}$ be the metric-induced norm of the inverse of the Jacobian matrix.

$$\boldsymbol{K} = \left\| \begin{bmatrix} \boldsymbol{A^T} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} & \boldsymbol{I} \\ \boldsymbol{Z_0} & \boldsymbol{0} & \boldsymbol{\Lambda_0} \end{bmatrix}^{-1} \right\| \tag{A.6}$$

For any perturbation $\boldsymbol{\epsilon}$ to $[\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{w}]^T$, the deviation of the optimal solution is bounded as

per Eq.(A.7)

$$\left\| \begin{bmatrix} \boldsymbol{\Delta\lambda} \\ \boldsymbol{\Delta x} \\ \boldsymbol{\Delta z} \end{bmatrix} \right\| = \left\| G(\boldsymbol{\epsilon}) \right\| = \left\| \begin{bmatrix} \boldsymbol{A^T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A} & \boldsymbol{I} \\ \boldsymbol{Z_0} & \mathbf{0} & \boldsymbol{\Lambda_0} \end{bmatrix}^{-1} \right\| \|\boldsymbol{\epsilon}\| \leq \boldsymbol{K} \|\boldsymbol{\epsilon}\| \tag{A.7}$$

Therefore, the function G is locally continuous at the optimal solution $[\boldsymbol{\lambda_0}, \boldsymbol{x_0}, \boldsymbol{z_0}]^{\boldsymbol{T}}$.

A.2 Example of WPDM

To explain the reason of rank deficient Hessian matrix clearly, an example of LP with multiplicity is presented here as Eq. (A.8). v_1 and v_2 are decision variables. Multiple optimal solutions exist on the hyperplane defined by constraint $2v_1 + v_2 = 8$ from points $(4, 0)$ to $(3, 2)$. If simplex solver is used, either vertex $(4, 0)$ or $(3, 2)$ can be obtained. Any solution between $(4, 0)$ to $(3, 2)$ is also optimal solution but cannot be obtained by simplex solver. And vertex $(4, 0)$ or $(3, 2)$ correspond to different active constraints and inactive constraints. If $(4, 0)$ is obtained by the simplex solver, the active constraints are $2v_1 + v_2 = 8$, $v_1 = 4$ and $v_2 = 0$. Since only two of three active constraints can determine the unique optimal solution, one of them is redundant and can be removed by Gaussian elimination. If $(3, 2)$ is obtained by the simplex solver, the active constraints are $v_1+3v_2 = 9$ and $2v_1 + v_2 = 8$. Two active constraints determine the unique optimal solution. If the problem has parameter $\boldsymbol{\theta}$ in constraints, different active constraints determine different critical regions. This is the reason that multiplicity issue influences the determination of critical region and WPDM can avoid the problem.

$$\min_{\boldsymbol{v_k}} \quad -2v_1 - v_2 \tag{A.8a}$$

$$\text{subject to} \quad v_1 + 3v_2 \leq 9 \tag{A.8b}$$

$$2v_1 + v_2 \leq 8 \tag{A.8c}$$

$$v_1 \leq 4 \tag{A.8d}$$

$$-v_1 \leq 0 \tag{A.8e}$$

$$-v_2 \leq 0 \tag{A.8f}$$

The corresponding WPDM with slack variables $\boldsymbol{z}$ is defined as Eq. (A.9).

$$\inf_{\mu\to 0,\boldsymbol{v},\boldsymbol{z}} \quad -2v_1 - v_2 - \mu(w_1 ln(z_1) + w_2 ln(z_2) + w_3 ln(z_3) + w_4 ln(z_4) + w_5 ln(z_5)) \tag{A.9a}$$

$$\text{subject to} \quad v_1 + 3v_2 + z_1 = 9 \tag{A.9b}$$

$$2v_1 + v_2 + z_2 = 8 \tag{A.9c}$$

$$v_1 + z_3 = 4 \tag{A.9d}$$

$$-v_1 + z_4 = 0 \tag{A.9e}$$

$$-v_2 + z_5 = 0 \tag{A.9f}$$

If different interior-point weights $\boldsymbol{w}$ are used, different optimal solutions between line section from $(4, 0)$ to $(3, 2)$ can be obtained through WPDM because these alternative optimal solutions are interior points of the hyperplane $2v_1 + v_2 = 8$. Points sufficiently close to vertexes can also be obtained with high accuracy once the μ is sufficiently small. For all optimal solutions between $(4, 0)$ to $(3, 2)$ (excluding vertexes), only one constraint $2v_1+v_2 = 8$ is active. While the simplex solver cannot find these optimal solutions, these points can be found by the interior point algorithm with different $\boldsymbol{w}$ values.

The strictly convex surfaces of WPDM for $\mu = 1$ and $\mu = 0.1$ are presented in the Fig. (A.1). And corresponding contours of the surfaces are plotted on the bottom. On the contour plot, constraints Eqs. (A.8b)-(A.8f) or Eqs. (A.9b)-(A.9f) define the feasible space. The red dots denote the optimal solution $\boldsymbol{v} = \begin{bmatrix}3.52 & 0.96\end{bmatrix}^T$ for given $\boldsymbol{w} = [2\ 1\ 1\ 1\ 1]^T$ as $\mu \to 0$ and notice point $(3.52, 0.96)$ is an interior point on the hyperplane $2v_1 + v_2 = 8$. The "zig-zag" boundaries of the convex surfaces are barrier functions defined by the constraints. Since barrier functions are abruptly increasing to infinity at the boundary of each constraint, the figure presents the abrupt increasing by "zig-zag" shapes, which is also the reason of illness of interior-point methods.

When $\mu = 1$, the objective convex surface is like a "bowl". As μ decreases to 0.1, the bowl-like surface become a super-linear plane but still strictly convex surface. When $\mu \to 0$, the convex surface will be sufficiently similar to the linear objective hyperplane $-2v_1 - v_2$ in Eq. (A.8a). The corresponding elliptical contours become super linear parallel contours because the gradient of objective hyperplane $-2v_1 - v_2$ is a constant covector $[-2\ -1]$. Notice the condition number is the ratio of semi-major axis to the semi-minor axis of the ellipse. When the condition number is very large, the rank of the Hessian matrix is rank deficient. When $\mu \to 0$, super linear contours will have infinitesimal semi-minor axis and rank deficient Hessian matrix. Therefore, local quadratic programming cannot

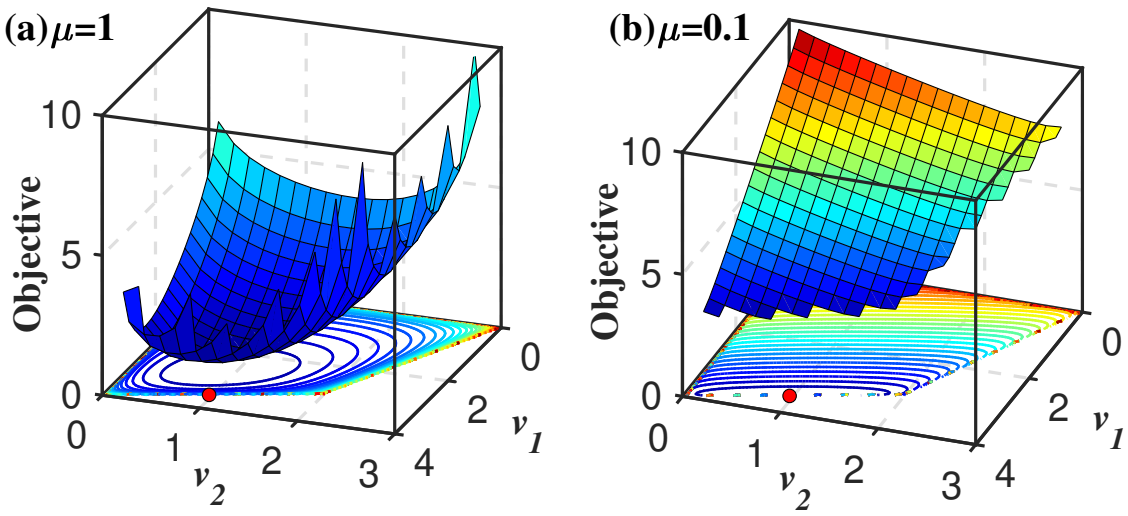


Figure A.1: Surfaces and contours of objective in WPDM as $\mu \to 0$

approximate WPDM well directly.

Back to the example of Eq. (A.9) above, since the active constraint is the second constraint, it can be approximated by problem P'_w defined in Eq. (A.10). The solution to Eq. (A.10) is exactly optimal solution $\boldsymbol{v} = \begin{bmatrix} 3.52 & 0.96 \end{bmatrix}^T$ for given $\boldsymbol{w}$.

$$\inf_{\mu \to 0, \boldsymbol{v}, \boldsymbol{z}} \quad -w_1 ln(z_1) - w_3 ln(z_3) - w_4 ln(z_4) - w_5 ln(z_5) \tag{A.10a}$$

$$\text{subject to} \quad v_1 + 3v_2 + z_1 = 9 \tag{A.10b}$$

$$2v_1 + v_2 = 8 \tag{A.10c}$$

$$v_1 + z_3 = 4 \tag{A.10d}$$

$$-v_1 + z_4 = 0 \tag{A.10e}$$

$$-v_2 + z_5 = 0 \tag{A.10f}$$

A.3 Proof related to ERM

Proof of theorem 5 continuity of ERM: As proved before, the QP problem defined in Eq. (6.13) is a strict convex optimization problem regardless of the reflection direction $\boldsymbol{r_1}$ is. Therefore, the KKT condition holds for this case and and the Lagrangian is given by

Eq. (A.11).

$$L(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{y}^T\boldsymbol{Q}\boldsymbol{y} + \boldsymbol{\lambda}^T(\boldsymbol{A}_{\mathcal{N}}\boldsymbol{y} - \boldsymbol{A}_{\mathcal{N}}\boldsymbol{d} - \boldsymbol{b}_{\mathcal{N}}) + \boldsymbol{\mu}^T(\hat{\boldsymbol{A}}\boldsymbol{y} - \hat{\boldsymbol{A}}\boldsymbol{d} - \hat{\boldsymbol{b}}) \tag{A.11}$$

The corresponding KKT condition is given by Eq. (A.12). Eqs.(A.12d)-(A.12f) is the complementary condition and i is the index of inequality constraint. If the Lagrangian multiplier $\lambda_i > 0$, the i-th inequality constraint is activated and $\boldsymbol{A}_{\mathcal{N}i}\boldsymbol{y} - \boldsymbol{A}_{\mathcal{N}i}\boldsymbol{d} - \boldsymbol{b}_{\mathcal{N}i} = \boldsymbol{0}$ holds (complementary slackness). $|\mathcal{N}|$ is the cardinality of set $\mathcal{N}$, i.e. the number of elements of set $\mathcal{N}$.

$$\boldsymbol{Q}\boldsymbol{y} + \boldsymbol{A}_{\mathcal{N}}^T\boldsymbol{\lambda} + \hat{\boldsymbol{A}}^T\boldsymbol{\mu} = \boldsymbol{0} \tag{A.12a}$$
$$\hat{\boldsymbol{A}}\boldsymbol{y} = \hat{\boldsymbol{A}}\boldsymbol{d} + \hat{\boldsymbol{b}} \tag{A.12b}$$
$$\boldsymbol{A}_{\mathcal{N}}\boldsymbol{y} \leq \boldsymbol{A}_{\mathcal{N}}\boldsymbol{d} + \boldsymbol{b}_{\mathcal{N}} \tag{A.12c}$$
$$\boldsymbol{\lambda} \geq \boldsymbol{0} \tag{A.12d}$$
$$\lambda_i(\boldsymbol{A}_{\mathcal{N}i}\boldsymbol{y} - \boldsymbol{A}_{\mathcal{N}i}\boldsymbol{d} - \boldsymbol{b}_{\mathcal{N}i}) = \boldsymbol{0} \tag{A.12e}$$
$$i = 1 \cdots |\mathcal{N}| \tag{A.12f}$$

Denote $\mathcal{B}$ the set of indices of all active inequality constraints defined by $\mathcal{B} = \{i \mid \lambda_i > 0\}$. If the active inequality constraints $\boldsymbol{A}_{\mathcal{N}\mathcal{B}}\boldsymbol{y} = \boldsymbol{A}_{\mathcal{N}\mathcal{B}}\boldsymbol{d} + \boldsymbol{b}_{\mathcal{N}\mathcal{B}}$ and equality constraints $\hat{\boldsymbol{A}}\boldsymbol{y} = \hat{\boldsymbol{A}}\boldsymbol{d} + \hat{\boldsymbol{b}}$ are collected together as in Eq. (A.13).

$$\boldsymbol{A}_{\boldsymbol{\eta}}\boldsymbol{y} = \boldsymbol{b}_{\boldsymbol{\eta}} \tag{A.13a}$$
$$\boldsymbol{A}_{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{N}\mathcal{B}} \\ \hat{\boldsymbol{A}} \end{bmatrix} \tag{A.13b}$$
$$\boldsymbol{b}_{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{N}\mathcal{B}}\boldsymbol{d} + \boldsymbol{b}_{\mathcal{N}\mathcal{B}} \\ \hat{\boldsymbol{A}}\boldsymbol{d} + \hat{\boldsymbol{b}} \end{bmatrix} \tag{A.13c}$$

Correspondingly, define $\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{\lambda}_{\mathcal{B}}^T & \boldsymbol{\mu}^T \end{bmatrix}^T$. Then, Eq. (A.12a) can be expressed as $\boldsymbol{Q}\boldsymbol{y} + \boldsymbol{A}_{\boldsymbol{\eta}}^T\boldsymbol{\eta} = \boldsymbol{0}$ resulting in Eq. (A.14).

$$\boldsymbol{R}\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\eta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b}_{\eta} \end{bmatrix} \tag{A.14a}$$

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}_{\boldsymbol{\eta}}^T \\ \boldsymbol{A}_{\boldsymbol{\eta}} & \boldsymbol{0} \end{bmatrix} \tag{A.14b}$$

If the linear independence constraint qualification (LICQ) condition is satisfied, $\boldsymbol{A}_{\boldsymbol{\eta}}$ is full row rank. Then, the matrix $\boldsymbol{R}$ is also full rank because matrix $\boldsymbol{Q}$ is invertible. Then,

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\eta} \end{bmatrix} = \boldsymbol{R}^{-1}\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b}_{\eta} \end{bmatrix} \tag{A.15}$$

If a pertubation $\boldsymbol{\epsilon}$ is added into the RHS of (A.15)

$$\begin{bmatrix} \boldsymbol{y}' \\ \boldsymbol{\eta}' \end{bmatrix} = \boldsymbol{R}^{-1}(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b}_{\eta} \end{bmatrix} + \boldsymbol{\epsilon}) = \boldsymbol{R}^{-1}\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b}_{\eta} \end{bmatrix} + \boldsymbol{R}^{-1}\boldsymbol{\epsilon} \tag{A.16}$$

Then,

$$\begin{bmatrix} \boldsymbol{\Delta y} \\ \boldsymbol{\Delta \eta} \end{bmatrix} = \boldsymbol{R}^{-1}\boldsymbol{\epsilon} \tag{A.17}$$

Let $\boldsymbol{K}$ be the metric-induced norm of the matrix $\boldsymbol{R}^{-1}$, as $\boldsymbol{K} = \left\|\boldsymbol{R}^{-1}\right\|$. For any perturbation $\boldsymbol{\epsilon}$ to $\hat{\boldsymbol{b}}$ and $\boldsymbol{b}_{\mathcal{N}}$, the deviation of the optimal solution is bounded as per Eq.(A.18)

$$\left\| \begin{bmatrix} \boldsymbol{\Delta y} \\ \boldsymbol{\Delta \eta} \end{bmatrix} \right\| = \left\|\boldsymbol{R}^{-1}\boldsymbol{\epsilon}\right\| \leq \left\|\boldsymbol{R}^{-1}\right\| \|\boldsymbol{\epsilon}\| = \boldsymbol{K}\|\boldsymbol{\epsilon}\| \tag{A.18}$$

Therefore, the optimal solution is locally continuous according to the Lemma 2 in the Appendix.

A.4 Codes related to WPDM

All Matlab codes are based on MATLAB 2018a version.

A.4.1 WPDM

The codes of WPDM are the following:

```
function [X,fval,exitflag,output] = PrimalDualMethod(c,A,b,w)

%% pretreatment
A_ = sparse([A eye(size(A,1))]);
c_ = [c; zeros(size(A,1),1)];
[tmp_m, tmp_n] = size(A_);
if size(c_,1) < size(c_,2) || size(b,1) < size(b,2) || ...
size(c_,1) ~= tmp_n || size(b,1) ~= tmp_m
   error('LP: data dimensions does not match.');
end

%% parameter
maxIter = 1000;
verbose = 0;
tau = 1e-8;
maxDiag = 5.e+15;
etaMin  = .995;
factor = 0.75;

iterN = 0;
mu=10;
exitflag = -1;

if verbose == 0
%         warning('off','MATLAB:nearlySingularMatrix');
        warning('off');
else
        warning('on','all');
end
```

```
if isempty(w)
    w = ones(size(A,1)+size(A,2),1);
end
%% initial solution
e = ones(tmp_n,1);

% solution for min norm(s) s.t. A'*y + s = c
y0 = (A_*A_.')\(A_*c_);
s0 = c_-A_.'*y0;

% min norm(x) s.t. Ax = b
x0 = A_.'*( (A_*A_.')\b );

% delta_x and delta_s
delta_x = max(-1.5*min(x0),0);
delta_s = max(-1.5*min(s0),0);

% delta_x_c and delta_s_c
pdct = 0.5*(x0+delta_x*e)'*(s0+delta_s*e);
delta_x_c = delta_x+pdct/(sum(s0)+tmp_n*delta_s);
delta_s_c = delta_s+pdct/(sum(x0)+tmp_n*delta_x);

% output
x = x0+delta_x_c*e;
s = s0+delta_s_c*e;
y = y0;

% x = rand(size(x0));
% s = rand(size(s0));
% y = rand(size(y0));

%% calculate residual

Rd = A_.' * y + s - c_;
Rp = A_  * x - b;
Rc = x.*s - mu*w;
residual = norm([Rp;Rd;Rc]);
```

```
mu = factor * mu;

if verbose == 2
    fprintf('%4s %9s %9s\n','ITER', 'MU', 'RESIDUAL');
end

%% iteratate

while true

    % Check Tolerance
    if mu < tau
        exitflag = 1;
        optx = x;
        opty = y;
        opts = s;
        break;
    end

    % Check maxIter
    if iterN >= maxIter
        exitflag = 0;
        optx = x;
        opty = y;
        opts = s;
        break;
    end

    % Output
    if verbose == 2
        fprintf('%4d %9.2e %9.2e\n',iterN, mu, residual);
    end


    % Use augmented system to solve the directions
    rhs = sparse([-Rp; -Rd + Rc ./ x]);

    % Set up the scaling matrix and form the coef matrix for normal equations
```

```
DD = min(maxDiag,-s ./ x);
B = [sparse(tmp_m,tmp_m) A_; A_.' sparse(1:tmp_n,1:tmp_n,DD)];

% ldl' factorization
[L, D, pm] = ldl(B,'vector');

% Solve linear system
dxy = zeros(tmp_m + tmp_n, 1);
dxy(pm, :) =L'\(D\(L\(rhs(pm, :))));

% Get the directions
dy = dxy(1:tmp_m);
dx = dxy( tmp_m + 1 : tmp_m + tmp_n );
ds = -( Rc + s .* dx ) ./ x;

% Get step length
eta = max(etaMin, 1-mu);

alphax = -1/min(min( dx ./ x),-1);
alphax = min(1, eta * alphax);

alphas = -1/min(min( ds ./ s),-1);
alphas = min(1, eta * alphas);

% Update the iterate
x = x + alphax*dx;
s = s + alphas*ds;
y = y + alphas*dy;

% Calculate residuals
Rd = A_.' * y + s - c_;
Rp = A_  * x - b;
Rc = x.*s - mu*w;
residual = norm([Rp;Rd;Rc]);
mu = factor * mu;

% Increase counter
iterN = iterN + 1;
```

```
end

len = size(c,1);
X = optx(1:len);
fval = c_' * optx;

if verbose == 2
    fprintf('Iteration = %u\n',iterN);
    fprintf('Function value = %9.2e\n',fval);
    fprintf('Residual = %9.2e\n',residual);
    fprintf('mu = %9.2e\n',mu);
    if exitflag==1
        disp('Optimization is terminated by tolerance');
    else
        disp('Optimization is terminated by maxIter');
    end
end

if verbose == 1
    fprintf('Function value = %9.2e\n',fval);
    if exitflag==1
        disp('Optimization is terminated by tolerance');
    else
        disp('Optimization is terminated by maxIter');
    end
end



output.tau = mu;
output.x = optx;
output.y = opty;
output.s = opts;
end
```

A.4.2 Example of WPDM

Codes of example in (3.19) are the following:

```
%%
A = [1 1 0; 1 0 2];
b = [4 8].';

Vertex = [4 0 2; 0 4 4; 0 4 0; 4 0 0; 0 0 0; 0 0 4];
fig = figure('Position',[100, 100, 452+300, 477]);
fig.Color = [1 1 1];
ax = axes('Position',[0.1 0.12 0.5 0.8]);
ax.ActivePositionProperty = 'outerposition';
ax.GridLineStyle = '--';
ax.LineWidth = 1.5;
ax.FontSize = 16;


patch('Faces',[3 4 5],'Vertices',Vertex,'EdgeColor','black',...
'FaceColor',[0.259 0.526 0.957],'LineWidth',0.1,'FaceAlpha',0,'EdgeAlpha',1);
patch('Faces',[5 4 1 6],'Vertices',Vertex,'EdgeColor','black',...
'FaceColor',[0.259 0.957 0.467],'LineWidth',0.1,'FaceAlpha',0,'EdgeAlpha',1);
patch('Faces',[6 2 3 5],'Vertices',Vertex,'EdgeColor','black',...
'FaceColor',[0.957 0.945 0.259],'LineWidth',0.1,'FaceAlpha',0,'EdgeAlpha',1);
patch('Faces',[3 4 1 2],'Vertices',Vertex,'EdgeColor','black',...
'FaceColor',[0.3010 0.7450 0.9330],'LineWidth',0.1,'FaceAlpha',0.5,'EdgeAlpha',1);
patch('Faces',[2 6 1],'Vertices',Vertex,'EdgeColor','black',...
'FaceColor',[0.679 0.259 0.957],'LineWidth',0.1,'FaceAlpha',0,'EdgeAlpha',1);
xlabel('\it x_1','FontSize',18,'FontName','Times New Roman',...
'FontWeight','bold','position',[2.34 -0.456 -0.254]);
ylabel('\it x_2','FontSize',18,'FontName','Times New Roman',...
'FontWeight','bold','position',[-0.70 2.24 -0.135]);
zlabel('\it x_3','FontSize',18,'FontName','Times New Roman',...
'FontWeight','bold','position',[-1.6 2.17 3.41]);

hold on;
grid on;
```

```
view(-32.2,32.8);
hold on;

%% path 1
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [klist(ii) 1 1 1 1].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end

sct1 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'p',...
'MarkerEdgeColor',[0 0 0],'LineWidth',1);

%% path 2
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [1 klist(ii) 1 1 1 ].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end
```

```
sct2 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'*',...
'MarkerEdgeColor',[0 0 0],'LineWidth',0.5);

%% path 3
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [1 1  klist(ii) 1 1].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end

sct3 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'*',...
'MarkerEdgeColor',[0 0 0],'LineWidth',3);

%% path 4
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [1 1 1 klist(ii) 1 ].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end
```

```
sct4 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'O',...
'MarkerEdgeColor',[0 0 0],'LineWidth',1);

%% path 5
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [1 1 1 1 klist(ii)].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end

sct5 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'s',...
'MarkerEdgeColor',[0 0 0],'LineWidth',2);

%% path 6
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [klist(ii) 1 1 1 klist(ii)].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end
```

```
sct6 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'d',...
'MarkerEdgeColor',[0 0 0],'LineWidth',1);

%% path 7
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
klist = logspace(log10(1),log10(1000),num);
list = zeros(num,5+3+1);

count = 1;
for ii = 1:num
    w = [klist(ii) klist(ii)  klist(ii)  1 1 ].';
    [x,fval,exitflag,output] = PrimalDualMethod(c,A,b,w);
    list(count,:) = [w; x; fval].';
    count = count + 1;
end

sct7 = scatter3(list(:,6),list(:,7),list(:,8),50,[0 0 0],'x',...
'MarkerEdgeColor',[0 0 0],'LineWidth',1);

%%
hL = legend([sct1,sct2,sct3,sct4,sct5,sct6,sct7],{'Increasing w_1',...
'Increasing w_2','Increasing w_3','Increasing w_4','Increasing w_5',...
'Increasing w_1 and w_5','Increasing w_1, w_2 and w_3'});
hL.EdgeColor = [1 1 1];
hL.FontSize = 14;
hL.FontName= 'Times New Roman';
hL.FontWeight= 'bold';
% Programatically move the Legend
newPosition1 = [0.67 0.35 0.2 0.25];
newUnits1= 'normalized';
set(hL,'Position', newPosition1,'Units', newUnits1);
```

A.5 SME under the Assumption of Unique Solution of LP

The related codes of SME in chapter 3 in MATLAB. More codes can be found on the website: https://github.com/SetMembershipEstimationDFBM/E.coliExample.

```
% Shen, X., & Budman, H. (2021). Set Membership Estimation with Dynamic
% Flux Balance Models. Processes, 9(10), 1762.
% https://www.mdpi.com/2227-9717/9/10/1762

% Copyright 2021 Xin Shen
% Copyright 2021 Hector Budman

% Multi-Parametric Toolbox 3.0 is required
% Algorithm has been tested on MATLAB R2018a.

%% Parameter calculations
clear;
clc;
startTime = datetime('now');

% stoichiometry coefficient
A = [0     -9.46   -9.84   -19.23;...   %glc
    -35    -12.92  -12.73  0;...       %o2
    -39.43   0       1.24    12.12;...    %ace
    1        1      1        1];       %X
% objective vector
c = ones(4,1);

% true plant initial concentration
z0 = [0.4 0.21 0.2 0.001]'; %(glc [mM], o2[mM], ace[mM], X[mM])
vol = 0.3;    % initla volume V[L]

% lower bound and upper bound
lb = [0.38; 0.1995; 0.19; 0.00095];
ub = [0.42; 0.2205; 0.21; 0.00105];
```

```
z = lb + (ub-lb).*rand(length(z0),1);

% define initla set
Box = Polyhedron('lb',lb,'ub',ub);

Km = 0.015;         % [mM]
GUR_max = 6.5;      % [mM/g-dw/hr]
OUR_max = 12;       % [mM/g-dw/hr]
kla = 4;            % [hr^-1]
z_fd = 5;

nstep = 160;
tend = 8; % h, total time of cell culture growth, guess
dt = tend/nstep; % h, time, guess
Timeseries = (1:nstep)*dt;

Constr = [-A(2,:); A(3,:); -A(1:3,:)*dt;-A(1,:);-A(4,:)*dt];
MptOptions.fastbreak = true;

b0 = [OUR_max; 100];
flux = sdpvar(4,1);     % decision variables
theta = sdpvar(5,1);  % parameters variables
obj = -c.'*flux;        % LP objective
G = [zeros(2,5);eye(5)];
B = [b0;zeros(5,1)];
Con1 = [Constr*flux <= B+G*theta, flux>=0];
plp = Opt(Con1, obj, theta, flux);
% solve the mpLP to obtain different critical regions
Solution = plp.solve();
% Only critical region 12 and 14 are related to this research.
% when concentration is very low, MPT toolbox may locate the state coexist
% in several critical regions because of tolerance issue.
% 12 and 14 are the true critical regions.

%% Observer Initialization
% rng('default');
% sigma = [0.025,0.0033,0.0200,0.004].';
nstate = length(z0);
```

```
mindex = [4];    %measurements
nmeas = length(mindex);

oindex = cell(14,1);    %observable state(subsystem), include bio and volume
oindex{12} = [1,4];
oindex{14} = [1,3,4];

modmsindex = cell(24,1);    %model measurements index
modmsindex{12} = [2];
modmsindex{14} = [3];
CRindex = [12];
C = [0 0 0 1];

% noise
Q0 = diag(zeros(4,1));
% N0 process noise covariance (5 states)!!!
P0 = diag((z0*0.05/3).^2);
%initial state convariance (5 states)
R = diag((sigma(mindex)).^2);
%measurement noise covariance

Q = Q0(oindex{12},oindex{12});
%process noise covariance (3 states for CR12)
P = P0(oindex{12},oindex{12});
%initial state convariance (3 states for CR12)

%range
% This research assume no process noise.
Qlb = sigma*0;
%No process noise !!!
Qub = sigma*0;
%No process noise !!!
Mlb = -sigma*0.1;
Mub = sigma*0.1;
RBox = Polyhedron('lb',Mlb(mindex),'ub',Mub(mindex));

plantstate = z;
Obserstate = z0(oindex{CRindex});
```

```
xPlant = zeros(nstate,nstep);
xPlant(:,1) = plantstate;
switchpoint = [];

xlb = zeros(nstate,nstep);
xub = zeros(nstate,nstep);
xlb(:,1) = lb;
xub(:,1) = ub;
y = zeros(nmeas,nstep);

xPre = cell(nstep,1);
PPre = cell(nstep,1);
xCrr = cell(nstep,1);
PCrr = cell(nstep,1);
Amatrix = cell(nstep,1);
PhdSet = cell(nstep,1);
xCrr{12} = Obserstate;
xCrr{14} = [];
PCrr{1} = P0(oindex{CRindex},oindex{CRindex});

BoxSet = cell(nstep,2);
BoxSet{1,1} = Box;
BoxSet{1,2} = [];

PlantStateWrap = @(z,rf,rp)PlantState(z,rf,rp,OUR_max,z_fd,dt,...
GUR_max,Km,kla,A,c,Constr,Q0,Qlb,Qub,vol);
SensorWrap = @(plantstate)Sensor(plantstate,mindex,R,Mlb,Mub);

tol = 0.08;

IsSwitched = false;
method = 'vrep';

fig = figure;
fig.Color = [1 1 1];
subplot(2,2,1)
ylabel('Glucose');
hold on;
```

```
h1 = animatedline;
h1l = animatedline('Color','r');
h1u = animatedline('Color','r');
h1e = animatedline('Color','g');
xlim([0 8]);

subplot(2,2,2)
ylabel('Oxygen');
hold on;
h2 = animatedline;
h2l = animatedline('Color','r');
h2u = animatedline('Color','r');
xlim([0 8]);

subplot(2,2,3)
ylabel('Acetate');
hold on;
h3 = animatedline;
h3l = animatedline('Color','r');
h3u = animatedline('Color','r');
h3e = animatedline('Color','g');
xlim([0 8]);

subplot(2,2,4)
ylabel('Biomass');
hold on;
h4 = animatedline;
h4l = animatedline('Color','r');
h4u = animatedline('Color','r');
h4e = animatedline('Color','g');
xlim([0 8]);

flag = false;

%%
for k=2:nstep
 if IsSwitched
     % check observablility
```

```
     if rank(ObFun14(Oberstate))~=3 || rank(AFun14(Oberstate))~=3
         disp(Oberstate);
     end
 end

%batch process
    rf = 0;
    rp = 0;

%--------------------PLANT----------------------
    plantstate = PlantStateWrap(plantstate,rf,rp);
    xPlant(:,k) = plantstate;
    ym = SensorWrap(plantstate);
    y(:,k) = ym;

 %------------------Montinor--------------------
    if IsSwitched
    else
        [IsSwitched,CRindex,est] = Monitor(Oberstate,GUR_max,Km,Solution,...
        tol,CRindex,xlb(:,k-1),xub(:,k-1),rf,dt,vol);

        if IsSwitched && (flag == false)
            disp('Montior switch critical region!');
            switchpoint = [switchpoint k];
            temp = zeros(3,1);
            temp(1) = Oberstate(1);
            temp(3) = Oberstate(2);
            temp(2) = est;
            Oberstate = temp;
            Ptemp = zeros(3);
            Ptemp([1 3],[1 3]) = P;
            Ptemp(2,2) = ((xub(2,k-1)-xlb(2,k-1))/2/3)^2;
            P = Ptemp;
            P = P*1.1;
            %If necessary, the covariance can be initilized
            %larger to deal with the early or late swtich issue
            Q = Q0(oindex{CRindex},oindex{CRindex});
            %no process noise has been set as 0
```

```
            flag = true;
            xCrr{14} = [xCrr{14} Obserstate];
        end
    end


%----------Set-Membership-Estimation-----------------
    [Box] = BoxPropogate(Box,ym,Obserstate,rf,rp,z_fd,dt,GUR_max,Km,kla,A,...
        Solution,Qlb,Qub,RBox,C,CRindex,method,sigma,P,vol);
    BoxSet{k,1} = Box;
    BoxSet{k,2} = [];  %scaling

    xlb(:,k) = Box.Internal.lb;
    xub(:,k) = Box.Internal.ub;

%------------------Kalman Filter--------------------


    ModelStateWrap = @(z,rf,rp)ModelState(z,rf,rp,z_fd,dt,GUR_max,Km,...
    A,Solution,CRindex,vol);
    ModelMeasureWrap = @(state)ModelMeasure(state,modmsindex{CRindex});

    [xPre{k},PPre{k},Obserstate,P,Amatrix{k}] = ekf(@(z)ModelStateWrap...
    (z,rf,rp),Obserstate,P,ModelMeasureWrap,ym,Q,R);
    if ~IsSwitched
        xCrr{12} = [xCrr{12} Obserstate];
    else
        xCrr{14} = [xCrr{14} Obserstate];
    end
    PCrr{k} = P;


%--------------------Plot--------------------
    addpoints(h1,k*dt,plantstate(1));
    addpoints(h2,k*dt,plantstate(2));
    addpoints(h3,k*dt,plantstate(3));
    addpoints(h4,k*dt,plantstate(4));
```

```
    addpoints(h1l,k*dt,xlb(1,k));
    addpoints(h2l,k*dt,xlb(2,k));
    addpoints(h3l,k*dt,xlb(3,k));
    addpoints(h4l,k*dt,xlb(4,k));

    addpoints(h1u,k*dt,xub(1,k));
    addpoints(h2u,k*dt,xub(2,k));
    addpoints(h3u,k*dt,xub(3,k));
    addpoints(h4u,k*dt,xub(4,k));

    drawnow;

end
endTime = datetime('now');

function stateobserved = Sensor(plantstate,mindex,R,Mlb,Mub)
    stateobserved = plantstate(mindex);
    % add bounded Gaussian noise
    if length(mindex)>1
        noise = mvrandn(Mlb(mindex),Mub(mindex),R,1);
    else
        while true
            noise = normrnd(0,R);
            if noise>=Mlb(mindex) && noise<=Mub(mindex)
                break ;
            end
        end
    end
    stateobserved = plantstate(mindex) + noise;
end
function stateobserved = ModelMeasure(state,index)
    stateobserved = state(index);
end

function z = ModelState(z,rf,rp,z_fd,dt,GUR_max,Km,A,Solution,CRindex,vol)
switch CRindex
    case 12
        b = zeros(5,1);
```

```
        b(4) = GUR_max*z(1)/(Km+z(1));
        flux = round(Solution.xopt.Set(12).Functions('primal').F,6)*b +...
        round(Solution.xopt.Set(12).Functions('primal').g,6);
        z(1) = z(1) - rf*dt*z(1)/vol + A(1,:)*flux*z(2)*dt + rf*dt/vol*z_fd;
        z(2) = z(2) - (rf - rp)*dt/vol*z(2) + A(4,:)*flux*z(2)*dt;
    case 14
        b = zeros(5,1);
        b(3) = z(2)/z(3) - rf/vol*z(2)/z(3)*dt;
        b(4) = GUR_max*z(1)/(Km+z(1));
        flux = round(Solution.xopt.Set(14).Functions('primal').F,6)*b +...
        round(Solution.xopt.Set(14).Functions('primal').g,6);
        z(1) = z(1) - rf*dt*z(1)/vol + A(1,:)*flux*z(3)*dt + rf*dt/vol*z_fd;
        z(2) = z(2) - rf*dt*z(2)/vol + A(3,:)*flux*z(3)*dt;
        z(3) = z(3) - (rf - rp)*dt/vol*z(3) + A(4,:)*flux*z(3)*dt;
end

end



function z = PlantState(z,rf,rp,OUR_max,z_fd,dt,GUR_max,Km,kla,A,...
c,Constr,Q,Qlb,Qub,vol)
    global b1
    b0 = [OUR_max; 100];
    b1 = z(1:3)/z(4) - rf/vol*z(1:3)/z(4)*dt;
    b1(1) = b1(1) + rf/vol*z_fd/z(4)*dt;
    b1(2) = b1(2) + kla*(0.21 - z(2))/z(4)*dt;
    b2 = GUR_max*(z(1)/(Km + z(1)));
    b3 = 1 - (rf-rp)*dt/vol;

    global nu
    problem = Opt('f',-c,'A',Constr,'b',[b0;b1;b2;b3],'lb',zeros(4,1));
    solution = problem.solve;
    nu = round(solution.xopt,6);

    z = z + ODE(0,z,nu,rf,z_fd,rp,A,kla,vol)*dt;
    z(z<=0)=0;
```

```
    function dzdt = ODE(t,z,nu,rf,z_fd,rp,A,kla,vol)
        dzdt = zeros(4,1);
        dzdt(1:4) = A*nu*z(4) - rf/vol*z(1:4);
        dzdt(1) = dzdt(1) + rf/vol*z_fd;
        dzdt(2) = dzdt(2) + kla*(0.21 - z(2));
        dzdt(4) = dzdt(4) + z(4)*rp/vol;
    end
end

function [BoxNew] = BoxPropogate(Box,ym,Obserstate,rf,rp,z_fd,dt,GUR_max,...
                        Km,kla,A,Solution,Qlb,Qub,RBox,C,CRindex,method,...
                        sigma,P,vol)
switch CRindex
    case 12

        b = zeros(5,1);
        bhat = zeros(5,2);
        [b(4),bhat(4,1)]=jaccsd(@(x)(GUR_max*x/(Km+x)),Obserstate(1));
        flux = round(Solution.xopt.Set(12).Functions('primal').F,6)*b +...
        round(Solution.xopt.Set(12).Functions('primal').g,6);
        bffHat = zeros(4,2);
        translate = [z_fd*rf*dt/vol; 0.21*kla*dt; 0; 0];

        bff = A*flux*Obserstate(2)*dt + translate;

        Aff = [1 - rf*dt/vol; 1 - rf*dt/vol-kla*dt;...
               1 - rf*dt/vol; 1 - (rf-rp)*dt/vol];
        T = diag(Aff);

        termB = (3*P(2,2)^0.5 + Obserstate(2))*dt*A*round(...
        Solution.xopt.Set(12).Functions('primal').F,6)*bhat + bffHat;
        termD = dt*A*flux;

        var = 3*diag(P).^0.5;
        Err = Polyhedron('lb',-var,'ub',var);
        Errbio = Polyhedron('lb',-var(2),'ub',var(2));
        term1 = PolyUnion(termB*Err).outerApprox;
        term2 = PolyUnion(termD*Errbio).outerApprox;
```

```
    safetyl = term1.Internal.lb + term2.Internal.lb;
    safetyu = term1.Internal.ub + term2.Internal.ub;

    Prior0 = Box.affineMap(T,method) + bff;
    Prior1 = PolyUnion(Prior0);
    Prior1Box = Prior1.outerApprox;

    Prior1Boxlb = Prior1Box.Internal.lb + Qlb + safetyl;
    Prior1Boxub = Prior1Box.Internal.ub + Qub + safetyu;
    BoxPrior = Polyhedron('lb',Prior1Boxlb,'ub',Prior1Boxub);

case 14
    b = zeros(5,1);
    bhat = zeros(5,3);

    [b(3),bhat(3,:)]=jaccsd(@(x)x(2)/x(3) - rf/vol*x(2)/x(3)*dt,...
    Obserstate);
    [b(4),bhat(4,1)]=jaccsd(@(x)(GUR_max*x/(Km+x)),Obserstate(1));

    flux = round(Solution.xopt.Set(14).Functions('primal').F,6)*b +...
    round(Solution.xopt.Set(14).Functions('primal').g,6);


    bffHat = zeros(4,3);

    translate = [z_fd*rf*dt/vol; 0.21*kla*dt; 0; 0];

    bff = A*flux*Obserstate(3)*dt + translate;

    Aff = [1 - rf*dt/vol; 1 - rf*dt/vol-kla*dt;...
           1 - rf*dt/vol; 1 - (rf-rp)*dt/vol];
    T = diag(Aff);

    termB = (3*P(3,3)^0.5 + Obserstate(3))*dt*A*round(...
    Solution.xopt.Set(14).Functions('primal').F,6)*bhat + bffHat;
    termD = dt*A*flux;

    var = 3*diag(P).^0.5;
```

```
        Err = Polyhedron('lb',-var,'ub',var);
        Errbio = Polyhedron('lb',-var(3),'ub',var(3));
        term1 = PolyUnion(termB*Err).outerApprox;
        term2 = PolyUnion(termD*Errbio).outerApprox;
        safetyl = term1.Internal.lb + term2.Internal.lb;
        safetyu = term1.Internal.ub + term2.Internal.ub;

        Prior0 = Box.affineMap(T,method) + bff;
        Prior1 = PolyUnion(Prior0);
        Prior1Box = Prior1.outerApprox;

        Prior1Boxlb = Prior1Box.Internal.lb + Qlb + safetyl;
        Prior1Boxub = Prior1Box.Internal.ub + Qub + safetyu;
        BoxPrior = Polyhedron('lb',Prior1Boxlb,'ub',Prior1Boxub);

end

    R = Polyhedron('A',-RBox.A*C,'b',RBox.b-RBox.A*ym);
    BoxPost = intersect(R,BoxPrior);

    U = PolyUnion(BoxPost);
    OutBox = U.outerApprox;

    % When the state interval is very small. Numerical issue can happen. So
    % a very tiny interval is given artifically to let the algorithm
    % continue.
    lb = OutBox.Internal.lb - sigma.*0;
    lb(lb<0) = 0;

    ub = OutBox.Internal.ub + sigma.*0 ;
     ub(ub<0) = 0;
    lb1 = lb;
    ub1 = ub;
    distance = 0.000001;

    conver = abs(ub-lb)<=distance & lb>0;
    lb1(conver) = (lb(conver) + ub(conver))*0.5 - distance*0.01;
    ub1(conver) = (lb(conver) + ub(conver))*0.5 + distance*0.01;
```

```
    conver1 = abs(ub-lb)<=distance & lb<=0;
    lb1(conver1) = 0;
    ub1(conver1) = distance*0.01;

    BoxNew = Polyhedron('lb',lb1,'ub',ub1);
    BoxNew.computeVRep();
end

function [IsSwitched,CRindex,est] = Monitor(z,GUR_max,Km,Solution,tol,...
CRindex,xlb,xub,rf,dt,vol)
    theta3lb = xlb(3)/z(2) - rf/z(2)*xlb(3)/vol*dt;
    theta3ub = xub(3)/z(2) - rf/z(2)*xub(3)/vol*dt;
    theta3p = (theta3lb+theta3ub)/2;

    b = zeros(5,1);
    b(3) = theta3p;
    b(4) = GUR_max*z(1)/(Km+z(1));
    vLeft = round(Solution.xopt.Set(12).Functions('primal').F,6)*b +...
    round(Solution.xopt.Set(12).Functions('primal').g,6);
    vRight = round(Solution.xopt.Set(14).Functions('primal').F,6)*b +...
    round(Solution.xopt.Set(14).Functions('primal').g,6);
    diff = norm(vLeft-vRight);

    % detect switch
    if norm(diff)<=tol
        CRindex = 14;
        IsSwitched = true;

        Theta = sym('Theta',[5,1]);
        F = (round(Solution.xopt.Set(12).Functions('primal').F,6)-...
        round(Solution.xopt.Set(14).Functions('primal').F,6))*Theta...
        +(round(Solution.xopt.Set(12).Functions('primal').g,6)-...
        round(Solution.xopt.Set(14).Functions('primal').g,6));
        theta4 = b(4);
        F0 = subs(F,Theta(4),theta4);
        theta3 = eval(solve(F0==0));
        % estimate the unobserved concetration
```

```
        est = theta3./(1/z(2)- rf/z(2)/vol*dt);
    else
        IsSwitched = false;
        est = [];
    end
end

 function [x1,P,x2,P2,A]=ekf(fstate,x0,P,hmeas,y,Q,R)
 % Extended Kalman Filter
[x1,A]=jaccsd(fstate,x0);
P=A*P*A'+Q;
[y1,C]=jaccsd(hmeas,x1);
P12=P*C';
K=P12/(C*P12+R);
x2=x1+K*(y-y1);
P2=P-K*P12';
end

function [z,A]=jaccsd(fun,x)
% Jacobian Matrix
z=fun(x);
n=numel(x);
m=numel(z);
A=zeros(m,n);
h=n*eps;
for k=1:n
    x1=x;
    x1(k)=x1(k)+h*1i;
    A(:,k)=imag(fun(x1))/h;
end
end

function A = AFun14(in1)
%AFUN14
%    A = AFUN14(IN1)

%    This function was generated by the Symbolic Math Toolbox version 8.1.
```

```
%    30-Jul-2021 16:34:00

Z_01 = in1(1,:);
Z_04 = in1(3,:);
A = reshape([(Z_04.*(-3.24999246e-1))./(Z_01+3.0./2.0e2)...
+Z_01.*Z_04.*1.0./(Z_01+3.0./2.0e2).^2.*3.24999246e-1+1.0,...
0.0,Z_04.*(3.43551e-2./...(Z_01+3.0./2.0e2)-Z_01.*1.0./...
(Z_01+3.0./2.0e2).^2.*3.43551e-2),0.0,-2.000000024038329e-9,...
2.53614e-2,(Z_01.*(-3.24999246e-1))./(Z_01+3.0./2.0e2),0.0,...
(Z_01.*3.43551e-2)./(Z_01+3.0./2.0e2)+1.0],[3,3]);
```

A.6 SME with WPDM

A.6.1 Multiparametric programming of WPDM

Codes of mpNLP of WPDM for the *E. coli* example.

```
clear;
date = datetime('now','TimeZone','local','Format','MMdd_HHmm') ;
name = ['rfine' char(date) '.mat'];
load('paradataMP'); % simulation data to trim critical regions
mu = 1.01020837395261e-08;
C = -ones(4,1);
wc = [5 1 1 1 1   1 1 1 1 1  1 1]';

GUR_max = 6.5;     % [mM/g-dw/hr]
OUR_max = 12;      % [mM/g-dw/hr]
nstep = 600;
tend = 15; % h, total time of cell culture growth, guess
dt = tend/nstep; % h, time, guess
Coeff = [0     -9.46   -9.84   -19.23;...   %glc
    -35    -12.92  -12.73  0;...       %o2
    -39.43   0       1.24    12.12;...    %ace
    1        1      1        1];       %X
Constr = [-Coeff(2,:); Coeff(3,:); -Coeff(1:3,:)*dt;-Coeff(1,:);...
-Coeff(4,:)*dt];
```

```
Ac = [Constr;-eye(4);1 1 1 1];
bc = [OUR_max;100;0;0;0;0;0;zeros(4,1);0.5];
Fc = [zeros(1,5);zeros(1,5);eye(5);zeros(4,5);zeros(1,5)];
Fc(:,[4 5]) = Fc(:,[4 5])/100;
Len = sqrt(sum(Ac.^2,2));
Ac = Ac./Len;
bc = bc./Len;
Fc = Fc./Len;

% Parameter Space
z_lb = [0,0,0,0,0.9]';
z_ub = [500,250,250,GUR_max,1.005]';
z_lb([4 5]) = z_lb([4 5]) * 100;
z_ub([4 5]) = z_ub([4 5]) * 100;
Q = [-eye(5);eye(5)];
q = [-z_lb;z_ub];
Len1 = sqrt(sum(Q.^2,2));
Q = Q./Len1;
q = q./Len1;

xtol = 0.001;
v_tol = 250;
zero_tol = 1e-6;
maxRespectRatio = inf;
maxNRegions = 100000;
safegap = 1e8*eps;

quadoptions = optimoptions('quadprog');
quadoptions.ConstraintTolerance = 1e-16;
quadoptions.Display = 'off';
quadoptions.MaxIterations = 5000;
quadoptions.OptimalityTolerance = 1e-16;
quadoptions.StepTolerance = 1e-20;


%% containers and parameters
Ground = Polyhedron('A',Q,'b',q);
```

```
MultiActiveSet = {};
SingActiveSet = {};
clear MultiRegion
clear SingRegion
MultiRegion(1) = Polyhedron();
MultiRegion(1) = [];
SingRegion(1) = Polyhedron();
SingRegion(1) = [];
MultiRegionRefined(1) = PolyUnion();

if zero_tol<=safegap
    error('zero_tol should be greater than safegap');
end



%% find initial point
nx = length(C);
nth = size(Fc,2);
nCons = size(Ac,1);

wpdmObjFun = @(U)C'*U(1:nx)+sum(-mu*wc.*log(bc+[-Ac Fc]*U));
Acmb = [Ac -Fc;zeros(length(q),size(Ac,2)) Q];
bcmb = [bc; q];
Poly0 = Polyhedron('A',Acmb,'b',bcmb);
Pnt0 = Poly0.interiorPoint.x;
[x0,fval0,exitflag0,output0,lambda0,~,~] = fmincon(wpdmObjFun,Pnt0,...
Acmb,bcmb);
if exitflag0<0
    disp('The problem is infeasible.');
end

Point = x0(nx+1:end);
[x1,fval1,exitflag1,output1,lambda1] = NewPrimalDualMethod(C,Ac,...
bc+Fc*Point,wc);
active = find(abs(Ac*x1-bc-Fc*Point)<=zero_tol);
if length(active)<nx
    inactive = setdiff(1:nCons,active);
```

```
    A = Ac(inactive,:);
    b = bc(inactive,:);
    F = Fc(inactive,:);
    Ae = Ac(active,:);
    be = bc(active,:);
    Fe = Fc(active,:);
    P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],...
    'Ae',[Ae -Fe],'be',be).projection(nx+1:nx+nth);
    P = P.normalize();
    MultiRegion = [MultiRegion;P];
    MultiActiveSet{length(MultiActiveSet)+1,1} = active;

    w = wc(inactive,:);
  [Explored] = ExploreMultiplicity(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,B,P,...
  nx,zero_tol,xtol,quadoptions,v_tol,maxRespectRatio);
    if length(MultiRegionRefined)==1
    if isempty(MultiRegionRefined(1).Set)
        len = 0;
    else
        len = 1;
    end
    else
    len = length(MultiRegionRefined);
    end
    MultiRegionRefined(len+1) = PolyUnion(Explored);
elseif length(active)==nx
    inactive = setdiff(1:nCons,active);
    A = Ac(inactive,:);
    b = bc(inactive,:);
    F = Fc(inactive,:);
    Ae = Ac(active,:);
    be = bc(active,:);
    Fe = Fc(active,:);
    P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],...
    'Ae',[Ae -Fe],'be',be).projection(nx+1:nx+nth);
    SingRegion = [SingRegion;P];
    SingActiveSet{length(SingActiveSet)+1,1} = active;
else
```

```
flag = true;
while flag
    purDirc = randn(1,nth)';
    purDirc = purDirc/sqrt(sum(purDirc.^2));
    thPerturbed = Point + purDirc*safegap*1e3;
    if ~any(contains(Ground,thPerturbed))
        continue;
    end
    [x2,fval2,exitflag2,output2,lambda2] = NewPrimalDualMethod(...
    C,Ac,bc+Fc*thPerturbed,wc);
    if exitflag2<0
        continue;
    end
    active = find(abs(Ac*x2-bc-Fc*thPerturbed)<=zero_tol);
    if length(active)>nx
        continue;
    end
    flag = false;
    if length(active)<nx
        inactive = setdiff(1:nCons,active);
        A = Ac(inactive,:);
        b = bc(inactive,:);
        F = Fc(inactive,:);
        Ae = Ac(active,:);
        be = bc(active,:);
        Fe = Fc(active,:);
        P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],'Ae',...
        [Ae -Fe],'be',be).projection(nx+1:nx+nth);
        P = P.normalize();
        MultiRegion = [MultiRegion;P];
        MultiActiveSet{length(MultiActiveSet)+1,1} = active;

        w = wc(inactive,:);
        [Explored] = ExploreMultiplicity(C,Ae,be,Fe,A,b,F,Ac,bc,...
        Fc,w,wc,B,P,nx,zero_tol,xtol,quadoptions,v_tol,maxRespectRatio);
        if length(MultiRegionRefined)==1
            if isempty(MultiRegionRefined(1).Set)
                len = 0;
```

```
                else
                    len = 1;
                end
            else
                len = length(MultiRegionRefined);
            end
            MultiRegionRefined(len+1) = PolyUnion(Explored);
        else
            inactive = setdiff(1:nCons,active);
            A = Ac(inactive,:);
            b = bc(inactive,:);
            F = Fc(inactive,:);
            Ae = Ac(active,:);
            be = bc(active,:);
            Fe = Fc(active,:);
            P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],'Ae',...
            [Ae -Fe],'be',be).projection(nx+1:nx+nth);
            thC = mean(P.V)';
            w = wc(inactive,:);
            [xFun] = qpApp(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,thC,nx,zero_tol,...
            quadoptions);
            P.addFunction(xFun,'opt');
            Region = [Region;P];
            SingRegion = [SingRegion;P];
            SingActiveSet{length(SingActiveSet)+1,1} = active;
            nRegions = nRegions + 1;
        end
    end
end

%%
currentregion = 1;
nRegions = 1;
Region = [SingRegion;MultiRegion];
while currentregion <= nRegions & nRegions <= maxNRegions
    Region(currentregion).normalize();
    FacePoints = Region(currentregion).facetInteriorPoints';
    FaceDirc = Region(currentregion).A;
```

```
nBords = size(FacePoints,2);
for bord = 1:nBords
    xOut = FacePoints(:,bord) + FaceDirc(bord,:)'/norm(FaceDirc(bord,:))...
    *safegap*1e3;
    if ~any(contains(Ground,xOut))
        continue;
    end
    if any(contains(Region,xOut))
        continue;
    end
    [x3,fval3,exitflag3,output3,lambda3] = NewPrimalDualMethod(C,Ac,...
    bc+Fc*xOut,wc);
    if exitflag3<0
        continue;
    end
    active = find(abs(Ac*x3-bc-Fc*xOut)<=zero_tol);
    isRepeated = false;
    if length(active)<nx
        for ii=1:length(MultiActiveSet)
            if isempty(setdiff(active,MultiActiveSet{ii}))
                isRepeated = true;
                break;
            end
        end
        if isRepeated
            continue;
        end
        inactive = setdiff(1:nCons,active);
        A = Ac(inactive,:);
        b = bc(inactive,:);
        F = Fc(inactive,:);
        Ae = Ac(active,:);
        be = bc(active,:);
        Fe = Fc(active,:);
        P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],'Ae',...
        [Ae -Fe],'be',be).projection(nx+1:nx+nth);
        P = P.normalize();
        MultiRegion = [MultiRegion;P];
```

```
        MultiActiveSet{length(MultiActiveSet)+1,1} = active;
        Region = [Region;P];
        nRegions = nRegions + 1;

        w = wc(inactive,:);
        [Explored] = ExploreMultiplicity(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,...
        wc,B,P,nx,zero_tol,xtol,quadoptions,v_tol,maxRespectRatio);
        if length(MultiRegionRefined)==1
            if isempty(MultiRegionRefined(1).Set)
                len = 0;
            else
                len = 1;
            end
        else
            len = length(MultiRegionRefined);
        end
        MultiRegionRefined(len+1) = PolyUnion(Explored);

    elseif length(active)==nx
        for ii=1:length(SingActiveSet)
            if isempty(setdiff(active,SingActiveSet{ii}))
                isRepeated = true;
                break;
            end
        end
        for ii=1:length(MultiActiveSet)
            if all(ismember(MultiActiveSet{ii},active))
                isRepeated = true;
                break;
            end
        end

        if isRepeated
            continue;
        end
        inactive = setdiff(1:nCons,active);
        A = Ac(inactive,:);
        b = bc(inactive,:);
```

```
    F = Fc(inactive,:);
    Ae = Ac(active,:);
    be = bc(active,:);
    Fe = Fc(active,:);
    P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],'Ae',...
    [Ae -Fe],'be',be).projection(nx+1:nx+nth);
    thC = mean(P.V)';
    w = wc(inactive,:);
    [xFun] = qpApp(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,thC,nx,zero_tol,...
    quadoptions);
    P.addFunction(xFun,'opt');
    Region = [Region;P];
    SingRegion = [SingRegion;P];
    SingActiveSet{length(SingActiveSet)+1,1} = active;
    nRegions = nRegions + 1;

else
    xOut1 = FacePoints(:,bord) + 0.5*FaceDirc(bord,:)'/...
    norm(FaceDirc(bord,:))*safegap*1e3;
    [x4,fval4,exitflag4,output4,lambda4] = NewPrimalDualMethod(...
    C,Ac,bc+Fc*xOut1,wc);
    [~,active] = maxk(Ac*x4-bc-Fc*xOut1,nx);
    active = sort(active);
    for ii=1:length(SingActiveSet)
        if isempty(setdiff(active,SingActiveSet{ii}))
            isRepeated = true;
            break;
        end
    end
    for ii=1:length(MultiActiveSet)
        if all(ismember(MultiActiveSet{ii},active))
            isRepeated = true;
            break;
        end
    end

    if isRepeated
        continue;
```

```
            end

            inactive = setdiff(1:nCons,active);
            A = Ac(inactive,:);
            b = bc(inactive,:);
            F = Fc(inactive,:);
            Ae = Ac(active,:);
            be = bc(active,:);
            Fe = Fc(active,:);
            P = Polyhedron('A',[A -F;zeros(length(q),nx) Q],'b',[b;q],'Ae',...
            [Ae -Fe],'be',be).projection(nx+1:nx+nth);
            thC = mean(P.V)';
            w = wc(inactive,:);
            [xFun] = qpApp(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,thC,nx,zero_tol,...
            quadoptions);
            P.addFunction(xFun,'opt');
            Region = [Region;P];
            SingRegion = [SingRegion;P];
            SingActiveSet{length(SingActiveSet)+1,1} = active;
            nRegions = nRegions + 1;
        end
    end
    currentregion = currentregion + 1;
end

Space = SingRegion;
for ii=1:length(MultiRegionRefined)
    Space = [Space;MultiRegionRefined(ii).Set];
end

save(name,Space)


function [xFun] = qpApp(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,thC,nx,zero_tol,...
quadoptions)

[xC,fvalC,exitflagC,outputC,lambdaC] = NewPrimalDualMethod(C,Ac,bc+Fc*thC,wc);
if exitflagC<0
```

```
    error('infeasbile!');
end

L = (sum(w.*A./(b+F*thC-A*xC),1))';
H = (w.*A./(b+F*thC-A*xC))'*(A./(b+F*thC-A*xC));
G = -(w.*F./(b+F*thC-A*xC))'*(A./(b+F*thC-A*xC));
P = L - H*xC - G'*thC;
bue = Ae*H^-1*P + be;
Fue = Ae*H^-1*G' + Fe;

bu = A*H^-1*P + b;
Fu = A*H^-1*G' + F;
Right = bu+Fu*thC;
Righte = bue+Fue*thC;

[xQP,fvalQP,exitflagQP,outputQP,lambdaQP] = quadprog(H,zeros(nx,1),...
A,Right,Ae,Righte,[],[],xC,quadoptions);
if exitflagQP<0
    error('QP is infeasible');
end

%find active constraints
act=find(abs(A*xQP-Right)<=zero_tol, 1);
if ~isempty(act)
    error('Equality constraint should be the only active constraint');
end

HAact = H^-1*Ae';
AHAinv = (Ae*HAact)^-1;
temp = H^-1*Ae'*AHAinv;
S = temp*Fue;
R = temp*bue;

%explicit solution for current CR
GHinv = G*H^-1;
xFun = AffFunction (S-GHinv',R - H^-1*P); %solution to original mpQP

end
```

```
function [Space] = ExploreMultiplicity(C,Ae,be,Fe,A,b,F,Ac,bc,...
Fc,w,wc,B,Space,nx,zero_tol,xtol,quadoptions,v_tol,maxRespectRatio)

Split = Function;
Space.addFunction(Split,'Split');

flag = true;
while flag
    flag = false;
    if any(Space.hasFunction('Split'))
        flag = true;
        count = 1;
        while any(Space.hasFunction('Split'))
            splitId = find(Space.hasFunction('Split')==1,1);
            [L, U] = SplitRegion(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,...
            Space(splitId),nx,zero_tol,xtol,Split,quadoptions,...
            v_tol,maxRespectRatio);
            Space(splitId) = [];
            len = length(Space);
            Space(len+1) = L;
            Space(len+2) = U;

             if mod(count,50)

                position = zeros(length(B),1);
                parfor ii=1:length(B)
                    [isin,inwhich,~] = isInside(Space,B(:,ii));
                    if isin
                        position(ii) = inwhich;
                    end
                end
                set = unique(position);
                Space0 = Space;
                Space = Polyhedron;
                Space(1:length(set)) = Space0(set);
                Space0 = [];
            end
```

```
            count = count + 1;

        end
    end
end

end

function [L, U]= SplitRegion(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,region,...
nx,zero_tol,xtol,Split,quadoptions,v_tol,maxRespectRatio)

dim = region.Dim;
if ~region.hasVRep
    region.computeVRep();
end

lb = region.Internal.lb;
ub = region.Internal.ub;
mid = 0.5 * (lb + ub);
shape = ub-lb;

[~,id] = sort(shape);
id = flip(id);
rA = region.A;
rb = region.b;
rAe = region.Ae;
rbe = region.be;
Err = zeros(dim,1);
FunL = cell(dim,1);
FunU = cell(dim,1);
PolyLows = cell(dim,1);
PolyUps = cell(dim,1);
needSplitL = zeros(dim,1);
needSplitU = zeros(dim,1);
errid = ones(dim,1);

    parfor ii = 1:dim
        high = inf*ones(dim,1);
```

```
    high(ii) = mid(ii);
    low = zeros(dim,1);
    low(ii) = mid(ii);
    PolyLows{ii} = Polyhedron('A',rA,'b',rb,'Ae',rAe,'be',rbe,'ub',...
    high,'lb',zeros(dim,1));
    PolyUps{ii} = Polyhedron('A',rA,'b',rb,'Ae',rAe,'be',rbe,'lb',low);
    PolyLows{ii}.computeVRep();
    PolyUps{ii}.computeVRep();
    if ~(region.contains(PolyLows{ii}) & region.contains(PolyUps{ii}))
        errid(ii) = 0;
    end
    LowCenter = mean(PolyLows{ii}.V)';
    UpCenter = mean(PolyUps{ii}.V)';
    FunL{ii} = qpApp(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,LowCenter,nx,...
    zero_tol,quadoptions);
    FunU{ii} = qpApp(C,Ae,be,Fe,A,b,F,Ac,bc,Fc,w,wc,UpCenter,nx,...
    zero_tol,quadoptions);
    [errL,needSplitL(ii)] = SplitErr(C,Ac,bc,Fc,wc,PolyLows{ii},...
    FunL{ii},xtol,v_tol);
    [errU,needSplitU(ii)] = SplitErr(C,Ac,bc,Fc,wc,PolyUps{ii},...
    FunU{ii},xtol,v_tol);
    Err(ii) = 0.5 * (errL + errU);
end

if any(errid==0)
    flag = false;
    for jj = id'
        if ismember(jj,find(errid>0))
            minID = jj;
            ErrMin = Err(jj);
            flag = true;
            break;
        end
    end
    if ~flag
        error('No dimesion is suitable for division!');
    end
else
```

```
        [ErrMin, minID ] = min(Err,[],1);
    end

    shapeL = PolyLows{minID}.Internal.ub - PolyLows{minID}.Internal.lb;
    shapeU = PolyUps{minID}.Internal.ub - PolyUps{minID}.Internal.lb;
    respectRatioL = max(shapeL)/min(shapeL);
    respectRatioU = max(shapeU)/min(shapeU);

    if needSplitL(minID) | (respectRatioL >= maxRespectRatio)
        PolyLows{minID}.addFunction(Split,'Split');
    end
    if needSplitU(minID) | (respectRatioU >= maxRespectRatio)
        PolyUps{minID}.addFunction(Split,'Split');
    end
    L = PolyLows{minID}.addFunction(FunL{minID},'opt');
    U = PolyUps{minID}.addFunction(FunU{minID},'opt');

end

function [err,needSplit] = SplitErr(C,Ac,bc,Fc,wc,region,xFun,xtol,v_tol)

Points = region.V';
np = size(Points,2);
derr = zeros(np,1);
parfor ii=1:np
    xQp = xFun.Handle(Points(:,ii));
    [x,fval,exitflag,output,lambda] = NewPrimalDualMethod(C,Ac,bc+Fc...
    *Points(:,ii),wc);
    if exitflag < 0
        error('Infesible theta');
    end
    derr(ii) = norm(xQp-x);
end
err = sum(derr) / np;
needSplit = any(derr>=xtol);

try
    if region.volume <= v_tol
```

```
        needSplit = false;
    end
catch
    needSplit = false;
end
end
```

A.6.2 Modified WPDM Used for SME

Modified WPDM for the *E. coli* example.

```
function [opty,fval,exitflag,output,lambda] = NewPrimalDualMethod(c,A,b,w)

c_ = b;
A_ = A';
b_ = -c;

[tmp_m, tmp_n] = size(A_);


if size(c_,1) < size(c_,2) || size(b_,1) < size(b_,2) ||...
size(c_,1) ~= tmp_n || size(b_,1) ~= tmp_m
   error('LP: data dimensions does not match.');
end

%% parameter
maxIter = 1000;
verbose = 0;
tau = 1e-10;
maxDiag = 5.e+6;
etaMin  = .995;
factor = 0.7;

iterN = 0;
mu=10;
exitflag = 0;
```

```
if verbose == 0
%         warning('off','MATLAB:nearlySingularMatrix');
        warning('off');
else
        warning('on','all');
end

if isempty(w)
    w = ones(size(A_,2),1);
end
%% initial solution
e = ones(tmp_n,1);

% solution for min norm(s) s.t. A'*y + s = c
y0 = (A_*A_.')\(A_*c_);
s0 = c_-A_.'*y0;

% min norm(x) s.t. Ax = b_
x0 = A_.'*( (A_*A_.')\b_ );

% delta_x and delta_s
delta_x = max(-1.5*min(x0),0);
delta_s = max(-1.5*min(s0),0);

% delta_x_c and delta_s_c
pdct = 0.5*(x0+delta_x*e)'*(s0+delta_s*e);
delta_x_c = delta_x+pdct/(sum(s0)+tmp_n*delta_s);
delta_s_c = delta_s+pdct/(sum(x0)+tmp_n*delta_x);

% output
x = x0+delta_x_c*e;
s = s0+delta_s_c*e;
y = y0;

if any(isnan(x)) || any(isnan(s)) || any(isnan(y))
    x = x0;
    s = s0;
    y = y0;
```

```
end


%% calculate residual

Rd = A_.' * y + s - c_;
Rp = A_  * x - b_;
Rc = x.*s - mu*w;
residual = norm([Rp;Rd;Rc]);
mu = factor * mu;

if verbose == 2
    fprintf('%4s %9s %9s\n','ITER', 'MU', 'RESIDUAL');
end

%% iteratate

while true

    % Check Tolerance
    if mu < tau | residual<=1e-8
        exitflag = 1;
        mu = mu / factor;
        optx = x;
        opty = y;
        opts = s;
        break;
    end

    % Output
    if verbose == 2
        fprintf('%4d %9.2e %9.2e\n',iterN, mu, residual);
    end


    % Use augmented system to solve the directions
    rhs = sparse([-Rp; -Rd + Rc ./ x]);
```

```
% Set up the scaling matrix and form the coef matrix for normal equations
DD = min(maxDiag,-s ./ x);
B = [zeros(tmp_m,tmp_m) A_; A_.' diag(DD)];

% ldl' factorization
[L, D, pm] = ldl(B,'vector');

% Solve linear system
dxy = zeros(tmp_m + tmp_n, 1);
dxy(pm, :) = L'\(D\(L\(rhs(pm, :))));
if any(isnan(dxy))
    dxy = B^-1*rhs;
end

% Get the directions
dy = dxy(1:tmp_m);
dx = dxy( tmp_m + 1 : tmp_m + tmp_n );
ds = -( Rc + s .* dx ) ./ x;

% Get step length
eta = max(etaMin, 1-mu);

alphax = -1/min(min( dx ./ x),-1);
alphax = min(1, eta * alphax);

alphas = -1/min(min( ds ./ s),-1);
alphas = min(1, eta * alphas);

% Update the iterate
x = x + alphax*dx;
s = s + alphas*ds;
y = y + alphas*dy;

% Calculate residuals
Rd = A_.' * y + s - c_;
Rp = A_  * x - b_;
Rc = x.*s - mu*w;
residual = norm([Rp;Rd;Rc]);
```

```
    mu = factor * mu;

    % Increase counter
    iterN = iterN + 1;

    if norm(dx)>=1e16 | any(isnan([Rp;Rd;Rc]))
        break;
    end
end

if norm(dx)>=1e16 | any(isnan([Rp;Rd;Rc]))
    exitflag = -1;
    opty = NaN(size(y));
    fval = NaN;
    output = [];
    lambda = [];
    return;
end


if verbose == 2
    fprintf('Iteration = %u\n',iterN);
    fprintf('Function value = %9.2e\n',fval);
    fprintf('Residual = %9.2e\n',residual);
    fprintf('mu = %9.2e\n',mu);
    if exitflag==1
        disp('Optimization is terminated by tolerance');
    else
        disp('Optimization is terminated by maxIter');
    end
end

if verbose == 1
    fprintf('Function value = %9.2e\n',fval);
    if exitflag==1
        disp('Optimization is terminated by tolerance');
    else
        disp('Optimization is terminated by maxIter');
```

```
    end
end

output.tau = mu;
output.s = opts;
lambda = optx;
fval = -b_' * opty;

end
```

A.6.3 SME with WPDM

SME with WPDM for the *E. coli* example in fed-batch operation.

```
clc
startTime = datetime('now');


%% parameters calculation
syms y_bio y_vol z_glc z_oxy z_ace
syms rf dt z_fd kla GUR_max Km
Th = sym('Th',[5,1]);
Th(1) = (-rf*dt/(y_bio*y_vol)+1/y_bio)*z_glc + rf*dt*z_fd/(y_bio*y_vol);
Th(2) = ((1-kla*dt)/y_bio-rf*dt/(y_bio*y_vol))*z_oxy + 0.21*kla*dt/y_bio;
Th(3) = (-rf*dt/(y_bio*y_vol)+1/y_bio)*z_ace;
Th(4) = 100*GUR_max*z_glc/(Km+z_glc);
Th(5) = 100*(1-rf*dt/y_vol);
ThEq = matlabFunction(Th,'Vars',{z_glc,z_oxy,z_ace,y_bio,y_vol,...
GUR_max,Km,dt,kla,rf,z_fd});

z2Th = jacobian(Th, [z_glc;z_oxy;z_ace;y_bio;y_vol]);
z2ThEq = matlabFunction(z2Th,'Vars',{z_glc,z_oxy,z_ace,y_bio,y_vol,...
GUR_max,Km,dt,kla,rf,z_fd});

Coeff = [0     -9.46   -9.84   -19.23;...   %glc
    -35    -12.92  -12.73  0;...       %o2
    -39.43   0       1.24    12.12;...    %ace
```

```
    1        1      1        1];       %X

Km = 0.015;         % [mM]
GUR_max = 6.5;      % [mM/g-dw/hr]
OUR_max = 12;       % [mM/g-dw/hr]
kla = 4;            % [hr^-1]

Fmax = 0.003; % L/h

z_fd = 5;

% # of time steps and step size
nstate = 5;
nstep = 600;
tend = 15; % h, total time of cell culture growth, guess
dt = tend/nstep; % h, time, guess
maps = zeros(6,nstep);
Timeseries = (1:(nstep+1))*dt;

Constr = [-Coeff(2,:); Coeff(3,:); -Coeff(1:3,:)*dt;-Coeff(1,:);-Coeff(4,:)*dt];
b0 = [OUR_max; 100];

rng('default')
x0 = [0.4 0.21 0.2 0.001 0.3]'; %(glc [mM], o2[mM], ace[mM], X[mM], V[L])
lb = x0 - x0*0.05;
ub = x0 + x0*0.05;
x = lb + (ub-lb).*rand(length(x0),1);
Box.lb = lb;
Box.ub = ub;

mindex = [4 5];    %measurements, include volume
sigma = [0.02,0.0047,0.0100,0.01,7.5e-4].';
R = 0.05*diag((sigma(mindex)).^2);           %measurement noise covariance
Mlb = -[0.002 0.00105 0.001 1e-03 0.015];
Mub = [0.002 0.00105 0.001 1e-03 0.015];
SensorWrap = @(plantstate)Sensor(plantstate,mindex,R,Mlb,Mub);

state = zeros(nstate,nstep+1);
```

```
xlb = zeros(nstate,nstep+1);
xub = zeros(nstate,nstep+1);
y = zeros(length(mindex),nstep+1);
state(:,1) = x;
BoxSet = cell(nstep,2);
BoxSet{1,1} = Box;
ym = SensorWrap(x);
y(:,1) = ym;

scaling = 1;
xlb(:,1) = Box.lb;
xub(:,1) = Box.ub;

fig = figure('Position',[100, 100, 800, 220]);
fig.Color = [1 1 1];
subplot(1,3,1)
ylabel('Glucose');
hold on;
h1 = animatedline;
h1l = animatedline('Color','r');
h1u = animatedline('Color','r');

subplot(1,3,2)
ylabel('Oxygen');
hold on;
h2 = animatedline;
h2l = animatedline('Color','r');
h2u = animatedline('Color','r');

subplot(1,3,3)
ylabel('Acetate');
hold on;
h3 = animatedline;
h3l = animatedline('Color','r');
h3u = animatedline('Color','r');

flag = false;
global ii th time
```

```
time = 0;
dtflag = false;
Time = time;
ii = 2;
Set = [];


while time<=10

    if x(1)<=0.03
        flag = true;
    end
    if flag==false
        F = 0;
        P = 0;
    else
        F = 0.02;
        P = 0;
    end

    if time>=7.3 && dtflag == false
        dt = dt;
        dtflag = true;
    end

    time = time + dt;
    Time = [Time time];

    [x,~] = Plant(x,F,P,OUR_max,z_fd,dt,GUR_max,Km,kla,Coeff,Constr);
    state(:,ii) = x;

    [Box,set] = BoxPropogation(Box,ym,F,P,OUR_max,z_fd,dt,GUR_max,...
    Km,kla,Coeff,Constr,xlb(:,ii-1),xub(:,ii-1),Space,Mlb,Mub,z2ThEq,ThEq);
    Set = unique([Set; set]);
    BoxSet{ii,1} = Box;

    xlb(:,ii) = Box.lb;
    xub(:,ii) = Box.ub;
```

```
    if ~(all(xlb(:,ii)<=x) && all(xub(:,ii)>=x))
        disp(['Crossover:   ' num2str(time)]);
    end

    ym = SensorWrap(x);
    y(:,ii) = ym;

    addpoints(h1,time,x(1));
    addpoints(h2,time,x(2));
    addpoints(h3,time,x(3));

    addpoints(h1l,time,xlb(1,ii));
    addpoints(h2l,time,xlb(2,ii));
    addpoints(h3l,time,xlb(3,ii));

    addpoints(h1u,time,xub(1,ii));
    addpoints(h2u,time,xub(2,ii));
    addpoints(h3u,time,xub(3,ii));

    drawnow;

    ii = ii + 1;
end
disp(datetime('now')-startTime);

function [z,lambda] = Plant(z,rf,rp,OUR_max,z_fd,dt,GUR_max,Km,...
kla,A,Constr)
    global th nu state0 state1
    b1 = zeros(3,1);
    b0 = [OUR_max; 100];
    b1(1) = z(1)/z(4) - rf/z(5)*z(1)/z(4)*dt + rf/z(5)*z_fd/z(4)*dt;
    b1(2) = z(2)/z(4) - rf/z(5)*z(2)/z(4)*dt + kla*(0.21 - z(2))/z(4)*dt;
    b1(3) = z(3)/z(4) - rf/z(5)*z(3)/z(4)*dt;
    b2 = GUR_max*(z(1)/(Km + z(1)));
    b3 = 1 - (rf-rp)*dt/z(5);
    w = [5 1 1 1 1   1 1 1 1 1  1 1]';
    th = [b1;b2*100;b3*100];
```

```
    [nu,mu,exitflag,output,lambda] = NewPrimalDualMethod(-ones(4,1),...
    [Constr;-eye(4);1 1 1 1],[b0;b1;b2;b3;zeros(4,1);0.5],w);
    state0 = z;
    z = z + ODE(0,z,nu,rf,z_fd,rp,A,kla)*dt;
    z(z<=0)=0;
    state1 = z;

    function dzdt = ODE(t,z,nu,rf,z_fd,rp,A,kla)
        dzdt = zeros(5,1);
        dzdt(1:4) = A*nu*z(4) - rf/z(5)*z(1:4);
        dzdt(1) = dzdt(1) + rf/z(5)*z_fd;
        dzdt(2) = dzdt(2) + kla*(0.21 - z(2));
        dzdt(4) = dzdt(4) + z(4)*rp/z(5);
        dzdt(5) = rf - rp;
    end
end

function [BoxNew,NonemptID] = BoxPropogation(Box,y,rf,rp,OUR_max,z_fd,...
dt,GUR_max,Km,kla,A,Constr,xlb,xub,CRs,Mlb,Mub,z2ThEq,ThEq)
    global ii th nu state0 state1 time

    yB_lb = y(1)-Mub(4) ;
    if yB_lb<0
        yB_lb = eps;
    end
    yB_ub = y(1) - Mlb(4);
    yV_lb = y(2) - Mub(5) ;
    yV_ub = y(2) - Mlb(5) ;

    ub = Box.ub;
    lb = Box.lb;
    lb(lb<0) = 0;
    ub(ub<0) = eps*1e6;
    lb(4) = yB_lb;
    ub(4) = yB_ub;
    lb(5) = yV_lb;
    ub(5) = yV_ub;
```

```
Perm = de2bi(0:31,5,'left-msb');
Perm(Perm(:,1)==0,1) = lb(1);
Perm(Perm(:,1)==1,1) = ub(1);
Perm(Perm(:,2)==0,2) = lb(2);
Perm(Perm(:,2)==1,2) = ub(2);
Perm(Perm(:,3)==0,3) = lb(3);
Perm(Perm(:,3)==1,3) = ub(3);
Perm(Perm(:,4)==0,4) = lb(4);
Perm(Perm(:,4)==1,4) = ub(4);
Perm(Perm(:,5)==0,5) = lb(5);
Perm(Perm(:,5)==1,5) = ub(5);
Box = Perm.';

boxV = 0.5*(lb+ub);
Th = ThEq(boxV(1),boxV(2),boxV(3),boxV(4),boxV(5),GUR_max,...
Km,dt,kla,rf,z_fd);
z2Th = z2ThEq(boxV(1),boxV(2),boxV(3),boxV(4),boxV(5),GUR_max,...
Km,dt,kla,rf,z_fd);
theta = z2Th*(Box-boxV) + Th;
thetaVmax = max(theta,[],2);
thetaVmin = min(theta,[],2);
thetaLen =  thetaVmax - thetaVmin;

Box = Polyhedron('lb',lb,'ub',ub);
theta = z2Th*(Box-boxV) + Th;
theta.minVRep();
theta.computeHRep();

r = rank(z2Th);
ASet = [A;0 0 0 0] * dt;


len = size(CRs,1);
determinant = zeros(len,1);
index = 1;
thetaCR = {};
for jj = 1:len
    intersection = intersect(CRs(jj),theta);
```

```
    intersection.computeVRep();
    determinant(jj) = ~intersection.isEmptySet;
    intV = intersection.V';
    if determinant(jj)
       if any(max(intV,[],2)>=thetaVmax*1.00001)
            scaleIn = [100;100;100;100;100]./thetaLen;
            scaleIn(isinf(scaleIn)) = 1;
            ScaleIn = diag(scaleIn);
            ScaleInInv = ScaleIn^-1;
            thetaScaledIn = ScaleIn*theta;
            intersection = intersect(ScaleIn*CRs(jj),thetaScaledIn);
            intersection.computeVRep();
            determinant(jj) = ~intersection.isEmptySet;
            if determinant(jj)
                 intV = ScaleInInv * intersection.V';
                 if any(max(intV,[],2)>=thetaVmax*1.00001)
                    warning('Obev bounds');
                 end
                 thetaCR{index} = intV;
                 index = index + 1;
            end
       else
          thetaCR{index} = intV;
          index = index + 1;
       end
    end
end
NonemptID = find(determinant>0);
Nonemptlen = length(NonemptID);


thetaDecmpTrans = cell(Nonemptlen,1);
thetaDecmpTransProj = cell(Nonemptlen,1);
zRecover = {};
z = {};
pseg = {};
for kk=1:Nonemptlen
    thetaDecmpTrans{kk} = thetaCR{kk} - Th;
```

```
    if r==4
        thetaDecmpTransProj{kk} = thetaDecmpTrans{kk}(1:4,:);
        zRecover{kk} = (z2Th(1:4,1:4)^-1 * thetaDecmpTransProj{kk}...
        + boxV(1:4))';
        z{kk} = [zRecover{kk} repmat(yV_lb,size(zRecover{kk},1),1);...
        zRecover{kk} repmat(yV_ub,size(zRecover{kk},1),1)];
    elseif r==5

        z{kk} = (z2Th^-1 * thetaDecmpTrans{kk} + boxV)';
    else
        error('rank problem!');
    end


    F = CRs(NonemptID(kk)).getFunction('opt').F;
    g = CRs(NonemptID(kk)).getFunction('opt').g;

    BioL = min(z{kk}(:,4));
    BioU = max(z{kk}(:,4));
    xest = mean(z{kk},1);
    term1 = F*z2Th;

    TSet = diag([1 - rf*dt/xest(5); 1 - rf*dt/xest(5)-kla*dt;...
    1 - rf*dt/xest(5); 1 - (rf-rp)*dt/xest(5); 1]);
    TransSet = [z_fd*rf*dt/xest(5); 0.21*kla*dt; 0; 0; (rf-rp)*dt];

    Conv = {};
    Conv{1} = (TSet+BioL*ASet*term1)*z{kk}' + BioL*ASet*...
    (-term1*boxV+F*Th+g)+TransSet;
    Conv{2} = (TSet+BioU*ASet*term1)*z{kk}' + BioU*ASet*...
    (-term1*boxV+F*Th+g)+TransSet;
    pseg{1}(:,kk) = min([Conv{1} Conv{2}],[],2);
    pseg{2}(:,kk) = max([Conv{1} Conv{2}],[],2);


end

lb0 = min(pseg{1},[],2);
```

```
    lb0(lb0<0) = eps;
    ub0 = max(pseg{2},[],2);
    ub0(ub0<0) = eps;
    lb1 = lb0;
    ub1 = ub0;
    conver = lb0<=1e-6;
    lb1(conver) = eps;
    conver1 = ub0<=9e-3;
    ub1(conver1) = 9e-3;
    BoxNew.lb = lb1;
    BoxNew.ub = ub1;

end

function stateobserved = Sensor(plantstate,mindex,R,Mlb,Mub)
    stateobserved = plantstate(mindex);
    noise = mvrandn(Mlb(mindex),Mub(mindex),R,1);
    stateobserved = plantstate(mindex) + noise;
end
```

A.7 Codes related to ERM

A.7.1 ERM

```
function [x,fval,exitflag] = ER6(C,A,b,Ae,be,lb,ub,dir2)
if length(C)>1
    if size(lb,2)>1
        lb = lb.';
    end
    if size(ub,2)>1
        ub = ub.';
    end
    if size(C,2)>1
        C = C.';
    end
end
```

```
try
    ConstraintTolerance = 1e-6;
    options = optimoptions('linprog','Algorithm','dual-simplex',...
    'Display','off','ConstraintTolerance',ConstraintTolerance);
    [x0,fval0,exitflag,output,lambda] = linprog(C,A,b,Ae,be,lb,ub,options);
catch
    [x0,fval0,exitflag,output,lambda] = cplexlp(C,A,b,Ae,be,lb,ub);
end


if exitflag>=0 && ~isempty(x0)

    n_ine = length(b);
    n_lb = length(lb);
    n_ub = length(ub);

    if n_lb
        act_lb = find(lambda.lower>0);
    else
        act_lb = [];
    end

    if n_ub
       act_ub = find(lambda.upper>0);
    else
       act_ub = [];
    end

    if n_ine
       act_ine = find(lambda.ineqlin>0);
    else
       act_ine = [];
    end

    iact_ine = setdiff( 1:n_ine, act_ine);
    iact_lb = setdiff( 1:n_lb , act_lb);
    iact_ub = setdiff( 1:n_ub, act_ub);
```

```
Idm = eye(length(C));
A_ = [A(iact_ine,:); -Idm(iact_lb,:); Idm(iact_ub,:)];
b_ = [b(iact_ine,:); -lb(iact_lb,:); ub(iact_ub,:)];

Ae_ = [A(act_ine,:); Idm(act_lb,:); Idm(act_ub,:); Ae;C.'];
be_  = [b(act_ine,:); lb(act_lb,:); ub(act_ub,:); be;fval0];

QPoptions = optimoptions('quadprog','Algorithm',...
'interior-point-convex','Display','off','ConstraintTolerance',...
ConstraintTolerance*100,'StepTolerance',1e-10,...
'OptimalityTolerance',1e-6);
warning('off','optim:quadprog:HessianNotSym');

if any(abs(be_)<=1)

     scaledir = Ae_*C;
     v_min = min(abs(be_));
     idx_min = find(abs(be_)==v_min);
     if min(abs(scaledir(idx_min)))==0
         scale = 100;
     else
         scale = 1./min(abs(scaledir(idx_min)));
     end

     b_ = A_*C*scale + b_;
     be_ =  scaledir*scale + be_;

     y0 = x0+C*scale;

    if isempty(dir2)
        dir2 = y0./norm(y0);
    else
        if norm(dir2)~=0
            dir2 = dir2 +C*scale;
            dir2 = dir2./norm(dir2);
        else
            dir2 = y0./norm(y0);
```

```
            dir2 = dir2 +C*scale;
            dir2 = dir2./norm(dir2);
        end
    end

    dir1 = zeros(length(C),1);
    dir1(1) = 1;
    dir = dir1 - dir2;
    H = eye(length(C)) - 2*(dir*dir.')*(dir.'*dir)^-1;
    base = Idm;
    base(1,1) = 1e-6;
    Q = H*base*H.';
    Q = 0.5*(Q+Q.');

    try
        y = quadprog(Q,zeros(length(C),1),A_,b_,Ae_,...
        be_,[],[],y0,QPoptions);
    catch
        y = cplexqp(Q,zeros(length(C),1),A_,b_,Ae_,be_,...
        [],[],y0);
    end

    try
        x = y - C*scale;
    catch
        x = x0;
    end
    exitflag = 1;
    fval = C.'*x;

else
    if isempty(dir2)
        dir2 = x0./norm(x0);
    else
        if norm(dir2)~=0
            dir2 = dir2./norm(dir2);
        else
            dir2 = x0./norm(x0);
```

```
                dir2 = dir2./norm(dir2);
            end
        end

        dir1 = zeros(length(C),1);
        dir1(1) = 1;
        dir = dir1 - dir2;
        H = eye(length(C)) - 2*(dir*dir.')*(dir.'*dir)^-1;
        base = Idm;
        base(1,1) = 1e-6;
        Q = H*base*H.';
        Q = 0.5*(Q+Q.');

        try
            try
               x = quadprog(Q,zeros(length(C),1),A_,b_,Ae_,...
               be_,[],[],x0,QPoptions);
            catch
               x = cplexqp(Q,zeros(length(C),1),A_,b_,Ae_,...
               be_,[],[],x0);
            end
        catch
            x = x0;
        end

        fval = C.'*x;
exitflag = 1;
    end

else
    x = x0;
    fval = [];
exitflag = -1;
    return;
end

end
```

A.7.2 Example

Example of

```
Vertex = [4 0 2; 0 4 4; 0 4 0; 4 0 0; 0 0 0; 0 0 4];

fig = figure('Position',[100, 100, 452+300, 477]);
fig.Color = [1 1 1];
ax = axes('Position',[0.1 0.12 0.5 0.8]);
ax.ActivePositionProperty = 'outerposition';
ax.GridLineStyle = '--';
ax.LineWidth = 1.5;
ax.FontSize = 16;


patch('Faces',[3 4 5],'Vertices',Vertex,'EdgeColor',...
'black','FaceColor',[0.259 0.526 0.957],'LineWidth',0.1,...
'FaceAlpha',0,'EdgeAlpha',1);
patch('Faces',[5 4 1 6],'Vertices',Vertex,'EdgeColor',...
'black','FaceColor',[0.259 0.957 0.467],'LineWidth',0.1,...
'FaceAlpha',0,'EdgeAlpha',1);
patch('Faces',[6 2 3 5],'Vertices',Vertex,'EdgeColor',...
'black','FaceColor',[0.957 0.945 0.259],'LineWidth',0.1,...
'FaceAlpha',0,'EdgeAlpha',1);
patch('Faces',[3 4 1 2],'Vertices',Vertex,'EdgeColor',...
'black','FaceColor',[0.3010 0.7450 0.9330],'LineWidth',0.1,...
'FaceAlpha',0.5,'EdgeAlpha',1);
patch('Faces',[2 6 1],'Vertices',Vertex,'EdgeColor',...
'black','FaceColor',[0.679 0.259 0.957],'LineWidth',0.1,...
'FaceAlpha',0,'EdgeAlpha',1);
xlabel('\it x_1','FontSize',18,'FontName','Times New Roman',...
'FontWeight','bold','position',[2.34 -0.456 -0.254]);
ylabel('\it x_2','FontSize',18,'FontName','Times New Roman',...
'FontWeight','bold','position',[-0.70 2.24 -0.135]);
zlabel('\it x_3','FontSize',18,'FontName','Times New Roman',...
'FontWeight','bold','position',[-1.6 2.17 3.41]);
```

```
hold on;
grid on;


view(-32.2,32.8);
hold on;

%% path 1
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
list = zeros(num,3+3+1);

count = 1;
for ii = 1:num
    r1 = [1.44+2^(ii-1); 2.56; 1.64];
    [x,fval,exitflag] = ER6(c,A,b,[],[],[0;0;0],[],r1);
    list(count,:) = [r1; x; fval].';
    count = count + 1;
end

sct1 = scatter3(list(:,4),list(:,5),list(:,6),50,[0 0 0],...
'p','MarkerEdgeColor',[0 0 0],'LineWidth',1);

%% path 2
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
list = zeros(num,3+3+1);

count = 1;
for ii = 1:num
    r1 = [1.44; 2.56+2^(ii-1); 1.64];
    [x,fval,exitflag] = ER6(c,A,b,[],[],[0;0;0],[],r1);
    list(count,:) = [r1; x; fval].';
    count = count + 1;
```

```
end

sct2 = scatter3(list(:,4),list(:,5),list(:,6),50,[0 0 0],...
'*','MarkerEdgeColor',[0 0 0],'LineWidth',0.5);

%% path 3
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
list = zeros(num,3+3+1);

count = 1;
for ii = 1:num
    r1 = [1.44; 2.56; 1.64*1.08^(ii-1)];
    [x,fval,exitflag] = ER6(c,A,b,[],[],[0;0;0],[],r1);
    list(count,:) = [r1; x; fval].';
    count = count + 1;
end

sct3 = scatter3(list(:,4),list(:,5),list(:,6),50,[0 0 0],...
'O','MarkerEdgeColor',[0 0 0],'LineWidth',1);

%% path 4
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
list = zeros(num,3+3+1);

count = 1;
for ii = 1:num
    r1 = [1.44+2^(ii-1); 2.56+2^(ii-1); 1.64];
    [x,fval,exitflag] = ER6(c,A,b,[],[],[0;0;0],[],r1);
    list(count,:) = [r1; x; fval].';
    count = count + 1;
end
```

```
sct4 = scatter3(list(:,4),list(:,5),list(:,6),50,[0 0 0],...
's','MarkerEdgeColor',[0 0 0],'LineWidth',2);

%% path 5
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 20;
list = zeros(num,3+3+1);

count = 1;
for ii = 1:num
    r1 = [1.44+2*(ii-1); 2.56; 1.64+1*(ii-1)];
    [x,fval,exitflag] = ER6(c,A,b,[],[],[0;0;0],[],r1);
    list(count,:) = [r1; x; fval].';
    count = count + 1;
end

sct5 = scatter3(list(:,4),list(:,5),list(:,6),50,[0 0 0],...
'd','MarkerEdgeColor',[0 0 0],'LineWidth',1);

%% path 6
c = [-2 -2 0].';
A = [1 1 0; 1 0 2];
b = [4 8].';
num = 10;
list = zeros(num,3+3+1);

count = 1;
for ii = 1:num
    r1 = [1.44; 2.56*1.5^(ii-1); 1.64*1.57^(ii-1)];
    [x,fval,exitflag] = ER6(c,A,b,[],[],[0;0;0],[],r1);
    list(count,:) = [r1; x; fval].';
    count = count + 1;
end

sct6 = scatter3(list(:,4),list(:,5),list(:,6),50,[0 0 0],...
'x','MarkerEdgeColor',[0 0 0],'LineWidth',1);
```

```
%%
hL = legend([sct1,sct2,sct3,sct4,sct5,sct6],{'Increasing r_1(1)',...
'Increasing r_1(2)',.'Increasing r_1(3)','Increasing r_1(1) and r_1(2)',...
    'Increasing r_1(1) and r_1(3)','Increasing r_1(2) and r_1(3)'});
hL.EdgeColor = [1 1 1];
hL.FontSize = 14;
hL.FontName= 'Times New Roman';
hL.FontWeight= 'bold';

newPosition1 = [0.7 0.35 0.2 0.25];
newUnits1= 'normalized';
set(hL,'Position', newPosition1,'Units', newUnits1);
```

Glossary

DFBA Dynamic Flux balance analysis v

DFBM Dynamic Flux balance model v

DO Dissolved oxygen 24

EKF Extended Kalman filter vi

ERM Ellipsoidal reflection method v

FBA Flux balance analysis 21

HO Hierarchical optimization 27

IPM Interior point method v

KKT Karush–Kuhn–Tucker conditions 14

LP Linear programming v

MCA Monte Carlo algorithm 74

MNAR Minimization of the number of active reactions 42

mpLP Multiparametric linear programming vi

mpNLP Multiparametric nonlinear programming vii, 5

mQP Multiparametric quadratic programming 17

MTF Minimization of the total flux 42

NLP Nonlinear programming 81

NPDM Naive primal-dual method 25

QP Quadratic programming vi

SME Set membership estimation vi

SSE Summation of Squared Errors 42

TN Truncated multivariate normal distribution 82

VSS Variable structure system vi

WPDM Weighted primal-dual method v