Robust Nonlinear Model Predictive Control of Biosystems described by Dynamic Metabolic Flux Models

by

Honghao Zheng

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Chemical Engineering

Waterloo, Ontario, Canada, 2019

© Honghao Zheng 2019

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this 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.

Abstract

The accuracy of the model used for prediction in Nonlinear Model Predictive Controller (NMPC) is one of the main factors affecting the closed loop performance. Since it is impossible to formulate a perfect model for a real process, there are always differences between the responses predicted by the model and the responses observed from the process. Hence, robustness to model error is an essential property that the controller must have to be adopted in industrial applications. Propagating the uncertainty in the model onto the variables used by the controller is one the key challenges for efficient implementation of a robust controller. Uncertainty propagation approaches such as Monte Carlo simulations and the Polynomial Chaos Expansions (PCE) has been found to suffer from exponentially increasing computational effort with the number of uncertain parameters. Accordingly, the main goal of this thesis is to develop a novel formulation of NMPC based on an uncertainty propagation approach that is more computationally efficient as compared to previously reported approaches. The proposed robust controller in this thesis is specifically targeted to biosystems that are modeled by Dynamic Metabolic Flux models. These models that are becoming increasingly popular for modelling bioprocesses are based on the premise that microorganisms have learned through natural evolution to optimally allocate their resources (nutrients) to maximize a biological objective such as growth or ATP production. Accordingly, these flux models are formulated by LP (Linear Programming) optimizations with constraints that are solved at each time interval and then can be used in conjunction with mass balances to predict the dynamic evolution of different metabolites. The uncertainty in these models is associated to inaccuracies of model parameters involved in the constraints. Thus, although the problem can be solved for particular model parameters by an LP, in the presence of uncertainty the problem becomes nonlinear since different active sets of constraints may become active for parameters' values within their possible range of variation. Accordingly, the solution space of this nonlinear system can be divided into a set of polyhedrons where each point corresponds to a particular set of parameters within their range of uncertainty. The solution space is often referred in the thesis as the RHS (Right Hand Side) space since it is defined by the variations in the RHS of the constraints with respect to the uncertain parameters. To identify these polyhedrons a dividing procedure has been developed. Since all the polyhedrons can be proven to be convex cones based on a standard simplex form of LP, this dividing method is referred to as the Convex Cone Method

(CCM). The regions found by the CCM method are then compared to regions calculated with 100 Percent Rule where the latter has been often used to find a region of existence of a particular tableau in the Simplex method. From this comparison it is found that the CCM can both identify all the possible tableaus with a given region of uncertain parameters and it can also be used the probability for occurrence of each one of the tableaus. These two facts make the CCM an attractive basis for uncertainty propagation in an LP problem instead of the 100 Percent Rule.

After identifying the possible tableaus for a given region of model parameters, a novel method is developed for propagating uncertainty onto the controlled variables to be referred to as Tableau Based Tree (TBT) method. The TBT method is based on the concept of propagating uncertainty into the prediction horizon of the controlled by using a tree structure which branches correspond to different tableaus identified by the CCM approach. It is then shown that the conservativeness of the NMPC controller can be significantly reduced based on this tree structure as compared to a Monte Carlo approach for uncertainty propagation. After propagating the uncertainty onto the relevant variables, the control actions for each branch of the tree structure can be obtained by a simple linear calculation. An EMPC (Economic Model Predictive Controller) is adopted in this work as a special realization of an NMPC algorithm where the controller pursues the maximization of an economic objective function. A simple theoretical comparison with a Monte Carlo uncertainty propagation approach shows that the TBT method have a potential to save considerable computational effort as compared to Monte Carlo simulation and PCEs. Finally, the TBT-based robust EMPC is applied in a case study dealing with a fed-batch reactor which is described by dynamic metabolic flux model (DMFM).

Acknowledgements

Firstly, I would like to express the most sincerely thanks to my supervisor Professor Hector Budman and Professor Luis Ricardez-Sandoval for their inspiration, guidance and support during my entire master's program. Thank you for your kindness, trust and patience, it really provides me much for my free study environment.

I would also like to extend my thanks to the readers of my thesis.

I would like to thank my office-mate Piyush Agarwal, Yue Yuan for their continued assistance and companionship throughout my research studies.

I am also grateful to all of the members from Professor Hector Budman's and Professor Luis Ricardez-Sandoval's group, all the staffs and faculty members of the Department of Chemical Engineering at the University of Waterloo.

Finally, I would like to express my deepest gratitude to my parents who has always been there for me.

Dedication

To my father, my mother and other family members.

I would not be here without your support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BF	Basic feasible
CCM	Convex cone method
CSTR	Continuous stirred-tank reactor
deFBA	Dynamic enzyme-cost flux balance analysis
DFBM	Dynamic flux balance modeling
E. coli	*Escherichia coli*
EMPC	Economic model predictive control
FBA	Flux balance analysis
LHS	Left hand side
LMI’s	Linear matrix inequalities
LP	Linear programming
MC	Monte Carlo
MFA	Metabolic flux analysis
MPC	Model predictive control
NLP	Nonlinear programming
NMPC	Nonlinear model predictive control
ODE	Ordinary differential equation
PCE	Polynomial chaos expansions
PI	Proportional and Integral Controller
QP	Quadratic programming
RHS	Right hand side
RTO	Real time optimization
SAA	Sample average approximation
SSV	Structured singular value
TBT	Tableau based tree

List of Symbols

List of English symbols

a_{ij}	Left hand side parameters of constraints
$\boldsymbol{A}$	The matrix of a_{ij}
$\boldsymbol{A^p}$	The square matrix selected form $[\boldsymbol{A} \quad \boldsymbol{I}]$ and corresponding to a certain $\boldsymbol{x}_{nc}^{p}$
$\boldsymbol{\mathcal{A}}$	Stoichiometric matrix
b_i	Right hand side parameters of constraints
$\boldsymbol{b}$	The column vector of b_i
$\boldsymbol{\mathscr{b}}$	Vector of consumption or production rate of extracellular metabolites
$\boldsymbol{B}$	Basis matrix
c_j	Parameters of the objective function
$\boldsymbol{c_B}$	Vector of objective function coefficients with zeros for slack variables
$\boldsymbol{c}$	The row vector of c_j
$\boldsymbol{c}_{nc}^{p}$	The coefficients of the vector $[-\boldsymbol{c} \quad \boldsymbol{0}]$ and corresponding to a certain $\boldsymbol{x}_{nc}^{p}$
$\mathcal{C}$	The amount of all possible tableaus in the RHS space
d	Uncertainty or disturbance
$\boldsymbol{d}$	The vector of uncertainty
e_i	An edge constraint of a polyhedron
$\boldsymbol{f_{ac}}$	The additional constraints of CCM
f_{bk}	Feed back
f_{nck}	A function that can generate a set $\boldsymbol{\mathcal{T}}$
F	Feed rate
F_b	Full complex blocks
$F(x)$	Linear matrix inequalities based analyses
GUR_{max}	The maximum uptake rate constraints
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{I}_B$	The set of possible indices of branches $\boldsymbol{I}_B(s,r)$ in the TBT scenario tree
$\boldsymbol{I}_B^{na}$	The set of non-active branches $\boldsymbol{I}_B^{na}(s,r)$ in the TBT scenario tree
$\boldsymbol{I}_B^{act}$	The set of active branches $\boldsymbol{I}_B^{act}(s,r)$ in the TBT scenario tree

k_La	The oxygen mass transfer coefficient
k_T	The number of branches of a certain node
K	A constant parameter
K_m	Substrate saturation constant
$\mathcal{K}_{RS}$	The number of active tableaus in the uncertainty region
$\mathcal{K}_{MC}$	The amount of the main tableaus that are active in RS
l	The time where each controller inputs u can be manipulated
m	Control horizon
M	Interconnection matrix for SSV analysis
$\boldsymbol{MR_{mc}}$	A main tableau
$\boldsymbol{M_{RHS}}$	RHS map
N_M	Number of the repeating random sampling in each time interval
OR_i	An overlap region is defined of two different $\boldsymbol{TR_i}$
OUR_{max}	The maximum rate of oxygen uptake reaction
p	Prediction horizon
$\mathscr{p}_i$	Vertex of a polyhedron region
$\boldsymbol{\mathscr{p}_{set}}$	A set of m points along the axes corresponding to $\Delta\boldsymbol{b_{max}}$ in a certain orthant
P	Probability
P_c	Product concentration
$\boldsymbol{P_{TR_i}}$	Probability of $\boldsymbol{TR_i}$
$\boldsymbol{\mathcal{P}_i}$	A polyhedron region
r_μ	Robust horizon
r_F	Feed rate
r_P	Perfusion rate
RS	RHS space of $\boldsymbol{b}$
$RS\|_{\Delta\boldsymbol{b}}$	The region RS with respect of $\Delta\boldsymbol{b}$
s	A stage in the TBT method
S	Substrate concentration
$SRS\|_{\Delta\boldsymbol{b}}$	The overall sensitivity range of $RS\|_{\Delta\boldsymbol{b}}$

S_b	Scalar blocks
$\boldsymbol{SR_{nc}}$	Sensitivity ranges of $\boldsymbol{TR_{nc}}$ with uncertainty range of $\Delta\boldsymbol{b}$
$\boldsymbol{S}_{\boldsymbol{\psi}_s^r}$	All of the possible states that are generated from each node $\boldsymbol{\psi}_s^r$
t	Time
t_f	Final time of the process
T_{nc}	A certain tableau corresponding to a particular $\boldsymbol{x_{nc}^p}$
$\boldsymbol{TR_{nc}}$	A region corresponding to a certain tableau in RHS space
$\boldsymbol{\mathcal{T}}$	A set consist all possible $\boldsymbol{x}_{nc}^p$'s combinations of tableaus T_{nc}
u	Manipulated variables or control actions
$\mathbb{U}_{\boldsymbol{f}}$	Constraints' set of u
$\boldsymbol{U}_L$	Index of the input values u of this model
V	Batch volume
V_f	Economic stage cost with a terminal cost
$V_{\mathcal{P}_i}$	Volume of $\boldsymbol{\mathcal{P}_i}$
$Vol\vert_{\Delta\boldsymbol{b}}$	The volumes of every sub-tableau in the region $\Delta\boldsymbol{b}$
$\boldsymbol{VR_{nc}}$	Vertices set of T_{nc}
$\boldsymbol{\mathcal{V}}$	A convex cone
$\vec{V}_{\hbar}$	Affine vectors that start from the origin $\boldsymbol{O}$
w_i	Amounts of growth precursors required per gram of biomass
x	Plant states, decision variables in optimization
X	Biomass concentrations
$\boldsymbol{x_s}$	Column vector of slack variables
$\boldsymbol{x_B}$	Vector of basic variables
$\hat{\boldsymbol{x}}$	Prediction of plant states
$\boldsymbol{x_{nc}^p}$	One of the possible solution vectors corresponding to a point $\boldsymbol{b}$ in RS
$\boldsymbol{X^p}$	Matrix of all of the possible solution vectors corresponding to a point $\boldsymbol{b}$ in RS
$\mathbb{X}_f$	Terminal set of plant states profile
y	Plant output
$\hat{y}$	Prediction of plant output

$\boldsymbol{y}^{sp}$	Set-point trajectory
$Y_{P\|S}$	Yield coefficient of product from substrate
$Y_{X\|S}$	Yield coefficient of biomass from substrate
$\boldsymbol{z}$	Current metabolites' concentrations
Z	Cost, value of overall measure of performance
Z_{nc}^{p}	The cost corresponding to each possible solution vector of a point $\boldsymbol{b}$ in RS

List of Greek symbols

$\boldsymbol{\alpha_{nc}}$	A set of coefficients corresponding to the tableau T_{nc}
$\alpha_{I,s}$	The number of active scenarios in stage s
$\boldsymbol{\beta}$	The main source of uncertainty in the model of current work
$\bar{\sigma}$	Perturbation within the given uncertainty set
Δ	Uncertainty description
$\mathcal{E}$	Tuning parameter of N_M
μ_{Δ}	Measure of the smallest perturbation within the given uncertainty set
δ_i	Certain pre-specified error
μ	Rate of cell growth
μ_{max}	Maximum growth rate
τ	Discrete-time of an MPC controller model
τ_c	Time consumed for each scenario
$\boldsymbol{\nu}$	Vector of reaction fluxes
θ	A set of the points that within the simplex generated region
Φ	Additional constrains in $\boldsymbol{f_{ac}}$
$\boldsymbol{\Psi_f}$	Constraints' set of $\boldsymbol{\psi}$
$\boldsymbol{\psi}$	Vector of plant states, such as the concentrations of metabolites
ψ_{obj}	A specific element from $\boldsymbol{\psi}$ as an objective function
ζ	The amount of uncertainty parameters in the system
$\boldsymbol{\rho}$	An upper triangular matrix used for the division of the non-convex region
$\boldsymbol{\rho_{km}}$	The $(km-1)$th column vector from $\boldsymbol{\rho}$

Chapter 1
Introduction

Automation is prevalent in every aspect of modern life, such as energy, technology, fertility, security and productivity. Controllers can be used to attain quality control and improve productivity while satisfying operational, safety and environmental constraints. For instance, controllers in the chemical industry can be used to produce desired products while minimizing the utilization of raw materials and energy and reduce the production of environmentally adverse(Gutierrez, Ricardez-Sandoval, Budman, & Prada, 2014; Mehta & Ricardez-Sandoval, 2016; Patil, Maia, & Ricardez-Sandoval, 2015) By using an optimized model-based control system, the process can be manipulated under physical, environmental and economical constraints and be able to maintain a certain level of robustness to disturbances arising from the environment, to uncertainty in the process measurements or model parameters or to variability in the plant's input, e.g. quality of the raw materials. In this thesis, the robustness of an Economic Model Predictive Controller (EMPC), a particular version of Model Predictive Control (MPC), is investigated. (He, Sahraei, & Ricardez-Sandoval, 2016; Rasoulian & Ricardez-Sandoval, 2016; Santander, Elkamel, & Budman, 2016)

A conventional MPC is a control methodology that provides a set of future moves in the control actions so as to minimize the variability of the controlled variables with respect to the targets while respecting constraints in both input and output variables. In MPC the optimal future control actions are calculated based on predicted trajectories of the controlled variables using a model. The control actions are calculated from a constrained optimization problem by taking into account the physical limits of the system and its parameters such as inputs and outputs. In most cases, the applications of MPC in industry practices are based on empirical linear models which are derived from experimental data or linearization of mechanistic (nonlinear) process models. While easy to implement, it has been shown that the application of linear models in the MPC formulation may produce a loss in plant's performance, particularly when the process that need to be controlled exhibits highly nonlinear dynamic behavior. Therefore, a nonlinear model is required for accurate prediction. The resulting predictive control strategy that is based on the nonlinear model is referred to as Nonlinear Model Predictive Control (NMPC).

MPC, as a typical multivariable controller, holds a key position in the organization of an advanced process system. The integration of optimization and control in chemical plants is generally achieved by hierarchical control structures that are typically of the form shown in Figure 1.1 (Ellis, Liu, & Christofides, 2017). This structure contains a Real Time Optimization (RTO) layer at the top of the pyramid a second layer that involves MPC or other multivariable control strategies and finally a lower level involving singular input/output regulatory controllers, e.g. PID controllers. In most cases, the RTO is generally used to optimize the economic cost of a system based on a steady state model. This RTO level will provide optimal set-points to the multivariable controllers, e.g. MPC, that are expected to maintain the plant on target. Thus, MPC aims to control the process around the set-points which are generated by RTO. Subsequently, the multivariable controller computes the values of the control actions either directly or indirectly by calculating the set-points for the PID controllers in the lower level. This hierarchical strategy generally maintains satisfactory performance. However, since chemical process are rarely at steady state, the set-points designed by the RTO, and enforced by MPC controller, may be suboptimal or even infeasible in the presence of transient scenarios (Santander et al., 2016). Following these considerations, there is a great incentive for improving the performance within this hierarchical control approach.

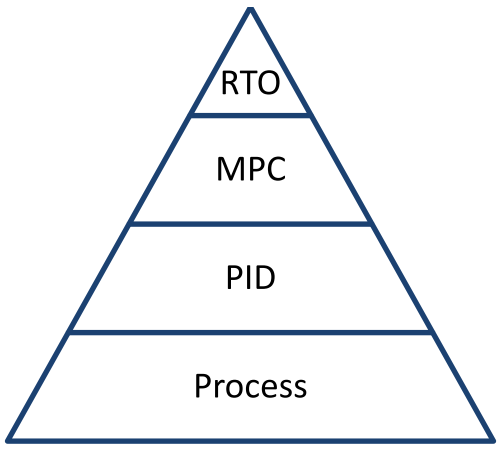


Figure 1.1 Hierarchical structure in process operations

When discussing model uncertainty, it is inherently assumed that there exists an accurate mathematical model that describes the process but this model is never available to the user due to the use of noisy data for model calibration, the limited amount of data for calibration and incomplete physical knowledge about the process phenomena. As a result of that the process models available to the user are never perfect and there is always model structure and/or parameter uncertainty that is collectively referred to as plant-model mismatch. Since predictive controllers rely heavily on models robustness to model error must be provided in order to ensure adoption of these controllers in industrial applications (Findeisen, Imsland, Allgower, & Foss, 2003; Lalo Magni & Scattolini, 2010). Accurate and fast uncertainty propagation onto the variables involved in the control strategy is the key element for formulating an efficient robust controller. Currently reported methods such as the Monte Carlo simulations (Kawohl, Heine, & King, 2007), Polynomial Chaos Expansions (PCE) (Ghanem & Spanos, 1990; Kumar & Budman, 2017; Mandur & Budman, 2012) or Power Series Expansions (PSE) (Rasoulian & Ricardez-Sandoval, 2015) cannot avoid the exponentially increasing computational load needed to estimate control actions as a function of the number of model uncertain parameters. Thus, developing a NMPC framework that is insensitive to the number of the uncertain parameters is one of the main objectives of this thesis.

1.1 Objectives of the Research

As discussed above, the motivation of current research is the development of a novel formulation of NMPC which requires less computational effort than conventional NMPC controllers. In the current research we have focused on bioprocesses that are modelled by a particular modelling approach referred to as Dynamic Metabolic Flux Model (DMFM). These models assume that organisms are able to allocate resources optimally so as to achieve a biological objective, e.g. growth rate. Accordingly, DMFM are formulated as constrained Linear Programming (LP) problems. The uncertainty in the problem is captured by allowing variation of the model parameters that define the constraints of the LP problem. The algorithms presented in this work explicitly make use of the mathematical properties of LP problems in order to propagate the uncertainty onto the variables involved in the EMPC strategy. Although the problem is an LP for each particular set of model parameters, in the presence of uncertainty the problem becomes nonlinear since it is described by a family of LP problems each corresponding to a particular set of model parameters' values.

With this in mind, an assumption is introduced here for the current study: the output/solution space of a nonlinear system can be divided into a set of polyhedrons each corresponding to a particular active set of constraints, i.e. a region corresponding to a Simplex tableau or part of it. Based on this partitioning approach, the main objectives that have been accomplished in the current study are outlined as follows:

- Provide a new uncertainty propagation algorithm that can quantify uncertainty into the solutions of an LP problem. This algorithm aims to divide and bound the output/solution space of a nonlinear system. Accordingly, it is able to generate the entire uncertainty regions which can be composed by a series of polyhedrons, the solutions in any one of these polyhedrons can be calculated by using simple linear operation.
- Compare the proposed uncertainty propagation algorithm with other conventional approaches, such as the partially uncertainty region generating algorithm based on the 100 Percent Rule where the latter has been proposed in the classical LP literature to calculate regions of existence of a particular Simplex tableau.
- Develop an uncertainty propagation algorithm based on a tree structure where each branch of the tree corresponds to one of the polyhedrons calculated by partitioning the space as explained above. Although a tree structure has been proposed before for MPC in (Lucia, Finkler, & Engell, 2013) the approach proposed in this work is more efficient as compared to the reported work since it exploits the LP structure of the model.
- Formulate a novel robust economic MPC (EMPC) controller for a biochemical process that uses steady state and dynamic metabolic flux model under uncertainty. The proposed robust EMPC makes use of the new uncertainty propagation algorithm as well as the tree structure.

1.2 Overview of the Thesis

Overall this thesis is organized in 5 chapters as follows:

- Chapter 2 discusses the background and literature review relevant to the proposed research objectives. The tree (scenario) based structure of NMPC, simplex algorithm for Linear Programming (LP), control and optimization of bioreactors and the dynamic Metabolic Flux Analysis (MFA) modeling are the fundamental concepts of the methodology that are reviewed in this section and that are relevant for this thesis.
- Chapter 3 presents two novel algorithms referred to as the 100 Percent Rule Based Method and the Convex Cone Method (CCM) for propagating the uncertainty in model parameters onto the solutions of an LP problem. This chapter shows that the 100 Percent Rule Based

Method can provide a necessary but not sufficient polyhedron region based bounds around a nominal point but it cannot cover the entire uncertainty region and thus is computationally inefficient. To address this problem, the novel CCM algorithm is proposed that it is able to account in an efficient fashion for the entire region of uncertainty. A series of lemmas and theorems are presented that served as the basis of the proposed CCM algorithm. Some simple case studies and examples are also discussed in this chapter to illustrate the algorithm in a graphical form.

- Chapter 4 presents a novel robust EMPC algorithm which is computationally efficient with respect to the number of uncertain parameters as compared to other methods. This algorithm is accomplished based on an algorithm, referred as CCM, for partitioning the parameter space of the family of LP problems describing the uncertain model into a series of polyhedrons where for the supremum for each one of them can be obtained by linear calculations. Based on the polyhedrons calculated by the CCM algorithm, a tree structure multistage uncertainty propagation method is proposed and is referred as Tableaus Based Tree (TBT) method. Each branch of the tree corresponds to a polyhedron identified by the CCM method. Based on a theoretical comparison this proposed method is found to be more computationally efficient than Monte Carlo or PCE method. Finally, the robust EMPC algorithm is applied in a case study where the final biomass amount in a bioreactor is maximized. The robustness of the proposed controller is assessed based to comparisons to a non-robust algorithm.
- Chapter 5 summarizes the key conclusions and contributions of this thesis; recommendations and research plan for future work are also discussed in this section.

Chapter 2

Background and Literature Review

In this section, the background and literature review on concepts relevant to this thesis are presented. Nonlinear Model Predictive Control (NMPC) and Economic Model Predictive Control (EMPC) that are the fundamental theories of this study are introduced in the first sections of this chapter. Then robustness of NMPC algorithms, a key focus of the current work, is reviewed. This is followed by a discussion on contributions related to the tree (scenario)-based structure on Robust NMPC and the sensitivity analysis of the Right Hand Side (RHS) of the constraints in LP problems which is often used for studying parametric sensitivity. Then, a review on control and optimization of bioreactors as well as metabolic flux models are presented.

2.1 Model Predictive Control

Model Predictive Control (MPC) is one of the most commonly used model-based control technologies where the internal model may be linear or nonlinear. When the process to be controlled exhibits highly nonlinear dynamic behavior, a nonlinear model is required for accurate prediction. The resulting predictive control strategy that is based on the nonlinear model is referred to as Nonlinear Model Predictive Control (NMPC). Since models are never perfect, it is essential to design a control strategy that is robust to model plant mismatch, where such mismatch is often referred to as model uncertainty. Although there are multiple techniques that can be used to analyze robustness of controllers that are based on linear models, e.g. SSV (Structured Singular Value or μ) and LMI's (Linear Matrix Inequalities), the design of nonlinear robust controllers is still challenging, particularly for NMPC; thus, this is a currently an active area of research (Allgöwer, Findeisen, & Nagy, 2004; Lalo Magni & Scattolini, 2010). Since chemical processes generally exhibit highly nonlinear behavior, most of the robust control techniques that have been developed for linear systems may not be applicable to NMPC or they may lead to highly conservative control actions. Accordingly, robust nonlinear predictive control algorithms are a very attractive alternative to control chemical systems that will result in improved performance as compared to robust linear strategies. Therefore, accounting for robustness to model errors has been identified as one of the key challenges in the research of NMPC controllers (Allgöwer et al., 2004).

Model Predictive Control usually refers to a series of control algorithms in which a performance criterion is optimized along a prediction horizon subject to a set of input/output constraints. In general, a dynamic nominal process model is used to predict the output along the prediction horizon, and some pre-defined norm of the errors between the controlled variables and pre-defined references' trajectories is optimized to ensure a desired closed loop performance. In most cases, a pre-defined number of future control moves are used as the decision variables for the optimization problem, and this number is referred to as the control horizon; whereas the number of future predictions considered in the objective function is referred to as the prediction horizon.

The optimization problem is solved at every time interval with respect to the decision variables (control moves) but only the first control move is actually implemented into the plant while the rest are ignored (Findeisen et al., 2003; Michael A Henson, 1998). By repeating the optimization procedure at every time interval, the controller can compensate for unmeasured disturbances that may enter the process as well as discrepancies between the model and the process outputs that lead to inaccurate predictions. In a linear MPC, the nominal model used for prediction is linear with respect to the manipulated variables and the controlled variables; similarly, process constrains, e.g. valve saturation limits, are also expressed as linear functions. The cost function in linear MPC is commonly formulated as a quadratic function thus resulting in a QP (quadratic programming) for which the global optimal solution can be found in a finite number of iterations using off-the-shelf standard algorithms. On the other hand, in NMPC the nominal model as well as the process constraints can be both linear and nonlinear. In most cases, the MPC schemes which involve nonlinear input/output constraints or with a non-quadratic cost function must be resolved with nonlinear optimization algorithms that can only guarantee local optima.

As it is illustrated in Fig. 2.1, the main steps of an NMPC algorithm are as follows:

1. Using a nonlinear model of the form,

$$\begin{aligned} &\hat{x}(i+1) = f(\hat{x}(i), u(i), d) \\ &\hat{y} = g\big(\hat{x}(i), u(i)\big) \\ &\hat{x}(i=0) = x(k) \\ &i = \{0,1,2,\dots,p\}, k \in \mathbb{N} \end{aligned} \tag{2.1}$$

   where x is plant states, $\hat{x}$ is prediction of plant states, $u = \{u(k), u(k+1), \dots, u(k+m)\}$ are the adjustable variables and they are equal to the manipulated variables or control actions, $x(k=0)$ is measured plant output, $\hat{y}$ is prediction of plant output, $\boldsymbol{d}$ is vector of uncertainty, which is set to its normal condition if no disturbances are considered, f and g are nonlinear vector functions of plant inputs and outputs which starting from a current output measurement $y = g(x(k=0))$ in order to correct for unmeasured disturbances or model errors, future predictions of the outputs are generated over a prediction horizon, p as a function of a sequence of inputs defined over a control horizon m, where $m \leq p$.

2. The optimal control actions are calculated to minimize the cost function with respect to the decision variables, i.e. the manipulated variables. In most cases, this cost function is assumed as a weighted sum of the errors in future predictions with respect to a reference trajectory (or set-point profile) and plant inputs, where the latter are included in the objective function to avoid excessive control actions. Therefore, the typical cost used in the MPC algorithm is as follows:

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

where $\boldsymbol{Q}$, $\boldsymbol{R}$ are weights in the MPC formulation and $\boldsymbol{y}^{sp}$ is the set-point profile.

3. The solution of the optimization problem shown in (2.1) consists of m control actions and as previously discussed, only the first control action, i.e. $u(k+1|k)$ is implemented in the plant.

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

The algorithm described above generally requires supplementary stability constrains since it cannot guarantee process closed-loop stability during operation. Eliminating the error between the prediction and set-points at the end of an infinite control horizon is one method proposed in the literature to ensure stability. This method, referred to as a “terminal constraint condition”, needs high computation effort (Chen & Allgöwer, 1998; Findeisen et al., 2003); also, it may be infeasible in the presence of input constrains or may result in overly conservative control actions. This terminal constraint condition can be mathematically implemented into the optimization problem based on two different options:

i - An exact terminal equality constraint is added in the optimization problem.

ii- A terminal inequality constraint as well as a terminal cost are added to the problem. In the second option the terminal constraint is relaxed by requiring the output terminal prediction to remain within a range of a certain pre-specified error. Thus, using this second approach, the optimization problem is re-formulated.

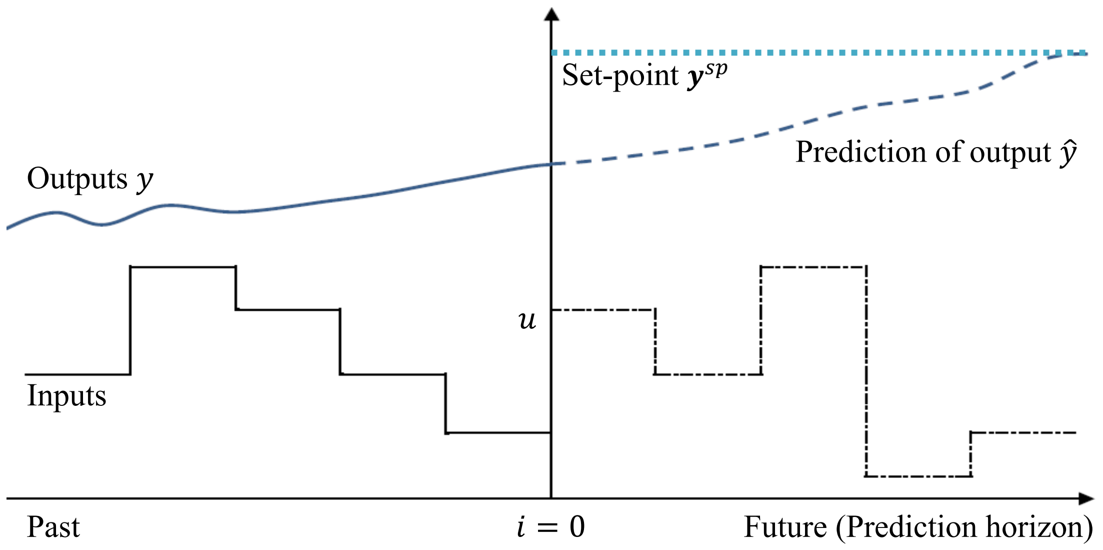

Set-point $\boldsymbol{y}^{sp}$
Prediction of output $\hat{y}$
Outputs y
u
Inputs
Past
$i = 0$
Future (Prediction horizon)

Figure 2.1 One step of a closed-loop MPC trajectory

The design of the terminal region and terminal penalty , have been reported elsewhere (Chen & Allgöwer, 1998; Michalska & Mayne, 1993). The output is forced to reach the region $\mathcal{E}$ by the terminal penalty weight $\boldsymbol{E}$ and the plant output would ultimately converge to the reference trajectory if $\mathcal{E}$ is selected appropriately.

2.2 Economic Model Predictive Control

When an MPC algorithm optimizes control actions based on linear or nonlinear formulations to satisfy economic cost functions instead of (or in addition to) traditional set-point tracking objectives, it is referred to as Economic Model Predictive Control (EMPC). As explained in Chapter 1, the hierarchical control structure consisting of an RTO level sending set-points to lower control levels is generally sub-optimal during transients since the RTO is based on steady state information. Thus, the hierarchical strategy is only optimal when the system is operating in the neighborhood of a particular steady state. However, since the process conditions might be far away from steady state due to dynamic disturbances or transitions between operating points, the RTO calculations for optimal solution is often found to be inaccurate. Figure 2.2 illustrates the situation in which the line projection of all steady state solutions is not optimal (Rawlings, Angeli, & Bates, 2012). In this figure, the profit function is illustrated in terms of a state as well as an input assuming for simplicity that the system involves only one-state and one-input. The data in the figure is expressed in deviation terms with respect to the economic global optimum. The blue plane indicates steady state conditions. It is obvious from this figure that the best steady state condition (maximum in the red surface), which is defined as the most profitable solution, is not the optimal point in terms of the RTO calculated set-points. Moreover, due to the highly nonlinear nature of chemical process, there may exist dynamic operating regimes which result in higher profitability than that obtained from steady state, i.e. periodic regimes (David Angeli, Amrit, & Rawlings, 2012). For this reason, to operate the system at the economic global optimum, the control actions must be calculated based on the system's dynamic behavior as it is performed in EMPC.

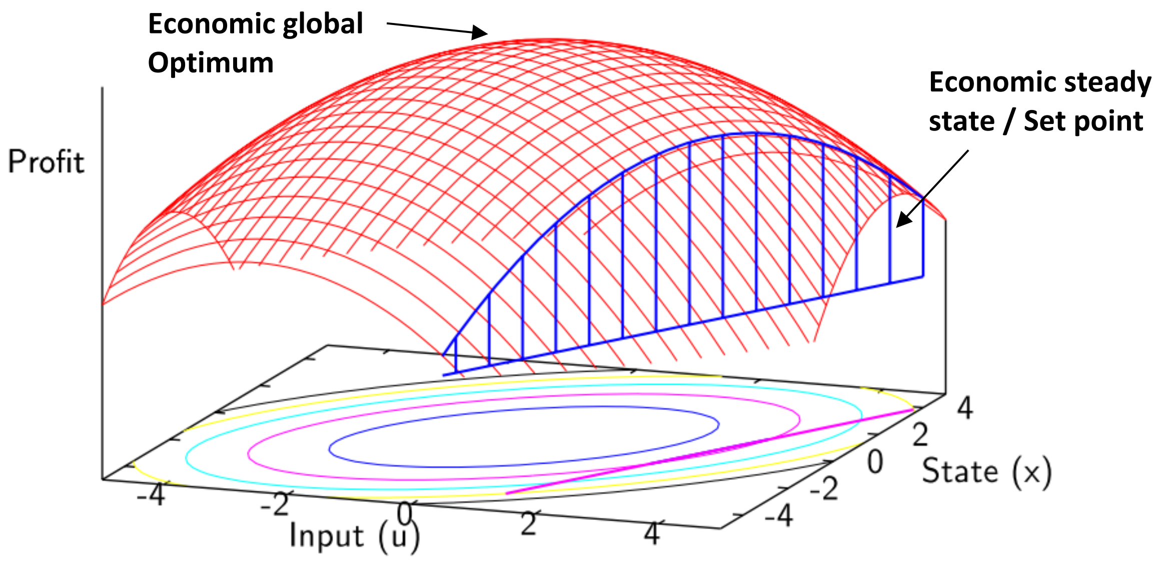


Figure 2.2 Graphic showing the most profitable solution is away from the calculated set-points (blue plane represent steady state solutions) (Rawlings et al., 2012)

In addition to providing a convenient method for computing the economic optimum, EMPC generally maintains a number of other benefits when compared to the standard MPC. For instance, direct handling of dynamic (path) constraints, the use of dynamic MIMO models for prediction (J.

Ma, Qin, Salsbury, & Xu, 2011), online computation of control actions, etc. Furthermore, since EMPC uses an economic performance index instead of a quadratic stage cost, it does not rely on any particular set point value calculation thus eliminating the need for the RTO level (Rawlings et al., 2012). The risk of bypassing the RTO level is that the stability or robustness of the controller has to be ensured. These aspects are challenging since generally nonconvex stage costs, which is generally defined as $f\big(x(i), u(i)\big)$ in (2.1), with multiple minima instead of quadratic cost as in NMPC, must be considered. The use of nonconvex stage cost might result in convergence to a local optimum instead of a global optimum point.

The classical discrete-time version of the EMPC with finite-time prediction horizon can be formulated as an online optimization problem as follows (Rawlings et al., 2012):

$$\min_{u} \sum_{i=0}^{p-1} l_e(\hat{x}(i), u(i), d = 0) + V_f(\hat{x}(k + p)) \tag{2.3}$$

s. t.

$$\hat{x}(i + 1) = f_d(\hat{x}(i), u(i), 0) \tag{2.4}$$

$$\hat{x}(i = 0) = x(k) \tag{2.5}$$

$$f\big(\hat{x}(i), u(i)\big) = 0 \;\; i = \{0,1,2, \dots, p\}, k \in \mathbb{N} \tag{2.6}$$

$$\hat{x}(k + p) \in \mathbb{X}_f \tag{2.7}$$

As per the general approach in dynamic optimization problems, the control actions are given by a series of piecewise constant inputs along the trajectory of the prediction horizon, i.e. in each time interval $[i, i + 1)$, $\hat{x}$ refers to the predicted state trajectory sequence over the prediction horizon, and p is the number of sampling intervals that form the prediction horizon.

The cost function (2.3) contains the economic stage cost with a terminal cost/penalty $V_f: \mathbb{X}_f \to \mathbb{R}$. The nominal dynamic model (2.4) is used to predict the future evolution of the system and it is initialized with each state's corresponding measurement (2.5). Equations of (2.6) represents the process or system constrains. Furthermore, the constraint (2.7) is a terminal constraint, which ensures that at the end of horizon the predicted $\hat{x}$ will be within a neighborhood of a terminal value defined by the terminal set $\mathbb{X}_f$.

Angeli et al.(2009) reported a receding horizon control algorithm to control a nonlinear plant with stage cost that is not necessarily convex (D Angeli, Amrit, -, 2009, & 2009, 2009). Those authors illustrated that, when the optimal steady state is used as a terminal constraint with feasible initial conditions, the average economic performance of their algorithm will at least achieve a same level performance as the optimal steady state. Nevertheless, no further explicit assumption was made in that study on the issue of closed-loop stability. After that, that group developed relatively simple stability conditions for EMPC based on Lyapunov arguments (Diehl, Amrit, On, & 2011, 2011). Based on a predefined Lyapunov function, they proposed a formulation of the EMPC algorithm for a continuous system $f(x,u)$, with continuous stage cost $l(x,u)$ and terminal point-wise constraints to ensure nominal stability (David Angeli et al., 2012). It should be emphasized that robustness to model error was not addressed in that study.

2.3 Robust Nonlinear Model Predictive Control (NMPC)

Robustness to model error remains as an essential challenge for the design of NMPC methods (Findeisen et al., 2003; Lalo Magni & Scattolini, 2010). Generally, there are several reasons that contribute to the model-plant mismatch, such as disturbances arising from changes in operating conditions, uncertainty derived from simplifications of the model structure or model reduction, lack of knowledge of key parameters, inaccurate understanding of physical mechanisms of the process, etc. Therefore, robustness should be a main emphasis in the design of NMPC algorithms to ensure its adoption in industrial applications.

A key consideration when addressing robustness to model error is that the robust controller would not become excessively conservative. The degree of conservatism will generally depend on how the model error is quantified with respect to the nominal model and on how this error is propagated onto the objective function and the constraints of the optimization problem. A nominal model which maintains a desirable accurate prediction over the prediction horizon would require a small level of uncertainty to explain the overall process behavior. Since uncertainty may also destabilize a control system, robust performance must be considered during the design stage. Following these considerations, some techniques that can be used to analyze the stability and performance of a robust controller are reviewed in the following sections.

2.3.1 LMI's for Robust Control

Linear Matrix Inequalities based analysis (LMI's) consists in formulating a series of algebraic matrix inequalities that serve to test the stability and performance properties of a closed loop system with respect to a polytopic representation of model uncertainty (Vanantwerp & Braatz, 2000). Three main problems are generally addressed by LMI's in the control field, the Feasibility problem, Generalized Eigenvalue problem, and Linear programming problems. This approach has gained considerable attention in the control field, since it can be used to analyze both robust stability and robust performance and can be also used for the formulations of on-line robust MPC control algorithms that involve input and output constrains (Kothare, Balakrishnan, & Morari, 1996). A formal theoretical method for synthesis of robust MPC with infinite horizon and different forms of uncertainty was proposed in the literature (Kothare et al., 1996). That approach was based on an infinite norm based objective function (S Boyd, Ghaoui, Feron, & Balakrishnan, 1994) and it was extended to include inputs as well as outputs constrains. This LMI formulation was then used for the design of Robust Distributed MPC for polytopic uncertainty with both time-varying and time-invariant models (Al-Gherwi, Budman, & Elkamel, 2011). Distributed MPC refers to the case where MPC controllers are applied to subsets of input and output variables in a chemical plant and communication is exchanged among these multivariable controllers. The sporadic loss of communication has also been addressed in Al-Gherwi, 2011 by using a robust estimator based on LMI's. The distributed MPC approach has been proposed as an option to decrease the complexity of computation in the algorithms in a real plant implementation (Kumar & Budman, 2017). Though LMI's have been generally used for linear robust control, it could also be used for analyzing robustness when a nonlinear process that can be approximated by uncertainty polytopes with respect to a nominal linear model (Santander et al., 2016). However, investigating polytopic uncertainty to describe the actual nonlinear process is challenging and may result in conservative uncertainty descriptions that result in conservative controller designs (Doyle, Packard, & Morari, 1989).

2.3.2 SSV for Robust Control

Another method for investigating stability and performance of controllers is Structured Singular Value (SSV) analysis, which also referred to as μ-analysis. This mathematical tool is generally used for analyzing controller based on uncertain models with either structured or unstructured uncertainties. The main idea of μ norms is to extract the uncertain part of the nominal model and then generate a Linear Fractional Transformation (LFT) relation between the inputs and outputs that can be schematically described by an interconnection matrix. Given a structure of the resulting interconnection matrix (M) and the uncertainty description Δ, the μ norm provides a measure of the smallest perturbation within the given uncertainty set, which can cause destabilization or violation of a pre-defined performance bound. The ability to quantify performance by an SSV based norm has been proposed for calculating worst-case deviations of the output with respect to the set-point along the prediction horizon based on a predictive control strategy (Kumar & Budman, 2017).

Some studies have used SSV in the context of robust NMPC algorithms in order to develop a competitive method which require lower computational effort. For example, several studies have used an analytical approach where the parametric uncertainty is propagated by using Taylor series and then an optimization problem is formulated in terms of the Structured Singular Values (SSV) of the decision variables (D.L. Ma & Braatz, 2001; David L. Ma, Chung, & Braatz, 1999; Nagy & Braatz, 2003). However, since the calculation of SSV-norm is a NP-hard type problem, the computational demand for these methods grows exponentially when the dimensions of the process with respect of inputs and outputs are large.

2.3.3 Main Algorithms on Robust NMPC

Robust NMPC is generally based on a max-min problem involving the minimization of a control cost calculated for the maximal uncertainty (Findeisen et al., 2003; Lalo Magni & Scattolini, 2010). Thus, this is paramount to the optimization of the worst case within the uncertainty set.

To address robustness, Mayne et al., 2011 proposed a tube-based method where NMPC calculated the control actions based on a nominal model and an additional controller was used to drive the outputs towards the nominal time trajectory calculated with the nominal model. In this way, the tube-based controller ensure that the outputs' trajectories are bounded within a tube (envelope) around the nominal output trajectories (L. Magni, Raimondo, & Scattolini, 2006; Mayne, Kerrigan, van Wyk, & Falugi, 2011). At each time interval the tubes around the nominal trajectories are determined by computing certain state constraints which are used to ensure Lyapunov stability with respect to bounded disturbances. A modification of this approach is proposed by Cannon et al., 2011, where the nonlinear model is converted into successive linear functions for every prediction into the time horizon. In most cases, these tube-based algorithms were designed based on mechanistic models and estimates of the potential disturbances that may enter the process. For this reason, it is difficult to apply this method for the case of uncertainty in parameters rather than uncertainty that captures disturbances. Therefore, when designing a robust NMPC controller with significant parametric uncertainty, the tube-based controller has not been used.

A key disadvantage of the min-max algorithms mentioned above is the high computational effort required to solve the optimization problem on-line. To circumvent this issue, some approximations have been proposed. For instance, using a bounded set description of the parametric uncertainties, Diehl et al., 2008 developed an algorithm that satisfies the necessary first order optimality conditions instead of the inner maximization problem, and assumed that the worst-case occurs at the boundary of the uncertainty set. Zavala and Biegler, 2009 performed a preliminary nominal model based estimation of the controller actions at given time steps thus reducing considerably the online calculations necessary in their approach. However, both two approaches need to calculate the derivatives of the objective function, constraints and disturbance uncertainty set which may be time consuming and may be very complex for nonlinear mechanistic models of high dimensions. Thus, there is still a good motivation to reduce the online computations of robust NMPC algorithms, which is the focus of the current research.

The general area of optimization with uncertainty is vast. Several different methods in terms of the optimization algorithms with uncertainty have been proposed. Some studies, i.e. Kawohl et al., 2007 provided a simulation based method where the Monte Carlo simulations was used to obtain the worst-case of the weighted contribution of the first two statistical momenta (Kawohl et al., 2007). Monte Carlo is a method where the uncertainty is propagated onto the outputs of interest for a large number of samples selected from a *priori* known uncertainty distribution. However, the high computation costs associated with this approach made it difficult to be applied for on-line implementations of MPC robust controllers (Birge & Louveaux, 2011; Niederreiter, 1978).

Alternatively, several competitive methods which require lower computational effort than the Monte Carlo approach have been proposed. For example, some studies have used an analytical approach where the parametric uncertainty is propagated by using Taylor series and then an optimization problem is formulated in terms of the Structured Singular Values (SSV) of the decision variables (D.L. Ma & Braatz, 2001; David L. Ma et al., 1999; Nagy & Braatz, 2003). This latter method was found valid when first order estimates are not sufficiently accurate to capture the uncertainty and second or higher order estimates are necessary. Using that approach, Diaz-Mendoza and Budman, 2010 proposed an RNMPC method based on SSV norms for continuous processes (Díaz-Mendoza & Budman, 2010). The cost function in that method is formulated as a function of SSV norm where the latter provides a bound on the worst possible output deviation with respect to the set-point in the presence of model errors. However, the computational demand for this method grows exponentially when the dimensions of the process with respect of inputs and outputs are large since the calculation of SSV-norm is a NP-hard type problem. Moreover, the performance of this algorithm might be conservative as it is based on the worst case of the output' deviations with respect to the set-point.

In an effort to address these drawbacks, an alternative method, which is based on the propagation of parametric uncertainty using Polynomial Chaos Expansions (PCE), was proposed. PCE refers to a random process where orthogonal basis functions are used to generate a spectral expansion as a function of random variables (Ghanem & Spanos, 1990). Hover and Triantafyllou, 2006 proposed a PCE based method to analyze the stability of an explicit nonlinear system with random initial conditions or random parameters and showed that the approach significantly reduces the computational load as compared to Monte Carlo approaches(Hover & Triantafyllou, 2006). Smith, Monti and Ponci, 2009 provided an LQG controller design for linear systems that was based on PCE approximations of the parametric uncertainty (Smith, Monti, & Ponci, 2009). Recently, Kumar and Budman, 2017 have proposed a new PCE-based method for nonlinear systems where the parametric uncertainty was propagated onto the cost and constraints of a robust optimization of a batch process. Both on-line and off-line robust optimization problems were addressed. In the online problem Kumar and Budman, 2017 also demonstrated the role of the feedback in reducing the conservatism of the controller (Kumar & Budman, 2017). On the other hand, the computational effort of this PCE based approach was shown to be highly sensitive to the number of parameters and states and the conservatism of the controller significantly increased with respect to the prediction horizon due to the cumulative effect of uncertainty.

2.3.4 Tree (Scenario) Based Structure of Nonlinear Model Predictive Control

To handle some of the limitations of robust NMPC and EMPC strategies such as conservatism and the computational complexity of the methods discussed above, scenario based strategies have been recently proposed (Lucia et al., 2013). In this type of strategy different scenarios, corresponding to different uncertainty realizations, can occur with different degrees of probability. Then, the future predictions of the controlled variables are assumed to follow different possible trajectories thus forming a tree like structure that it is used for formulating a robust optimization problem. This tree-based approach has been shown to reduce the conservatism of the resulting NMPC scheme as compared to previous robust approaches and it is valid a for variety of controller strategies, such as Multi-stage nonlinear model predictive control (Multi-stage NMPC) and economic NMPC (Lindhorst, Lucia, Findeisen, & Waldherr, 2016; Lucia, Andersson, Brandt, Diehl, & Engell, 2014; Lucia et al., 2013).

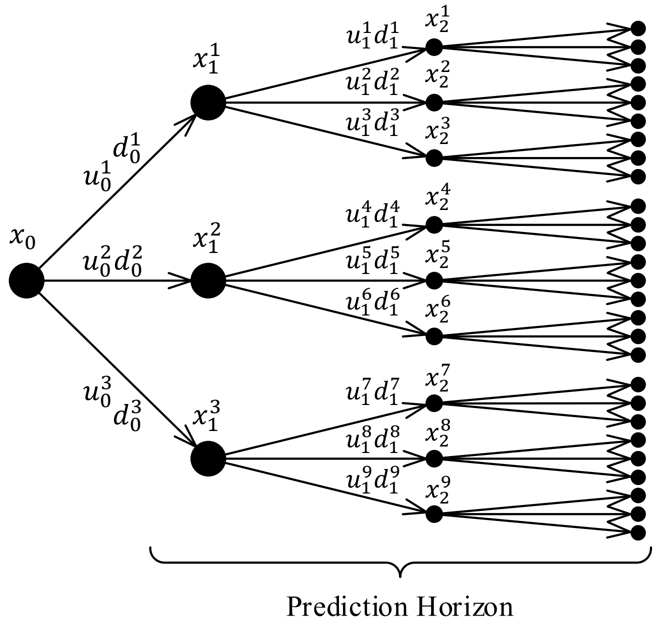


Figure 2.3 Scenario tree of the uncertainty evolution (Lucia et al., 2013)

The main idea of this method is that the time trajectories corresponding to different parameter uncertainty or disturbance realizations can be represented by a tree composed of discrete scenarios as shown in Fig. 2.3. For generality, both parameter uncertainty and disturbance uncertainty are treated in the same fashion. Each node of the branches in this tree structure is generated by uncertainty, i.e. parameter or disturbance related uncertainty. Each uncertainty realization is denoted as $d_k^{r(j)}$. Each one of the paths from the root node x_0 to a leaf node is referred to as a scenario. The scenario tree approach involves discrete time computations of the nonlinear system of equations defining the process as follows:

$$x_{k+1}^{j} = f(x_k^{p(j)}, u_k^j, d_k^{r(j)}) \tag{2.8}$$

where each state x_{k+1}^{j} depends on present state $x_{k}^{p(j)}$, the corresponding control input u_{k}^{j} as well as corresponding realization r of the uncertainty at stage k, $d_{k}^{r(j)}$. For simplicity, the branches from each one of the nodes are defined by the corresponding value of the uncertainty realization by $d_{k}^{r(j)} = \{d_{k}^{1}, d_{k}^{2}, \dots, d_{k}^{s}\}$ at stage stage k for s different possible values of the uncertainty.

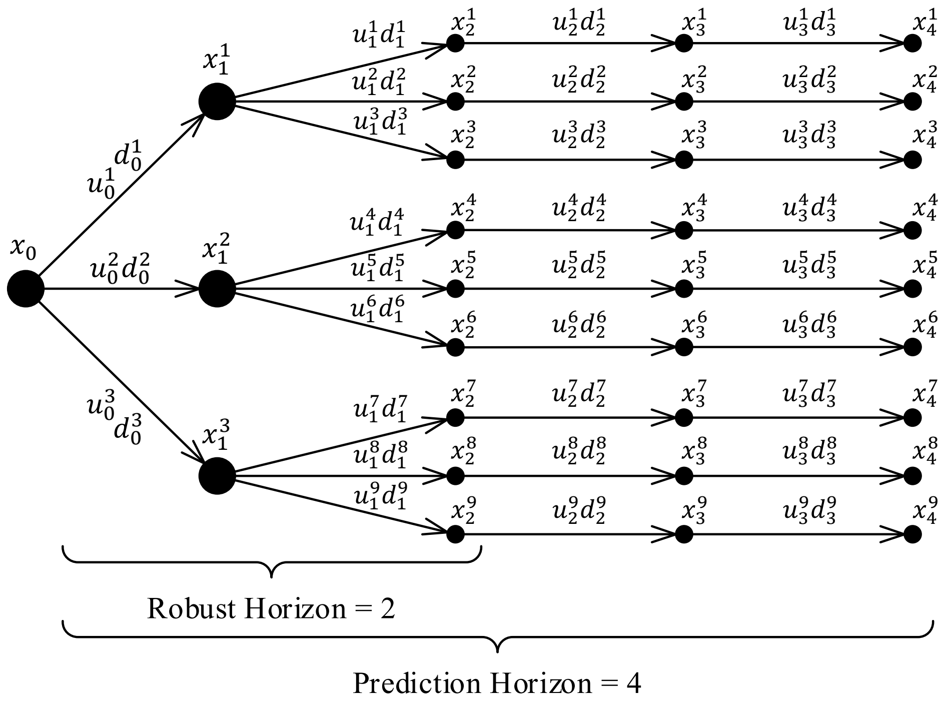

x_0
u_0^1 d_0^1
$u_0^2 d_0^2$
u_0^3 d_0^3
x_1^1
x_1^2
x_1^3
$u_1^1 d_1^1$ x_2^1 $u_2^1 d_2^1$ x_3^1 $u_3^1 d_3^1$ x_4^1
$u_1^2 d_1^2$ x_2^2 $u_2^2 d_2^2$ x_3^2 $u_3^2 d_3^2$ x_4^2
$u_1^3 d_1^3$ x_2^3 $u_2^3 d_2^3$ x_3^3 $u_3^3 d_3^3$ x_4^3
$u_1^4 d_1^4$ x_2^4 $u_2^4 d_2^4$ x_3^4 $u_3^4 d_3^4$ x_4^4
$u_1^5 d_1^5$ x_2^5 $u_2^5 d_2^5$ x_3^5 $u_3^5 d_3^5$ x_4^5
$u_1^6 d_1^6$ x_2^6 $u_2^6 d_2^6$ x_3^6 $u_3^6 d_3^6$ x_4^6
$u_1^7 d_1^7$ x_2^7 $u_2^7 d_2^7$ x_3^7 $u_3^7 d_3^7$ x_4^7
$u_1^8 d_1^8$ x_2^8 $u_2^8 d_2^8$ x_3^8 $u_3^8 d_3^8$ x_4^8
$u_1^9 d_1^9$ x_2^9 $u_2^9 d_2^9$ x_3^9 $u_3^9 d_3^9$ x_4^9
Robust Horizon = 2
Prediction Horizon = 4

Figure 2.4 The uncertainty evolution with robust horizon represented by scenario tree structure (Lucia et al., 2014)

As the prediction horizon is expanded, there is an exponential growth of the tree structure that will result in increasing computational costs. To avoid this increased computational cost, a modification has been proposed, as shown in Fig. 2.4, whereby branching with respect to different uncertainty realizations is only done for the initial time intervals of the prediction horizon (2 first intervals in Fig. 2.4) and then the uncertainty realization in each trajectory is assumed to remain constant until the end of the horizon (intervals 3 and 4 in Fig. 2.4). The rationale for this modification is that only the first steps of the prediction horizon may describe the actual process behavior whereas the later stages of the prediction are not accurate anyways and they will be significantly changed along the solution due to disturbances entering the process and random measurement noise. The number of intervals for which branching is done has been referred to as *the robust horizon*. This idea have been also used in many other scheduling problems such as a linear max-min MPC (de la Pena, Alamo, Bemporad, & Camacho, 2006).

The most challenging problem of the scenario tree structure based method is how to generate a reasonable tree structure that maintains a balance between an accurate estimation of uncertainty and acceptable size of the robust horizon. To address this issue, there are three main methods that have been discussed in the literature as follows: i) Monte Carlo simulations in combination with sample average approximations (SAA) method (Shapiro, 2003), for which the computation still increases exponentially with the horizon length; ii) a deterministic method, such as moment matching method (Høyland, Kaut, & Wallace, 2003) of a probability distribution or the minimization of a certain probability matrix like the Wasserstein distance (de Oliveira, Sagastizábal, Penna, Maceira, & Damázio, 2010); iii) machine learning techniques for generation of scenario trees (Defourny, 2010).

Several versions of the tree structure based MPC have been reported. Lucia, Finkler and Engell in 2013 initially established the method and applied it a multi-stage NMPC controller of semi-batch polymerization benchmark problem with uncertainty; later on, Lucia et al., 2014 further improved this model by using an economic NMPC controller which optimizes the process over a set of affine control policies. Lindhorst et al., 2016 have used this approach for bioreactor control and optimization. A robust controller framework was generated by Lindhorst et al, 2016 to handle the uncertainty of a dynamic enzyme-cost Flux Balance Analysis (deFBA) describing a bioreactor process. A receding prediction horizon and a simplified deFBA model describing only the short term behavior of the process were used in order deal with the exponential computation cost increase along the prediction horizon. However, in Lindhorst approach the number of scenarios still increases two-fold with each additional uncertainty. Thus, this framework is only acceptable for online implementation with a short prediction horizon or a limited number of uncertainties of the deFBA model.

Considering these limitations, in the current work, a novel tree structure based algorithm is proposed in Chapter 3 of the thesis that exploits the particular nature of the dynamic metabolic flux model describing the system, i.e. linear programming based model, to reduce the computational effort to estimate control actions in the presence of model uncertainty.

2.4 Sensitivity Analysis of RHS

This thesis is dealing with robust optimization of a biotechnological process that is described by a Linear Programming (LP) problem. This model will be formally introduced in Chapter 3. Sensitivity analysis of the Right Hand Side (RHS) of the inequalities in the LP optimization problem is an essential theoretical element of the strategy proposed in this work and it is thus reviewed here for completeness. A standard LP problem can be formulated as follows (Hillier, 2001):

$$max\ Z = \boldsymbol{cx} \tag{2.9}$$

s. t.

$$\boldsymbol{Ax} \leq \boldsymbol{b} \tag{2.10}$$

$$\boldsymbol{x} \geq 0$$

where Z is an overall measure of performance, $\boldsymbol{x}$ is the vector of x_j, x_j is decision variables (solution) with the amount of n $(j = 1, 2, \dots, n)$, $\boldsymbol{c}$ is the row vector of c_j, $\boldsymbol{A}$ is the matrix of a_{ij}, $\boldsymbol{b}$ is the vector of b_i, a_{ij} b_i and c_j (for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$) are the input constants which are generally referred as the parameters of the model. The matrix $\boldsymbol{A}$ in (2.10) is often referred to as the *Left Hand Side* of the constraints equations and it will be referred to as LHS for short; $\boldsymbol{b}$ is a vector of independent coefficients as will be referred to as the *Right Hand Side* (RHS).

2.4.1 Simplex Algorithm for Linear Programming

The Simplex algorithm (or Simplex method) is the most popular algorithm method for solving LPs. The solution of LPs is iterative in nature and it involves the use of tableaux that are particular matrix representations of either the original problem formulation or transformed formulations describing each one of the iterations of the optimization search. The standard initial tableau of the simplex method (the original set of equations) is as follows (Hillier, 2001):

$$\begin{bmatrix} 1 & -\boldsymbol{c} & 0 \\ 0 & \boldsymbol{A} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} Z \\ \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{b} \end{bmatrix} \tag{2.11}$$

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} \geq 0 \tag{2.12}$$

$$\boldsymbol{x_s} = \begin{bmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \end{bmatrix} \tag{2.13}$$

where $\boldsymbol{I}$ is $m \times m$ identity matrix, $\boldsymbol{x_s}$ is column vector of slack variables that is needed to obtain the augmented form of the problem as it is shown in (2.13). For the given matrix (2.11), the basic solutions of this tableau are the solutions of the m equations in (2.14), i.e.

$$[\boldsymbol{A} \quad \boldsymbol{I}]\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} = \boldsymbol{b} \tag{2.14}$$

Since the system of algebraic equations given by (2.14) is underdetermined n variables referred to as *non-basic* can be eliminated out of a total of $n + m$ variables of the vector $\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix}$ by equating these variables to zero. This transforms (2.14) into a set of m equations with m unknowns which are referred to as the *basic* variables, i.e.

$$\boldsymbol{Bx_B} = \boldsymbol{b} \tag{2.15}$$

where the vector of basic variables is as follows:

$$\boldsymbol{x_B} = \begin{bmatrix} x_{B1} \\ x_{B2} \\ \vdots \\ x_{Bm} \end{bmatrix}$$

and the basis matrix is as follows:

$$\boldsymbol{B} = \begin{bmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{m1} & \cdots & B_{mm} \end{bmatrix}$$

Matrix $\boldsymbol{B}$ is generated by eliminating the columns corresponding to coefficients of non-basic variables from $[\boldsymbol{A} \quad \boldsymbol{I}]$. The simplex method searches for a solution through a series of iterations where each involves a Gaussian elimination procedure. The rearrangement of the resulting tableau for the basic and non-basic variables identified at each iteration is referred to as pivoting. During the iterations of the simplex method, the order of $\boldsymbol{x_B}$ as well as the column order of $\boldsymbol{B}$ may be different from one iteration to the next. The detailed description of the pivoting process is shown elsewhere (Hillier, 2001). When the pivoting process of the Simplex Algorithm meets a user defined stopping criteria, the inverse matrix $\boldsymbol{B}^{-1}$ of $\boldsymbol{B}$ is obtained. Since $\boldsymbol{x_B}$ is not zero by definition (basic variables) then $\boldsymbol{B}$ is nonsingular and $\boldsymbol{B}^{-1}$ always will exist. This inverse of $\boldsymbol{B}$ is used to solve for the basic variables from (2.15) as follows:

$$\boldsymbol{x_B} = \boldsymbol{B}^{-1}\boldsymbol{b} \tag{2.16}$$

Let $\boldsymbol{c_B}$ be a vector of the objective function coefficients, including zeros for slack variables, for the corresponding elements of $\boldsymbol{x_B}$. The optimal solution Z of the objective function for the basic solution can be calculated as follows:

$$Z = \boldsymbol{c_B}\boldsymbol{x_B} = \boldsymbol{c_B}\boldsymbol{B}^{-1}\boldsymbol{b} \tag{2.17}$$

2.4.2 100 Percent Rule for Linear Programming

Sensitivity analysis of the RHS is conducted to assess the effect of changes in the elements of the vector $\boldsymbol{b}$ (RHS of LP), on the basic solutions $\boldsymbol{x_B}$ as well as the cost function Z. Since the uncertainty in most problems is captured through changes in b_i the sensitivity of the RHS is of particular interest in robust LP solutions. In most cases, the sensitivity analysis of the RHS is used to find an allowable feasibility range with respect to a nominal optimal basic feasible (BF) solution (Hillier, 2001). Within the feasible range, the structure of the tableau remains constants for all possible changes in $\boldsymbol{b}$. Thus, within the feasible range of the tableau, variables that are basic for one combination of b_i's are basic for another combination of b_i's, and $\boldsymbol{B}^{-1}$ is constant. Using this fact, the effects of changes in b_i on $\boldsymbol{x_B}$ and Z can be easily calculated by using (2.16) and (2.17), respectively. The rate at which Z may be increased by (slightly) changes in the $\boldsymbol{b}$ are referred to as a kind of *shadow prices*. Thus, the shadow prices, which is the partial derivative of Z with respect of $\boldsymbol{b}$ in some extent, inside a feasible range of one tableau is also constant.

It is straightforward and fast to calculate the allowable range of feasibility for changes in only one of the b_i. From the formula (2.16), the adjusted values in $\boldsymbol{B}^{-1}$ for the basic variables can be obtained. Consequently, the solution of the allowable feasibility range consists on calculating the range of values of b_i such that $\boldsymbol{x_B} \geq \boldsymbol{0}$. However, analyzing the effect of simultaneous changes in RHS is more involved. An approximation of the allowable feasibility region can be obtained by using the *100 percent rule* (Hillier, 2001). The idea of this method is that the shadow prices remain feasible for predicting the effect of simultaneously changing the RHS as long as the changes of b_i's are small enough so as to satisfy the 100% rule. The sum of the percentage changes of all b_i is used to check if the changes are small enough. When the summation of changes in $\boldsymbol{b}$ does not exceed 100 percent, the shadow prices will surely be allowed. However, this method only provides a necessary condition but not sufficient since it cannot provide an explicit judgement if the sum does exceed the 100 percent limit.

The100 percent rule has been often used to perform sensitivity analysis with respect to changes in the RHS that emerge due to model uncertainty. However, the application of this method for robust NMPC has been found limited and computationally inefficient in the current work as will be shown in Chapter 4. For instance, this algorithm is found to provide small regions of feasibility in the neighborhood of current solutions but cannot provide an accurate description about the entire feasibility range. This is especially problematic when, for a given uncertainty set, many ranges of feasibility, i.e. many tableaus, are possible. This requires an exhaustive search of allowable feasibility regions around different solutions which is computationally prohibitive for problems of high dimensions, e.g. problems involving several energies, momentum and mass conservation balance equations. Further discussion about these limitations are presented in the following chapter. In the present work, an alternative computationally efficient approach is proposed that is particularly targeted to seek for the feasibility regions contained within pre-defined ranges of changes of the elements b_i's in the RHS of the LP.

2.5 Bioreactor Control and Optimization

The industrial application to be considered in this work is the design of a robust NMPC controller for bioreactors. Modelling of biological systems is challenging in some extent, since these systems generally exhibit highly nonlinear behavior and the models have large dimensions due to the interconnected nature of the metabolic network of reactions that describe microorganism behavior. In addition to model complexity, bioreactors are generally operated in four different operating modes, batch, fed-batch, perfusion and continuous which adds further complexity to the controller design. Currently, the most popular mode of operation in the pharmaceutical industry is fed-batch mode where the substrate is gradually fed into the reactor and the product is only drawn at the end of the process. A key advantage of this mode is the avoidance of high initial concentrations of nutrients that may inhibit growth. For instance, high initial glucose concentration inhibits yeast growth but in fed-batch mode the gradual supplementation of substrate into the reactor can result in good growth while the glucose concentration is maintained at low (acceptable) levels. Another advantage of batch and fed-batch operations is that it is easier to maintain sterilized conditions in the bioreactor since the products are only drawn out after the end of the process (Yamuna Rani & Ramachandra Rao, 1999) as compared to continuous or perfusion operations were products or media are continuously withdrawn. Following these considerations, the review below is focused only on fed-batch reactor operation and control since this is the operating mode that will be considered in this research study.

Classical fed-batch reactors have been controlled based on simple PID controllers that are aimed at maintaining key operating parameters, e.g. PH, temperature or concentration of some inorganic composition, at their set-points by making changes in the substrate feed rate. The limitations of this approach is that closed loop bioreactor performance is impacted by the nonlinear dynamic behavior of the bioprocess as well as the difficulty to acquire online measurement of important metabolites (Lübbert & Jørgensen, 2001; Yamuna Rani & Ramachandra Rao, 1999). However, recent advances has been done in bioprocess sensor technology, such as stringent FDA guidelines for online bioprocess measurements (M A Henson, 2010), may improve the performance of model based controllers in the future.

Due to the lack of reliable sensors, many past studies have proposed offline dynamic optimization of the bioprocess or online open-loop economic controllers that aim to optimize the final product concentration by determining a suitable substrate feeding profile. Accordingly, these past open loop studies have often produced conservative results since they did not exploit the feedback error to correct for unmeasured disturbances. More recently a periodic online re-calculation method of feed-profiles have been introduced based on infrequent measurements (Banga, Alonso, & Singh, 1997).

There are three main reasons for addressing robustness of model based fed-batch bioreactor controllers (Kuhlmann, Bogle, & Chalabi, 1998): i) although the model parameters are assumed to be constant, they are generally time-varying due to metabolic shifts occurring along the process ii) unmodeled dynamics due to incomplete knowledge about the process, and iii) large disturbances occurring in the process. It should be noticed that in current industrial practice offline optimization is often preferred over online feedback strategies for fed-bath due to the lack of sophisticated online-measurement techniques in the current pharmaceutical industries as well as tight limitations imposed by regulatory bodies such as FDA. Hence it is important to investigate the effect of plant model mismatch and disturbances on recipes resulting from robust optimization calculations

(Srinivasan, Bonvin, Visser, & Palanki, 2003). As discussed in section 2.3, in most studies about robust controller optimization techniques the model uncertainty is propagated with the Monte Carlo sampling algorithm though this method requires high computational effort. Recently, PCE's has been used for uncertainty quantification and propagation in robust optimization problems in a bioreactor process modelled by a Dynamic Metabolic Flux Model as used in the current work (Kumar & Budman, 2017). In this work, uncertainty was propagated by representing the uncertain model parameters by PCE expansions where the coefficients of the expansions are calculated form the first principles equations in combination with Galerkin's projection methods. Since the current work consider a similar case study to the one studied by Kumar and Budman study, the latter will be used for comparison purposes with the current approach presented in Chapter 4.

2.6 Metabolic Flux Model

To conduct either off-line robust optimization or robust predictive control of a bioreactor system, it is essential to formulate an appropriate dynamic process model that describes the system in a wide range of operating conditions. This is especially critical in batch or fed-batch operation since in these processes the variables evolve with time over a wide range of conditions in contrast to continuous operations that remain in the neighborhood of a fixed operating point.

Dynamic models of biological systems are generally classified as unstructured and structured models based on the extent of the biological/metabolic detail included in the model. The unstructured models consist of simplistic substrate and biomass balances coupled to each other through the growth rate kinetic expression. However, those models often ignore the complex interactions among many different metabolites existent in the system. A classical dynamic unstructured model with enzyme kinetic that has been used extensively in past bioreactor control and optimization studies is as follows:

$$\frac{dV}{dt} = F, \qquad \frac{dX}{dt} = \mu X, \qquad \frac{dS}{dt} = FS|_{in} - \frac{\mu X}{Y_{X|S}}$$
$$\frac{dP_c}{dt} = \frac{\mu X Y_{P|S}}{Y_{X|S}}, \qquad \mu = \frac{\mu_{max}\, S}{K_m + S} \tag{2.18}$$

where F is feed rate, V is batch volume, X is concentration of biomass, μ is rate of cell growth, S is substrate concentration, P_c is product concentration, μ_{max} is maximum growth rate, K_m is substrate saturation constant, $Y_{P|S}$ and $Y_{X|S}$ are yield coefficients. Since the structured models are based on metabolic reactions that are specific to the organism under study, the unstructured models are simpler than structured ones while the latter can correctly describe the relations between each metabolite participating in the process.

Metabolic flux analysis (MFA) modeling is a method to generate structured models based on flux balance of metabolites at quasi-steady state (Varma & Palsson, 1994), i.e. steady state is either assumed at each time interval or over a certain culture duration. MFA consists of formulating a stoichiometric matrix $\boldsymbol{\mathcal{A}}_{m\times n}$ with respect of the reaction fluxes vector $\boldsymbol{v}_{n\times 1}$ and solving mass balances of extracellular metabolites as per the following equation:

$$\boldsymbol{\mathcal{A}}\boldsymbol{v} = \boldsymbol{\mathcal{b}} \tag{2.19}$$

where $\boldsymbol{\mathcal{b}}_{m\times 1}$ represents a vector of consumption or production rate of extracellular metabolites such as nutrients and by-products, $\boldsymbol{v}$ is the vector of fluxes (mol/h/mol biomass). Since each metabolite generally participates in more than one reaction the resulting system of algebraic equations in (2.20) is generally under-determined and thus additional constraints are needed in order to uniquely define the fluxes. Varma and Palsson, 1994 proposed an assumption where the organisms are continuously trying to maximize growth and allocate resources in order to complete this task. This fundamental assumption is generally justified by the occurrence of naturally evolutionary processes by which the cells have adapted to act as an optimizer of resources. For example, bacteria have evolved mostly to proliferate by optimally distributing their available resources/nutrients. Based on this assumption, and assuming that growth μ is to be maximized, the MFA modelling can be represented as a Linear Programming (LP) problem, with the flux balance equations in (2.19) as constraints, i.e.

$$\begin{aligned} &\max_{v_i} \mu = \sum_{i=1}^{n} w_i v_i \\ &s.t.\ \ \boldsymbol{\mathcal{A}}\boldsymbol{v} = \boldsymbol{\mathcal{b}} \end{aligned} \tag{2.20}$$

where w_i are the amounts of the growth precursors required per gram of biomass. Then, assuming that at every time step the growth rate is maximized by the organism the consumption or production of species can be calculated with time as follows:

$$\begin{aligned} &\max_{x,\boldsymbol{v}} \mu \\ &s.t.\ \ \frac{d\boldsymbol{z}}{dt} = \boldsymbol{\mathcal{A}}\boldsymbol{v}x, \qquad \frac{dx}{dt} = \mu x, \qquad \mu = \boldsymbol{w}^T\boldsymbol{v} \end{aligned} \tag{2.21}$$

where x and $\boldsymbol{z}$ are the current biomass and metabolites' concentrations respectively. This dynamic modeling approach of the cell metabolism is referred to as Dynamic Flux Balance Modeling (DFBM) and it has been applied successfully to explain the microbial growth of Escherichia coli (E.coli) in batch reactor (Mahadevan, Edwards, & Doyle, 2002). Another reported DFBM application described the ethanol producing yeast Saccharomyces cerevisiae (Hjersted & Henson, 2006). Both Mahadevan's and Henson's studies introduced additional kinetic rate constrains to achieve realistic flux distribution $\boldsymbol{v}$, metabolite concentration $\boldsymbol{z}$ as well as biomass concentration x. Moreover, this model has been used to accomplish robust NMPC with feedback controller currently (Kumar & Budman, 2017). A key advantage of the DFBM approach is its ability to fit data with a relatively smaller number of kinetic parameters in contrast with other structured models where each possible reaction is modelled by a separate kinetic term. The parsimonious nature of these modelling approach makes them attractive since they have the potential to avoid overfitting of noisy and limited amount of data. Due to these advantages, these models have received in the last decade significant attention from the pharmaceutical research community (Kumar & Budman, 2017). Therefore, the robust algorithms developed in the current work are based on this modelling approach and they exploit its particular mathematical structure to reduce computation time.

2.7 Summary

Economic Model Predictive Control (EMPC) has recently received attention by both academics and industrial practitioners since it has the ability to simultaneously control the system while optimizing an economic profit functions whereas in the past control and optimization has been traditionally conducted by separate (independent) algorithms. Thus, EMPC has the potential to simplify the implementation of advanced control and optimization strategies in chemical plants. Beyond this advantage, the ability of EMPC for maximizing a profit function along a dynamic time horizon confers it with a significant advantage over strategies that separately optimize and control the process since in such strategies the optimization module is often based on steady state information only. Robustness to model error is a crucial topic in the design of NMPC or EMPC algorithms that must be addressed to promote its adoption in industrial settings (Findeisen et al., 2003; Lalo Magni & Scattolini, 2010). Robustness is particularly critical for EMPC algorithms that do not explicitly minimize the feedback error. Though there are several robust NMPC algorithms that have been reported, the limitations are often related to computational demands, i.e.

their online implementation is often prohibitive due to the computational cost of the uncertainty propagation step that is based on sampling such as Monte Carlo simulations. Polynomial Chaos Expansions (PCE) and Power Series Expansions (PSE) representations have also been reported for uncertainty propagation and for designing robust controllers but when the numbers of uncertain parameters and states increases, e.g. more than ~10 parameters, these methods also become computationally demanding. These past studies motivate the need to develop efficient uncertainty propagation methods for which the computational effort does not increase significantly with the number of uncertain parameters and states. This thesis presents a novel algorithm to address robustness of an EMPC algorithm that is specifically applied to a process described by a dynamic metabolic flux model. Although the presented technique is specifically tailored to dynamic metabolic flux models, the problem is very relevant since such models are expected to be the standard modelling approach for biotechnological processes in the future.

Chapter 3
Convex Cone Methodology and Case Study

Model Predictive Control (MPC) based on linear models has become the standard multivariable control strategy in the process industries. However, when the process to be controlled exhibits highly nonlinear dynamic behavior, a nonlinear model is preferred for more accurate prediction that will result in improved performance. The resulting nonlinear model based predictive control strategy is referred to as Nonlinear Model Predictive Control (NMPC). The typical cost function that is generally minimized in either linear MPC or NMPC algorithms is the sum of squared feedback errors between the set point and the measured values of the controlled variables. On the other hand, there is an alternative family of NMPC algorithms where the function to be optimized describes an economic profit instead of the minimization of the feedback error done in NMPC. Due to the nature of the economic cost function this algorithm is referred to as Economic Model Predictive Control (EMPC). As stated in Chapter 2 this algorithm has recently received significant attention by both academics and industry since it has the ability to simultaneously control the system while optimizing an economic profit whereas in the past the control and optimization tasks have been traditionally conducted by separate algorithms. Thus, EMPC has the potential to simplify the implementation of advanced control and optimization in chemical plants. Beyond this advantage, the ability of EMPC of maximizing profit along a dynamic time horizon confers it with a significant advantage over strategies that separately optimize and control the process since in such strategies the optimization module is based on steady state models. The case study in the current work is based on an Economic MPC (EMPC) formulation which will be presented in the current chapter.

Robustness to model error remains as an essential challenge for the design of NMPC methods (Findeisen et al., 2003; Lalo Magni & Scattolini, 2010) and particularly for EMPC algorithms since the latter do not explicitly minimize the feedback error. As discussed in Chapter 2, there are several reasons that may contribute to the model-plant mismatch, such as disturbances arising from changes in operating conditions, uncertainty derived from simplifications of the model structure or model reduction, inaccurate understanding of physical mechanisms of the process, lack of knowledge of key parameters, etc. Therefore, robustness is crucial in the design of NMPC or EMPC algorithms to promote its adoption in industrial settings.

Robust NMPC is generally based on min-max formulations, where the maximization action is conducted with respect to model uncertainties or bounded disturbances and then, for the worst case calculated by the previous maximization, an economic cost is minimized with respect to the control actions (Findeisen et al., 2003; Lalo Magni & Scattolini, 2010).

In Chapter 2, several previously reported robust NMPC algorithms, which propagate model uncertainties and/or disturbances by different means as well as the limitations for each of these methods were reviewed. For instance, Monte Carlo simulations have been used to propagate model uncertainties and disturbances but the online implementation of this approach is often prohibitive due to its computational cost (Birge & Louveaux, 2011; Kawohl et al., 2007; Niederreiter, 1978; Srinivasan et al., 2003).

Polynomial Chaos Expansions (PCE) have also been used to propagate uncertainty to provide robustness to NMPC strategies. This approach has considerable advantages over Monte Carlo based algorithms in terms of computational time making it significantly more amenable for online implementation (Kumar & Budman, 2017). However, when the numbers of uncertain parameters and states is very large, e.g. more than ~10 parameters, this method also becomes computationally demanding.

These past studies motivate the finding of uncertainty propagation methods for which the computational effort does not increase significantly with the number of uncertain parameters and states. This chapter presents a novel algorithm to address robustness of an EMPC that is specifically applied to a process that is described by a dynamic metabolic flux model. Thus, although the presented technique is specifically tailored to dynamic metabolic flux models, the problem is very relevant since such models are increasingly becoming the standard approach for modelling biotechnological processes.

Dynamic metabolic models, as described in chapter 2, are given by a Linear Programming optimization where a biological cost is optimized with respect to reactions' fluxes subject to kinetic and positivity constraints. Since some of these constraints involve parameterized kinetic expressions in their RHS (right hand side), addressing robustness for these models is equivalent to studying the sensitivity of the LP based model to uncertainties in the parameters of the RHS expressions. In the following section, an approach for sensitivity analysis of the RHS of the constraints within an EMPC formulation will be introduced. Then, a series of rigorous proofs are presented to support the validity of the approach. Subsequently, two novel methods for sensitivity analyses with respect to changes in the RHS of the constraints within the LP problem describing the dynamic metabolic model will be formulated and investigated.

3.1 Proposed Robust EMPC

The robust EMPC algorithm proposed in this section is specifically tailored to biochemical processes that are described by a dynamic metabolic flux model (DMFM). Since the DMFM is formulated as a constrained LP problem the uncertainty propagation approach proposed here is based on the sensitivity analyses of the RHS of the constraints of the LP problem.

Although the method has been specifically tailored to DMFM problems, in principle, it could be also applied to any EMPC formulation where the internal model can be described by an LP. The general formulation of an EMPC controller involves the minimization of an economic terminal cost/penalty V_f as follows:

$$\min_{u} V_f\big(\hat{x}(k+p)\big) \quad \text{s. t.} \tag{3.1}$$

$$\hat{x}(i+1) = f_d(\hat{x}(i), u(i), \boldsymbol{d} = 0) \tag{3.2}$$

$$\hat{x}(i=0) = x(k) \tag{3.3}$$

$$f\big(\hat{x}(i), u(i)\big) = 0 \;\; i = \{0,1,2,\ldots,p\}, k \in \mathbb{N} \tag{3.4}$$

$$\hat{x}(k+p) \in \mathbb{X}_f \tag{3.5}$$

where x_m is measured plant states, $\hat{x}$ is prediction of plant states, $u = \{u(k), u(k+1), \ldots, u(k+m)\}$ are the decision variables and they are equal to the manipulated variables or control actions, $\boldsymbol{d}$ is the vector of disturbance or it may also represent the effect of model uncertainty. At each time interval indexed as $[i, i+1)$, the future predicted states' trajectories are calculated over a prediction horizon p, as a function of a sequence of manipulated variables defined over a control horizon m, where $m \leq p$. The nominal dynamic model defined by (3.2) is used to predict the future nominal evolution of the system and it is initialized with each state's corresponding measurement given by equation (3.3). Equations of (3.2) and (3.4) represents the process or system constrains of EMPC system. Furthermore, the constraint (3.5) is a terminal constraint, which ensures that at the end of horizon the predicted $\hat{x}$ will be within a neighborhood of a terminal value defined by the terminal set $\mathbb{X}_f$. If maximization of a profit is desired rather than minimization of a cost, it is straightforward to convert such maximization into a minimization by considering the negative value of the profit that has to be maximized.

Figure 3.1 is a schematic block diagram of the EMPC controller. The controller uses a nominal model for predicting future values of the states. If the model is perfect, i.e. the model is equal to the process and disturbances $\boldsymbol{d} = 0$, then the feedback signal to the EMPC controller is zero. However, it is clear from figure 3.1 that if either there is mismatch between the process and the model or there are unmeasured disturbance d entering the system, the feedback signal returned to the EMPC controller will not be equal to zero. Since the model is never perfect and disturbances may continuously enter the process this will cause the system to evolve along different time trajectories from the nominal dynamic internal model given by (3.2) and referred as the model in Figure 3.1. For the particular case treated in the current research that the process is described by a DMFM, equations (3.2) to (3.5) in the formulation above are substituted by an LP problem describing the dynamic metabolic model. Thus, the resulting control problem involves a two level optimization where the outer level involves minimization of cost with respect to the control actions in the final time t_f of the process which has been shown in (3.6) and the inner level involves the LP problems f_{LP} as in (3.7) to (3.11).

$$\begin{aligned}
&\min_{u} V_f\left(\hat{\psi}_{obj}(t_f)\right)\\
&\text{s.t.}\\
&\qquad \hat{\psi}(t+1) = f_{LP}\big(\hat{\psi}(t), u(t), \boldsymbol{d}\big)\\
&\qquad \hat{\psi}(t=0) = \psi_m(t=k)\\
&\qquad f_c\big(u(t)\big) = 0
\end{aligned} \tag{3.6}$$

$$\begin{aligned}
&\max_{\boldsymbol{v}} \hat{\psi}_{obj} = \boldsymbol{wv}\\
&\text{s.t.}
\end{aligned} \tag{3.7}$$

$$\boldsymbol{Av} \le \boldsymbol{b} \tag{3.8}$$

$$\boldsymbol{b} = f_b(\boldsymbol{\psi}(t), \boldsymbol{v_{max}}) \tag{3.9}$$

$$\boldsymbol{\psi}(t+1) = \boldsymbol{\psi}(t) + \int_t^{t+1} \boldsymbol{Av}\psi_{obj} dt \tag{3.10}$$

$$\boldsymbol{v} \ge 0, \qquad \boldsymbol{b} \ge 0 \tag{3.11}$$

where a specific element from $\boldsymbol{\psi}$, i.e. ψ_{obj} is a biological objective function such as growth rate or ATP production, $\boldsymbol{A}$ is a stoichiometric matrix describing the set of reactions occurring for the particular microorganism and $\boldsymbol{w}$ is a vector of weights describing the contribution of the fluxes to the biological objective ψ_{obj}, $\boldsymbol{v}$ is the vector of optimal solutions of the inner level optimization, $\boldsymbol{\psi}$ is the vector elements are the concentrations of all metabolites, and the uncertain parameters are on the Right Hand Side (RHS) $\boldsymbol{b}$. The vector $\boldsymbol{v}$ represents, within the DMFM model, the fluxes of the reactions composing the metabolic network. Thus, the robust EMPC problem consists in solving a bi-level optimization in the presence of uncertainty in the RHS of the constraints of the inner level, i.e. in terms of deviations $\Delta\boldsymbol{b}$.

Generally, nonlinear terms arise in the original inner LP optimization in equation (3.9) when uncertainty is considered. For instance, if a function of b_i is formulated as this form:

$$b_i = \frac{v_{i,max}\psi_i}{K + \psi_i} \tag{3.12}$$

where K is assumed to be a constant parameter while $v_{i,max}$ is uncertain. At the beginning of any sampling interval, if $\psi_i = \overline{\psi_i} + \Delta\psi_i$. The substitution of these uncertain values into a Monod type kinetic term, that is typically used to describe the bounds (RHS) on consumption/production rates of metabolites, can be easily split into a nominal term plus a perturbation as follows:

$$\overline{b_i} + \Delta b_i = \frac{v_{i,max}\overline{\psi_i}}{K + \overline{\psi_i}} + \frac{v_{i,max}K\Delta\psi_i}{(K + \overline{\psi_i})(K + \overline{\psi_i} + \Delta\psi_i)} \tag{3.13}$$

Although Δb_i is nonlinear from (3.13), using the fact that $K \geq 0$, $\boldsymbol{v} \geq 0$ and $\psi_i \geq 0$, it is possible to that the perturbation of b_i is linear with respect to the uncertainty of concentration ψ_i since the partial derivative of Δb_i with respect of $\Delta\psi_i$ is positive:

$$\frac{\partial\Delta b_i}{\partial\Delta\psi_i} = \frac{Kv_{i,max}}{(K + \psi_i)^2} \geq 0 \tag{3.14}$$

The positivity of the derivative according to (3.14) is crucial since it implies that the extreme values of the metabolites ψ_i occur at the extreme values of the perturbations Δb_i. Thus, the bounds of the metabolite's concentrations can be obtained by substituting the extreme values of the deviations Δb_i into the linear equations valid for each tableau. This result can be extended to the entire vector describing the RHS of the inequality constraints in the LP problem as follows:

$$\boldsymbol{b} = \overline{\boldsymbol{b}} + \Delta\boldsymbol{b} \tag{3.15}$$

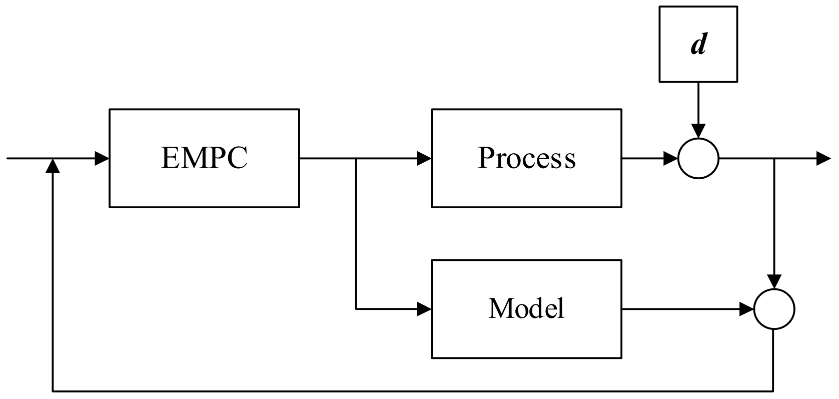


Figure 3.1 The control process in one time interval

A strategy is formulated in this work to solve the min-max problem in presence of uncertainty. This strategy requires the nonlinear terms arising from uncertainty to be approximated by linearized equations at each time interval. Then the maximum and minimum states and cost of the DMFM in the presence of uncertainty in the RHS terms of the LP can be calculated by the following sequence:

i. Calculate the uncertainty range $\left(\Delta \boldsymbol{v}|_{\Delta\psi_{obj}}, \Delta\psi_{obj}\right)$, i.e. maximum variations in the fluxes and cost.

ii. Conduct the outer optimization for the maximal variations in states obtained in step i.

iii. Integrate the concentrations one step ahead using the minimum and maximum values of the states resulting from the variations in the states calculated in step ii in combination with the Euler integrations of the equation that defined in (3.10).

iv. Go back to step i.

Step i involving the propagation of the parametric uncertainty onto the states is the most challenging one (Findeisen et al., 2003; Lucia et al., 2013; Lalo Magni & Scattolini, 2010). Therefore, establishing a method for this uncertainty propagation step is the key emphasis of this chapter. Step 1 requires a sensitivity analysis on RHS, since the goal is to investigate the effect of RHS on the states. A novel convex cone method is proposed in this thesis as an efficient RHS sensitivity analysis that is based on the generation of a map of the optimization solutions in the space defined by the RHS vector of the LP problem (DMFM) as explained further below in this Chapter.

To simplify the problem the function in RHS $\boldsymbol{b}$ are assumed to be positive linear or have been approximated into a linearized positive form. LP problems such as the one in the inner optimization in equation (3.7) are solved, as discussed in Chapter 2, by finding an active set of constraints for particular values of the RHS of the constraints. However, in the presence of uncertainty in the RHS, many solutions may arise corresponding to many possible active sets. Each active set define a linear problem for which the uncertainty can be easily propagated by a set of linear equations. Thus, although the problem for each active set of constraints is linear, in the presence of uncertainty the overall problem is nonlinear since it involves solutions for different active sets of constraints.

Following the explanations given above related to the block diagram in Figure 3.1 the disturbances and/or uncertainty $\boldsymbol{d}$ are lumped together since their effect is to cause a nonzero feedback signal back to the EMPC controller. The lumped effect due to disturbance and model uncertainty is denoted in the rest of the chapter by the uncertainty symbol Δ, if not otherwise specified.

3.2 Sensitivity Analysis of the RHS of the LP problem (DMFM)

As discussed in the previous section, the sensitivity analysis of the RHS is the basis for the propagation of uncertainty in the LP. As shown above, it is assumed that the nonlinear terms arising from the introduction of uncertainty are of bilinear form and can be properly bounded as shown in equation (3.15). Thus the problem boils down to assess the changes in the solution of the LP to simultaneous changes in RHS.

The sensitivity analysis of RHS is mainly depending on finding a set of tableaus where in each one of them the shadow prices remains constant and can be used for evaluation of the effect of changes in b_i (elements of the vector defining the RHS of the LP) on Z. Thus, the shadow prices, which are the partial derivative of Z with respect of $\boldsymbol{b}$, remain constant within each tableau where the latter correspond to a specific active set of constraints. The shadow prices can be organized into a Jacobian matrix of Z or $\boldsymbol{x}$ with respect to changes in $\boldsymbol{b}$. Since the LP representing the dynamic metabolic flux model represents a strategy of the cell to allocate resources to optimize a particular biological objective, the different tableaus obtained for a particular range of uncertainty in the RHS can be viewed as representing different strategies adopted by the cell to optimize its biological objective.

In the following section key concepts and methods to perform sensitivity analysis of the LP solution with respect to changes in the RHS of the constraints will be introduced. Then, a series of lemmas and proofs will be given to support the proposed sensitivity analysis approaches. Finally, two novel methods of sensitivity analyses of RHS space will be formulated and investigated.

3.2.1 Introduction and Motivation

In this section some key concepts and approaches will be described intuitively with figures for clarity of the presentation.

Fig. 3.2 illustrates the solution space of an LP that involves only two inequalities corresponding to RHS elements b_1 and b_2. The rectangle in Fig. 3.2 represents the region of uncertainty in the RHS at a particular time interval as defined by maximal deviations in b_1 and b_2.

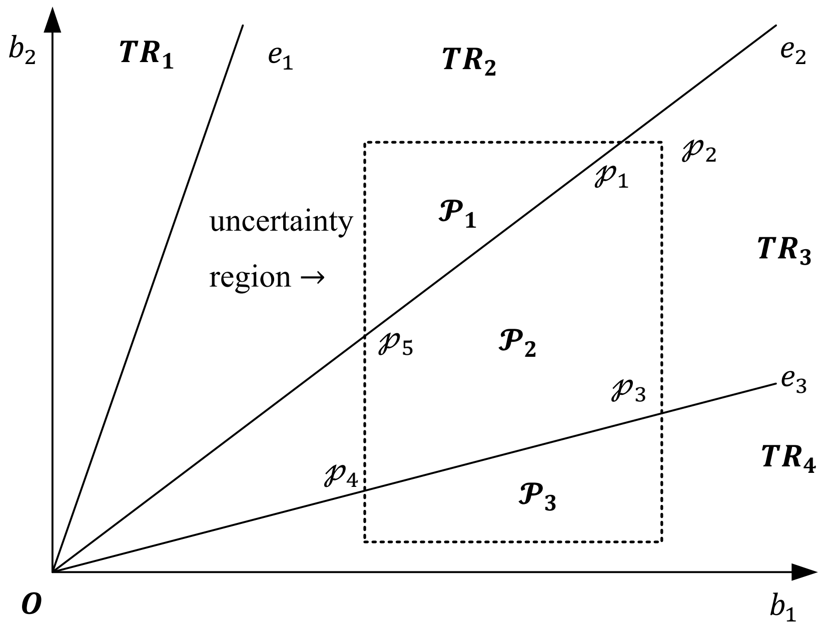


Figure 3.2 Uncertainty region and distribution of tableau regions in RHS space

It is first hypothesized, and later proven in section 3.2.2, that different regions of solutions of the LP problem can be identified in the space described by the elements of the RHS b_i's where each such region correspond to a specific tableau of the LP problem. Thus, it will be shown that the regions of solutions, each corresponding to a tableau, can be described in the RHS space by a series of adjacent non-overlap cones, e.g. $\boldsymbol{TR_1}$ to $\boldsymbol{TR_4}$ in Fig. 3.2. The edges of each cones can be described by the equations corresponding to each of the constraints at their equality limit e_1 e_2 and e_3. Then, since within each tableau region $\boldsymbol{TR_i}$ the shadow prices change linearly with respect to changes in the RHS and the basic solutions remain constant, the effect of changes in the RHS on the cost can be easily calculated. Thus, it is crucial to identify the regions corresponding to each tableau, , i.e. $\boldsymbol{\mathcal{P}_1}$, $\boldsymbol{\mathcal{P}_2}$ and $\boldsymbol{\mathcal{P}_3}$ for each corresponding tableaus, within the uncertainty region (rectangle in Fig 3.2) since it provides and efficient way of quantifying the effect of uncertainty on the cost Z.

For example, in Fig. 3.2 there are 5 different vertices $\wp_1$ to $\wp_5$ for the uncertainty region $\boldsymbol{\mathcal{P}_2}$ within the total uncertainty region described by the rectangle in the Figure. As shown later in the chapter these vertices can be efficiently calculated from the intersections of the maximal uncertainty region with the equations corresponding to the inequality constraints at their equality limit. Since the shadow prices are linear inside $\boldsymbol{\mathcal{P}_2}$, it is straightforward to obtain the maximum and minimum values of the cost Z as well as each of the basic solutions $\boldsymbol{x}$ in this tableau by solving a set of linear equations with respect to changes in the values of the elements of $\boldsymbol{b}$.

After calculating extreme (maximum/minimum) values of metabolites and costs within each tableau it is proposed to assign a particular probability for each of the solutions to occur. If the possible range of parameters values defining a particular tableau is defined as $\boldsymbol{P_{TR_i}}$, then, the possibility of any solution within $\boldsymbol{TR_i}$ is also can be obtained as:

$$P_{TR_i} = \frac{V_{\mathcal{P}_i}}{\sum_{i=1}^{\mathcal{K}_{RS}} V_{\mathcal{P}_i}} \tag{3.16}$$

under the assumption of uniform probability distribution of parameters, i.e. uniform distribution of $\boldsymbol{b}$ in the RHS space, where the number of active tableaus in the uncertainty region is $\mathcal{K}_{RS} = 3$ in this situation. The detailed description of this procedure will be further detailed in section 3.2.4. It should be noticed that although the 100% rule is a well-established tool for sensitivity analysis with respect to changes in the RHS, it is only necessary and not sufficient. Thus, it will be shown by an example in section 3.2.3 that the 100% rule is inefficient for uncertainty propagation as required in this thesis.

Following the above arguments, the main goal is to generate the different regions in the RHS variable space where each such region corresponds to a tableau. Finding the distribution of these regions in the RHS space will be referred heretofore as *generating an RHS map* While this task is relatively straightforward in the two dimensional example given above, it becomes significantly more complex in higher dimensions.

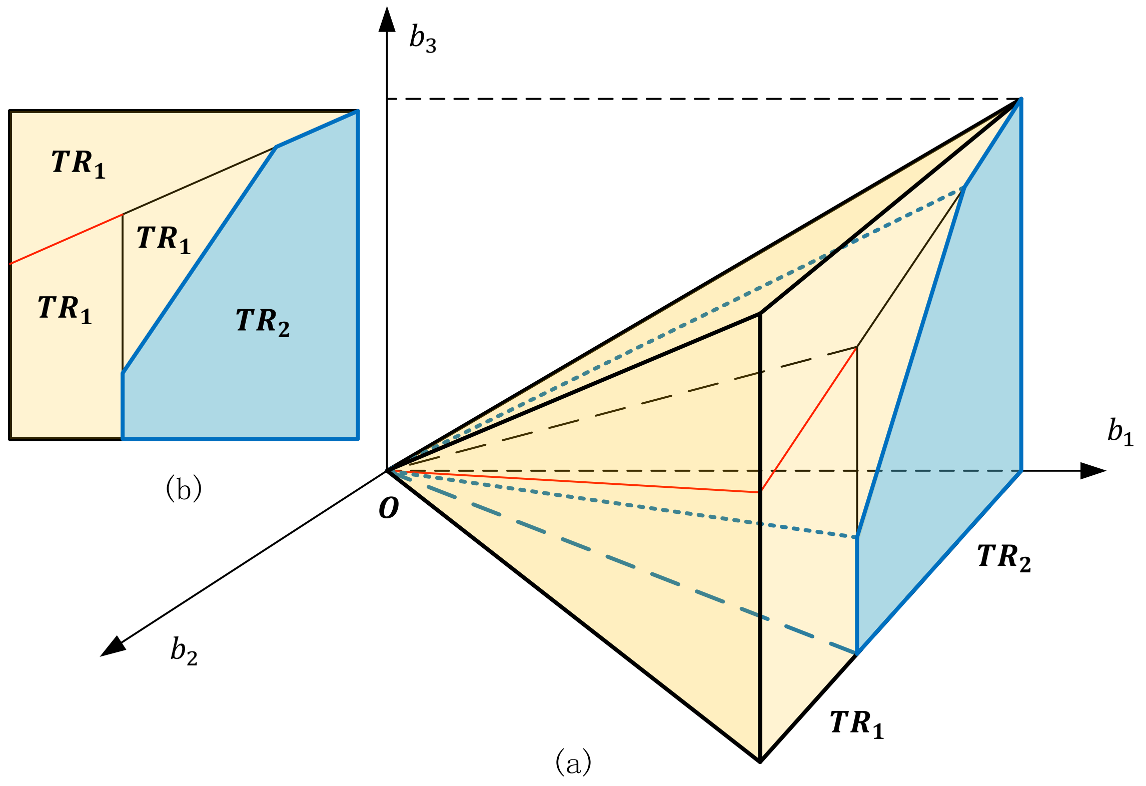


Figure 3.3 Distribution of two tableaus in a 3 dimensions RHS map

To illustrate the complexities associated with generating an RHS map in higher dimensions, a problem with 3 inequality constraints is used for illustration. The axes in Fig. 3.3 correspond to each one of the RHS bounds, i.e. b_1, b_2 and b_3, for each of the 3 inequalities considered in this illustrative problem. A main difficulty is that the cones that describe the region corresponding to each tableau in the RHS space are not necessarily convex in higher dimensions. The reason that convexity is lost derives from the occurrence of overlaps between regions delimited by the constraints of the problem. It is imperative to address this overlap since counting specific regions more than once will introduce error in the probability calculation given by equation (3.16).

An instance of two tableaus in a 3 dimensions RHS map is illustrated in Fig. 3.3 and corresponding projections in Fig. 3.4. In Fig. 3.3(a), two tableau regions are shown with different colors, the $\boldsymbol{TR_1}$ is in orange and the $\boldsymbol{TR_2}$ is in blue. Fig. 3.3(b) is the right side view of Fig. 3.3(a); it is obvious that in Fig. 3.3(b) although $\boldsymbol{TR_1}$ is not a convex polyhedron it can be described by 3 different convex cones to be referred to as sub-cones in the following discussion. Fig. 3.4 describes the procedure for generating these sub-cones generated:

i. In Fig. 3.4(a), the two constraints e_1 and e_2 divide the space into two polyhedrons $\boldsymbol{\mathcal{P}_1}$ and $\boldsymbol{\mathcal{P}_2}$. The shadow price strategy corresponding to tableau 1 ($\boldsymbol{TR_1}$) is used to obtain a feasible solution for both $\boldsymbol{\mathcal{P}_1}$ and $\boldsymbol{\mathcal{P}_2}$, but the shadow price strategy corresponding to tableau 2 ($\boldsymbol{TR_2}$) is used to obtain a feasible solution in $\boldsymbol{\mathcal{P}_1}$ only. Thus, the region $\boldsymbol{\mathcal{P}_1}$ is an overlap region. Since the initial polyhedron of $\boldsymbol{TR_1}$ and $\boldsymbol{TR_2}$ will be shown by a proof in section 3.2.2 to be convex, $\boldsymbol{\mathcal{P}_1}$ is an intersection of two convex polyhedron and thus $\boldsymbol{\mathcal{P}_1}$ is guaranteed to be convex. However, $\boldsymbol{\mathcal{P}_2}$ is the result of the subtraction of a convex polyhedron from another convex polyhedron, which cannot guarantee to be convex. Therefore, two different strategies need to be applied for overlap region $\boldsymbol{\mathcal{P}_1}$ and the non-convex region $\boldsymbol{\mathcal{P}_2}$, respectively.

ii. Fig. 3.4(b) shows the two different strategies that are applied for $\boldsymbol{\mathcal{P}_1}$ and $\boldsymbol{\mathcal{P}_2}$. For $\boldsymbol{\mathcal{P}_1}$, a new constraint e_3 is added and then the side of this constraint that results in a better optimum is identified by the shadow price strategy. In this example, e_3 is dividing $\boldsymbol{\mathcal{P}_1}$ into $\boldsymbol{\mathcal{P}_3}$ and $\boldsymbol{\mathcal{P}_4}$, and the superior optimal solution strategy is identified to belong to $\boldsymbol{TR_2}$ for $\boldsymbol{\mathcal{P}_3}$ and $\boldsymbol{TR_1}$ to $\boldsymbol{\mathcal{P}_4}$. Similarly for $\boldsymbol{\mathcal{P}_2}$, one of the edge constraints e_1 is extended to obtain two convex polyhedrons $\boldsymbol{\mathcal{P}_5}$ and $\boldsymbol{\mathcal{P}_6}$ where both of them belong to Tableau $\boldsymbol{TR_1}$.

iii. Fig. 3.4(c) shows that the overlap region and $\boldsymbol{\mathcal{P}_5}$ and $\boldsymbol{\mathcal{P}_6}$ have been allocated as follows:

$\boldsymbol{TR_2} = \{\boldsymbol{\mathcal{P}_3}\}$, and $\boldsymbol{TR_1} = \{\boldsymbol{\mathcal{P}_4}, \boldsymbol{\mathcal{P}_5}, \boldsymbol{\mathcal{P}_6}\}$

The theoretical basis and the detailed steps of the procedure outlined above are further described in 3.2.2 and 3.2.4.

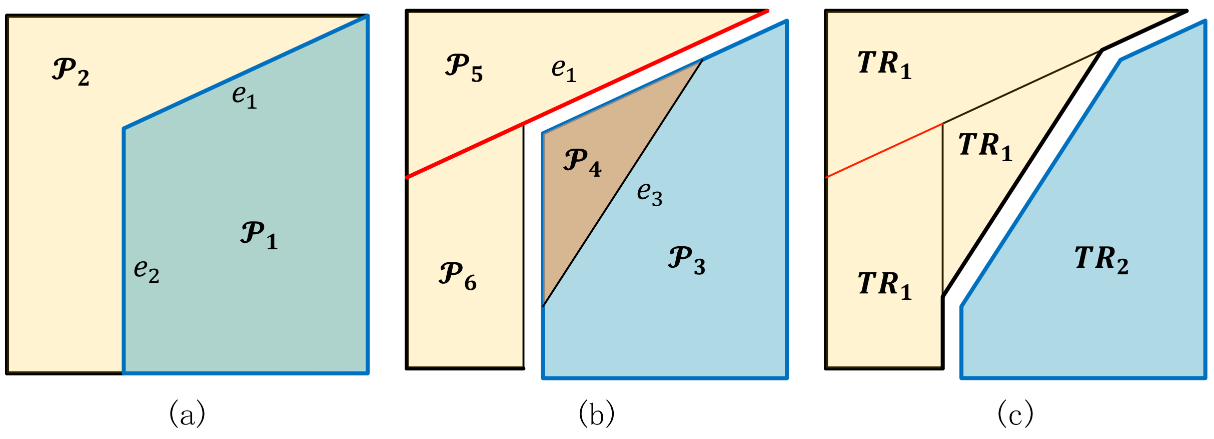


Figure 3.4 Procedure of the sub-cones generating

3.2.2 Map of RHS

In this section the method for generating the map of regions in the RHS space, each corresponding to a tableau, is presented. Using the notation presented in Chapter 2, the standard simplex form of an LP problem is as follows:

$$min\, Z = [-\boldsymbol{c} \quad \boldsymbol{0}] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} \tag{3.17}$$

s. t.

$$[\boldsymbol{A} \quad \boldsymbol{I}] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} = \boldsymbol{b} \tag{3.18}$$

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} \geq 0, \;\; \boldsymbol{b} \geq 0$$

$$\boldsymbol{x_s} = \begin{bmatrix} x_{s1} \\ x_{s2} \\ \vdots \\ x_{sm} \end{bmatrix} = \begin{bmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \end{bmatrix} \tag{3.19}$$

where $\boldsymbol{I}$ is $m \times m$ identity matrix, $\boldsymbol{x_s}$ is a column vector of slack variables that is used to obtain the augmented form of the problem where inequalities are converted into equalities and $\boldsymbol{b}$ is a column vector of m dimensional inequalities.

Definition 1. An LP sensitivity analysis function where this function mapping is the solution of the cost function Z is represented as $\mathcal{Z}(\boldsymbol{b}): RS \to \mathbb{R}$, where $RS \subseteq \mathbb{R}^{m+}$ is the RHS space of $\boldsymbol{b}$, $\forall b_i \in \mathbb{R}^+, i \;=\; 1,\; 2,\; \ldots,\; m.$

The basic solutions of a specific tableau are the solutions of the m equations in (3.18), where n nonbasic variables $\boldsymbol{x_{NB}}$ from the $n + m$ elements of $\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix}$ are eliminated by equating them to zero which leaves a set of m equations in m unknowns where the latter are referred to as the basic variables. This set of equations that is used to solve the basic variables is represented as follows:

$$\boldsymbol{Bx_B} = \boldsymbol{b} \tag{3.20}$$

where the vector of basic variables

$$\boldsymbol{x_B} = \begin{bmatrix} x_{B1} \\ x_{B2} \\ \vdots \\ x_{Bm} \end{bmatrix}$$

is obtained by eliminating the nonbasic variables from $\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix}$, and the basis matrix

$$\boldsymbol{B} = \begin{bmatrix} B_{11} & \cdots & B_{1m} \\ \vdots & \ddots & \vdots \\ B_{m1} & \cdots & B_{mm} \end{bmatrix}$$

is generated by eliminating the coefficients multiplying the nonbasic variables from the matrix $[\boldsymbol{A}\ \ \boldsymbol{I}]$. Then, using this standard formulation, the Simplex procedure can be applied as discussed in Chapter 2 using the pivoting concept to obtain the basic variables. After completing the Simplex procedure, the inverse matrix $\boldsymbol{B}^{-1}$ of the resulting matrix $\boldsymbol{B}$ is obtained, and the solution for the basic variables $\boldsymbol{x_B}$ and the solution Z of the cost function is computed as per the following equations:

$$\boldsymbol{x_B} = \boldsymbol{B}^{-1}\boldsymbol{b} \tag{3.21}$$

$$Z = \boldsymbol{c_B}\boldsymbol{x_B} = \boldsymbol{c_B}\boldsymbol{B}^{-1}\boldsymbol{b} \tag{3.22}$$

where, $\boldsymbol{c_B}$ is a vector which elements are the objective function coefficients, including zeros added to multiply the slack variables in the objective.

However, in the presence of uncertainty in $\boldsymbol{b}$ the map from Z to RS given by Definition 1 as

$$Z = \mathcal{Z}(\boldsymbol{b}) = \boldsymbol{c_B}\boldsymbol{x_B} \tag{3.23}$$

may result in more than one combination (more than one tableau), i.e. more than one $\boldsymbol{x_B}$ from which all possible combination of $\boldsymbol{X}$, i.e. $\boldsymbol{X^p}$, can meet same Z at same RS point, where $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix}$ such that $\left(x_j \middle| x_j \in \boldsymbol{x_{NB}}\right) = 0$ and $(x_{si}|x_{si} \in \boldsymbol{x_{NB}}) = 0$ (i relates to the number of inequality constraint $i = 1, \dots, m$), $j = 1, 2, \dots, n$, $\forall \boldsymbol{X} \in \mathbb{R}^{(m+n)+}$, $\forall X_k \in \mathbb{R}^{+}$, $k = 1, 2, \dots, m+n$.

Definition 2. A matrix generating function $\mathcal{X}(\boldsymbol{b})\colon RS \to \mathbb{R}^{m \times \mathcal{C}+}$, where $\mathcal{C} = C^{m}_{m+n}$, the result of this function is a $(m+n)$ by $\mathcal{C}$ matrix which column vectors are in $\boldsymbol{X^p}$ (all the possible solutions) $\boldsymbol{x}^p_{nc} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix}$. Thus X^p_{nc} is one of the possible solution vector corresponding to a point $\boldsymbol{b}$ in RS, i.e. $\forall X^p_{nc} \in \boldsymbol{X^p}$, $nc = 1, 2, \dots, \mathcal{C}$.

Then, the cost corresponding to each possible solution is:

$$Z^p_{nc} = \boldsymbol{c}^p_{nc}\boldsymbol{x}^p_{nc}$$
$$\text{such that } \boldsymbol{X^p} = \mathcal{X}(\boldsymbol{b}), \text{ and } \forall \left(Z^p_{nc}\middle|\mathcal{X}(\boldsymbol{b})\right) \in \left(Z^p\middle|\mathcal{X}(\boldsymbol{b})\right) \tag{3.24}$$

where $\boldsymbol{c}^p_{nc} = f_{T_{nc}}([-\boldsymbol{c}\quad \boldsymbol{0}])$, i.e. $\boldsymbol{c}^p_{nc}$ is the coefficients of the vector $[-\boldsymbol{c}\quad \boldsymbol{0}]$ which multiply nonzero x values of a solution $\boldsymbol{x}^p_{nc}$.

Definition 3. Any possible solution $\boldsymbol{x}_{nc}^{p}$ obtained for a particular element of $\boldsymbol{b}$ from RS corresponds to a certain tableau T_{nc}, $T_{nc} \in \mathbb{R}^{1\times m}$, is a row vector which elements describe the indices of the nonzero variables (basic variables $\boldsymbol{x_B}$) in the solution $\boldsymbol{x}_{nc}^{p}$. For instance, if $\boldsymbol{X}$ have 4 elements, and the $\boldsymbol{x_B}$ of $\boldsymbol{x}_{nc}^{p}$ is the first and fourth elements from $\boldsymbol{X}$, then $T_{nc} = [1\ 4]$. For this simple example, $\boldsymbol{c}_{nc}^{p}$ defined in Definition 2 is a row vector that contains the first and fourth element of the vector $[-\boldsymbol{c} \quad \boldsymbol{0}]$ followed by m zeros, i.e. m is the number of slack variables (inequality constraints in the problem).

Additionally, from equation (3.18), we can easily get all possible solutions $\boldsymbol{x}_{nc}^{p}$ for $\boldsymbol{\mathcal{T}}$ tableau based on Definition 3 as follows:

$$\boldsymbol{A^p}\boldsymbol{x}_{nc}^{p} = \boldsymbol{b} \tag{3.25}$$

where $\boldsymbol{A^p} \in \mathbb{R}^{m\times m}$ is the square matrix corresponding to $\boldsymbol{x}_{nc}^{p}$, such that $\boldsymbol{A^p} = f_{T_{nc}}([\boldsymbol{A} \quad \boldsymbol{I}])$, where the square matrix $\boldsymbol{A^p}$ is selected by extracting the number $\imath$ columns from $[\boldsymbol{A} \quad \boldsymbol{I}]$, where $\imath \in T_{nc}$, i.,e by considering only the columns of A that are multiplying basic variables $\boldsymbol{x_B}$ of the solution.

Theorem 1. Let RS be the set of all vectors $\boldsymbol{b}$, where $\boldsymbol{b}$ is referred to as a specific point in space RS, i.e. a particular combination of possible b_i values. Let assume that $\boldsymbol{A}$ and $\boldsymbol{c}$ are constant and that the elements in the vector $\boldsymbol{b}$ are linearly independent. For $\forall \boldsymbol{b} \in RS$, at least one column vector $\boldsymbol{x}_{nc}^{p}$ of the $\boldsymbol{X^p}$ from $\mathcal{X}(\boldsymbol{b})$, i.e. tableau T_{nc}, exist that satisfies the corresponding cost function solution $Z_{nc}^{p} = \mathcal{Z}(\boldsymbol{b})$, i.e. an optimal solution.

Proof. Applying Cramer's Rule (Poole, 2014) to equation in (3.25) for a specific type tableau T_{nc}:

$$\boldsymbol{x}_i = \frac{\det\left(\boldsymbol{A_i^p}\right)}{\det(\boldsymbol{A^p})} \tag{3.26}$$

where $\boldsymbol{x}_i$ is a column vector corresponding to one of the possible optimal solution matrix $\boldsymbol{x}_{nc}^{p} = [x_1, x_2, \ldots, x_m\,]^T = \left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{x_s}\end{bmatrix}\right)_{T_{nc}}$, $nc = 1, 2, \ldots, \mathcal{C}$, $\boldsymbol{A_i^p}$ is the matrix formed by replacing the number i column of $\boldsymbol{A^p}$ by the column vector $\boldsymbol{b}$. Thus, $\boldsymbol{A_i^p}$ is in terms of $\boldsymbol{A^p}$ and $\boldsymbol{b}$, additionally, $\boldsymbol{A^p} = ([\boldsymbol{A} \quad \boldsymbol{I}])_{T_{nc}}$. Since $\boldsymbol{A}$ is constant for a specific tableau T_{nc}, then x_i and consequently $\boldsymbol{x}_{nc}^{p}$ are functions of $\boldsymbol{b}$. This function can be represented as follows:

$$\boldsymbol{x}_{nc}^{p} = \mathcal{F}_{nc}(\boldsymbol{b}) \tag{3.27}$$

Meanwhile, Definition 3 indicated that there are at a set of T_{nc} in the RS region where the latter is defined in Definition 1. From (3.24) and (3.27), for a specific vector $\boldsymbol{b}$:

$$Z_{nc}^{p} = \boldsymbol{c}_{nc}^{p}\boldsymbol{x}_{nc}^{p} = \boldsymbol{c}_{nc}^{p}\mathcal{F}_{nc}(\boldsymbol{b}) \tag{3.28}$$

Thus, there are $\mathcal{C}$ potential optimal solution $\boldsymbol{X^p}$ corresponding to $\boldsymbol{Z^p}$ in terms of $\boldsymbol{b}$. As the goal of an LP program is to find a minimum value of Z_{nc}^{p} in $\boldsymbol{Z^p}$ then:

$$\begin{gathered}\mathcal{Z}(\boldsymbol{b}) = \min_{\boldsymbol{\mathcal{T}}} \boldsymbol{Z^p} \\ \boldsymbol{\mathcal{T}} = \{T_1, T_2, \dots, T_{\mathcal{C}}\}, \boldsymbol{Z^p} = \{Z_1^p, Z_2^p, \dots, Z_{\mathcal{C}}^p\}\end{gathered} \tag{3.29}$$

Thus for $\forall \boldsymbol{b} \in RS$, $\exists \boldsymbol{x}_{nc}^{p} \in \boldsymbol{X^p}$ such that $Z_{nc}^{p} = \mathcal{Z}(\boldsymbol{b})$. □

Theorem 1 shows that for an LP problem when the matrix $\boldsymbol{A}$ and vector $\boldsymbol{c}$ are constant, certain point $\boldsymbol{b}$ in the space of RHS, i.e. RS region, would always result in at least one specific tableau T_{nc}, where shadows prices $\boldsymbol{B}^{-1}$ and basic solutions $\boldsymbol{x_B}$ can be determined. If there is more than one tableau that satisfies the optimal solution $\mathcal{Z}(\boldsymbol{b})$ at the same point $\boldsymbol{b}$ of the RS region, then this region is referred as an *overlap* region

Remark: it is assumed in this work that a condition where more than one $\boldsymbol{x}_{nc}^{p}$ satisfies a same set of vectors $\boldsymbol{b}$ as well as Z in a corresponding RS region will not be occurred. If such situation is encountered in future studies, the condition should be identified, and procedure should be developed to address it.

Definition 4. The function $\mathcal{Cut}(\mathcal{P}_1, con) = \mathcal{P}_2 + \mathcal{P}_3$, where con is a constraint that may divide a polyhedron $\mathcal{P}_1$ into two sub-tableaus $\mathcal{P}_2$ and $\mathcal{P}_3$.

Also, $\mathcal{Con}(\mathcal{P}_1)$ are the constraints that bound the polyhedron $\mathcal{P}_1$. There is at most only one empty polyhedron in $\{\mathcal{P}_2, \mathcal{P}_3\}$, where an empty polyhedron is defined as one for which there is no feasible solution for a given set of constraints. The procedure of the function $\mathcal{Cut}(\mathcal{P}_1, con)$:for generating sub-tableaus (see Section 3.1.1) is as follows:

i. Generate another constraint as the negative of the constraint con, i.e. $-con$.

ii. Then $\mathcal{P}_2 \triangleq \{\mathcal{Con}(\mathcal{P}_1), con\}$ and $\mathcal{P}_3 \triangleq \{\mathcal{Con}(\mathcal{P}_1), -con\}$. If $\mathcal{P}_1$ is a convex polyhedron, then it can be guaranteed that $\mathcal{P}_2$ and $\mathcal{P}_3$ are also convex.

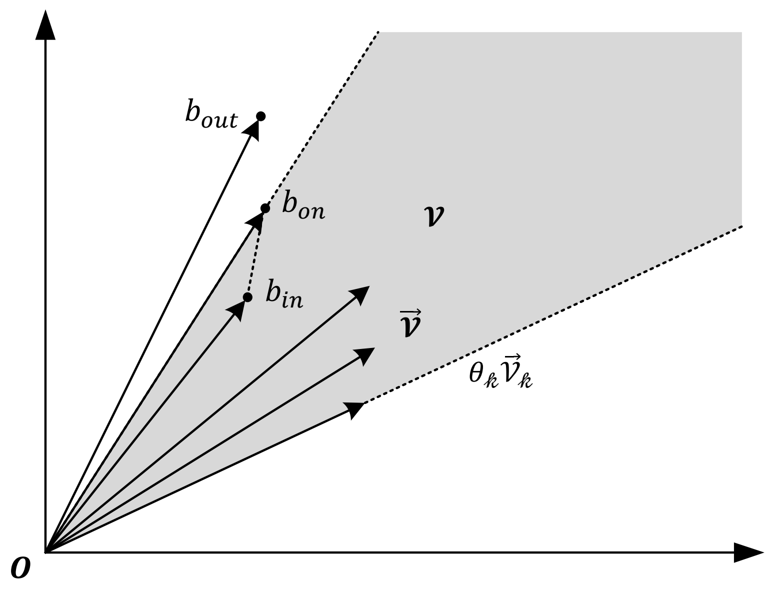

b_{out}
b_{on}
$\boldsymbol{\mathcal{V}}$
b_{in}
$\vec{\boldsymbol{\mathcal{V}}}$
$\theta_{k}\vec{\mathcal{V}}_{k}$
$\boldsymbol{O}$

Figure 3.5 The convex cone $\boldsymbol{\mathcal{V}}$ (shown shaded) of set $\vec{\boldsymbol{\mathcal{V}}}$

Lemma 1. If a polyhedron is defined by a set of constraints as in (3.30), where $\boldsymbol{\alpha} \in \mathbb{R}^{1\times m}$ is a row vector with m coefficients, $\boldsymbol{b} \in \mathbb{R}^{m\times 1}$, then this polyhedron is a convex cone.

$$\boldsymbol{\alpha b} \leq 0 \tag{3.30}$$

Proof. A convex cone $\boldsymbol{\mathcal{V}}$ is an affine set space (a space generated by affine vectors $\vec{\mathcal{V}}_{k}$ that start from the origin $\boldsymbol{O}$), which is defined by a set $\vec{\boldsymbol{\mathcal{V}}}$ of vector $\vec{\mathcal{V}}_{k}$ (Berger, Pansu, Berry, & Saint-Raymond, 2013), as what have been shown in function (3.31) and the shaded pie slice in Fig. 3.5.

$$\left\{\sum_{k=1} \theta_{k}\vec{\mathcal{V}}_{k} \middle| \vec{\mathcal{V}}_{k} \in \boldsymbol{\mathcal{V}}, \theta_{k} \geq 0, k = 1,2,3,\dots\right\} \tag{3.31}$$

If the polyhedron defined by (3.30) is not an affine set space, there must be at least an affine vector in RS passing across a point (b_{on}) on the constraint and crossing through a point either inside (b_{in}) or outside (b_{out}) of the polyhedron $\mathcal{P}$. Let assume the vector $\overrightarrow{b_{on}}$ defined by b_{on} and the vector $\overrightarrow{b_{\imath n}}$ defined by b_{in} are collinear with each other, then is $\exists\lambda \in \mathbb{R}$, such that:

$$\overrightarrow{b_{on}} = \lambda\overrightarrow{b_{\imath n}} \tag{3.32}$$

Since b_{on} is on the constraints (3.30), then vector $\overrightarrow{b_{on}}$ must satisfies:

$$\boldsymbol{\alpha}\overrightarrow{b_{on}} = 0 \tag{3.33}$$

Substituting (3.32) into (3.33), then $\lambda\boldsymbol{\alpha}\overrightarrow{b_{\imath n}} = 0 \Rightarrow \boldsymbol{\alpha}\overrightarrow{b_{\imath n}} = 0$.Thus, the point b_{in} is also on the boundary of the polyhedron, which contradicts the definition of b_{in} that was assumed to be a point within $\mathcal{P}$. So, there is no affine vector across $\mathcal{P}$, all vector $\vec{V}_{\mathscr{k}}$ would be either inside or outside of this cone. Therefore, the polyhedron $\mathcal{P}$ bounded by the constraints in (3.30) is a convex cone. □

The following theorem is given to provide a theoretical basis for the generation of the map of tableaus T_{nc} in RS.

Theorem 2. Let RS be the set of all vectors $\boldsymbol{b}$, where $\boldsymbol{b}$ is referred to as an explicit point in RHS space RS. Let assume that $\boldsymbol{A}$ and $\boldsymbol{c}$ have constant elements and the elements in vector $\boldsymbol{b}$ are linearly independent. For $\forall\boldsymbol{b} \in RS$, the distribution region of each T_{nc}, i.e. TR_{nc}, is a polyhedral region that can be described as a set of convex cones CH and the pointwise supreme of this set is also convex.

Proof. As discussed for equation (3.28), for a specific tableau T_{nc} , a corresponding solution $\boldsymbol{x}_{nc}^{p}$ is a function of $\boldsymbol{b}$. Then using (3.27), the function $\mathcal{F}_{nc}(\boldsymbol{b})$ is given as follows:

$$\mathcal{F}_{nc}(\boldsymbol{b}) = \boldsymbol{\alpha_{nc}}\boldsymbol{b}$$
$$\boldsymbol{\alpha_{nc}} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1m} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mm} \end{bmatrix} \tag{3.34}$$

where $\boldsymbol{\alpha_{nc}}$ is a set of coefficients corresponding to the tableau T_{nc}. Since, $\boldsymbol{x}_{nc}^{p} = \boldsymbol{\alpha b}$, then for each x_i from $\boldsymbol{x}_{nc}^{p}$:

$$x_i = \boldsymbol{\alpha_i b} \tag{3.35}$$

where $\boldsymbol{\alpha_i}$ is the i-th row vector from $\boldsymbol{\alpha_{nc}}$, where each of the elements in $\boldsymbol{\alpha_i}$ can be calculated with Cramer's rule as follows:

$$(\boldsymbol{\alpha_i})_{\boldsymbol{i}} = \partial\left(\frac{\det(\boldsymbol{A_i^p})}{\det(\boldsymbol{A^p})}\right)\Big/\partial b_i \tag{3.36}$$

Since $\boldsymbol{x_B} > \boldsymbol{0}$, then $\forall X_i \geq 0$, from (3.36), then a subset of the constraints of the TR_{nc}, defined asTR_{nc}^{*}, are given as follows:

$$-\boldsymbol{\alpha_{nc}b} \leq 0 \tag{3.37}$$

By substitute (3.34) into (3.28), Z_{nc}^{p} of T_{nc} is obtained:

$$Z_{nc}^{p} = \boldsymbol{c}_{nc}^{p}\boldsymbol{\alpha}_{nc}\boldsymbol{b} \tag{3.38}$$

Additionally, though many regions in RS with different T_{nc} are distinguished from each other by using (3.37), there still occur considerable overlap between regions. To avoid the consideration of regions more than once due to overlap, constraints are used that can explicitly divide the overlapping region into subset, and allocate each of these subsets to a one T_{nc} . These additional constraints that are needed to separate the overlapping regions are based on (3.28). Accordingly, the additional constraint/s are added so as to obtain the minimum Z_{nc}^{p} possible among all possible T_{nc}, i.e. T_{ic} ,$ic = 1,2,\dots,\mathcal{C}; ic \neq nc$, in each overlap region, where the overlap region is defined as OR_{ic}=$TR_{nc}^{*} \cap TR_{ic}^{*}$, the symbol $*$ means that this polyhedron has not been divided yet or tested yet for all possible overlaps after computing constrains in (3.37). The set of all TR_{ic}^{*} is referred as:

$$\boldsymbol{\mathcal{T}_{ic}} = \{TR_{ic}^{*} | ic = 1,2,\dots,\mathcal{C}; ic \neq nc\}$$

and the set of other Z_{nc}^{p} in terms of TR_{ic}^{*} is referred as $\left(Z_{ic}^{p}|TR_{ic}^{*}\right)$, thus the new necessary constraints that are used to separate one tableau from another tableau is emerging as follows:

$$Z_{nc}^{p} \leq \left(Z_{ic}^{p}|TR_{ic}^{*}\right) \tag{3.39}$$

By substituting (3.38) in to (3.39) for only one constraint in (3.39):

$$\boldsymbol{c}_{nc}^{p}\boldsymbol{\alpha}_{nc}\boldsymbol{b} \leq \boldsymbol{c}_{ic}^{p}\boldsymbol{\alpha}_{ic}\boldsymbol{b} \tag{3.40}$$

Or:

$$\left(\boldsymbol{c}_{nc}^{p}\boldsymbol{\alpha}_{nc} - \boldsymbol{c}_{ic}^{p}\boldsymbol{\alpha}_{ic}\right)\boldsymbol{b} \leq 0 \tag{3.41}$$

Thus, the use of constraints (3.41) results in a region to be divided into two sub-regions, where one sub-region is defined as $TR_{nc}^{*2,1}$ where (3.41) which constraints are as follows,

$$\begin{aligned} &-\boldsymbol{\alpha}_{nc}\boldsymbol{b} \leq 0 \\ &\left(\boldsymbol{c}_{nc}^{p}\boldsymbol{\alpha}_{nc} - \boldsymbol{c}_{ic}^{p}\boldsymbol{\alpha}_{ic}\right)\boldsymbol{b} \leq 0 \\ &-\boldsymbol{\alpha}_{ic}\boldsymbol{b} \leq 0 \end{aligned} \tag{3.42}$$

From (3.42), it is evident that all constraints can be converted into the following form:

$$\boldsymbol{\alpha b} \leq 0 \tag{3.43}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{1\times m}$, $\boldsymbol{b} \in \mathbb{R}^{m\times 1}$. Based on the discussion in Lemma 1, the polyhedron that is generated by constraint of the form of (3.43) is a convex cone.

Hence, the resulting TR_{nc}^{*} from the division of overlap regions is also a convex cone. And OR_{ic} ($OR_{ic} = TR_{nc}^{*} \cap TR_{ic}^{*}$) is also convex cone since it is the intersection of two convex cones.

However, another region defined by $TR_{nc}^{*2,*2} = TR_{nc}^{*} - TR_{ic}^{*}$, it is not necessarily convex, since it is generated by the subtraction of one convex polyhedron from another convex polyhedron. Then, as explained in an earlier section, the region resulting from the subtraction of two convex polyhedrons, must be further divided into smaller convex sub-regions (sub-tableaus):

$$TR_{nc}^{*2,km} = TR_{nc}^{*2,2}, TR_{nc}^{*2,3}, \dots, TR_{nc}^{*2,(m+1)},\ km = 2,3,\dots,m+1$$

by cutting off m times the region to be divided with one of the constrains given by $(\boldsymbol{\alpha_{ic}})_i \boldsymbol{b} \leq 0$ where each of these constraints is used only one time. The division of the non-convex region into convex sub-regions is done by using the $\mathcal{Cut}$ function given in definition 4:

$$\mathcal{Cut}\left(TR_{nc}^{*2,*km}, (\boldsymbol{\alpha_{ic}})_i \boldsymbol{b} \leq 0\right) = TR_{nc}^{*2,km} + TR_{nc}^{*2,*(km+1)} \tag{3.44}$$

where $i = km - 1$. After completing this division process, the constraints that describe any one of the resulting convex cones from $TR_{nc}^{*2,km}$ are given as follows:

$$\begin{aligned} &-\boldsymbol{\alpha_{nc} b} \leq 0 \\ &\boldsymbol{\rho_{km}} \cdot (\boldsymbol{\alpha_{ic} b}) \leq 0 \end{aligned} \tag{3.45}$$

where $\boldsymbol{\rho_{km}}$ is the $(km-1)$th column vector from $\boldsymbol{\rho}$, which is an $m \times m$ upper triangular matrix:

$$\boldsymbol{\rho_{km}} = (\boldsymbol{\rho})_{km-1}, \qquad \boldsymbol{\rho} = \begin{bmatrix} 1 & & \boldsymbol{-1} \\ & \ddots & \\ \boldsymbol{0} & & 1 \end{bmatrix} \tag{3.46}$$

Hence (3.47) implies that all the constraints for any $TR_{nc}^{*2,km}$, i.e. $con\left(TR_{nc}^{*2,km}\right)$, define a set of convex hulls. Then, using the constraints in (3.40) defined as TR_{nc}^{*2-1}, the resulting set of sub-regions (sub-tableaus) is obtained as $\boldsymbol{TR_{nc}^{*2}} = \left\{TR_{nc}^{*2,1}, TR_{nc}^{*2,2}, \dots, TR_{nc}^{*2,(m+1)}\right\}$. Thus, for each step of the division of a non-convex region into convex sub-regions at most $(m+1)$ convex cones are allocated to corresponding sub-tableaus $TR_{nc}^{*\mathcal{r},\mathcal{s}}$, $\mathcal{r} = 1,2,\dots,\mathcal{C}-1$, $\mathcal{s} = 1,2,\dots,(m+1)^{\mathcal{r}}$.

The procedure of dividing each overlap region and generating new region of sub-tableaus $TR_{nc}^{*\mathcal{r},\mathcal{s}}$ in the $\mathcal{r}$-th steps of procedure $\boldsymbol{\Pi}$ is summarized in Table 3.1:

a) Select basic tableau region TR_{ic}^{*} in $\boldsymbol{\mathcal{T}_{ic}}$ that have not been used to divide and generate new regions of sub-tableaus.

b)	Manipulating inner iteration (step i. to step ii) for $(m+1)^{r}$ times:
i.	Select one of the sub-tableaus $TR_{nc}^{*r,s}$ that generated from former r-th steps TR_{nc}^{*r} and have not been used to test overlap region in step b).
ii.	Calculate $TR_{nc}^{*(r+1),(s*2)} = TR_{nc}^{*r,s} \cap TR_{ic}^{*}$, which generate one new convex cone region of sub-tableau, and $TR_{nc}^{*(r+1),s} = TR_{nc}^{*r,s} - TR_{ic}^{*}$, which generate m new convex cones region of sub-tableau.
c)	Let $r = r + 1$. Then, the set of tasting tableaus in next iteration $\boldsymbol{TR_{nc}^{*r}} = \boldsymbol{TR_{nc}^{*(r+1)}}$
d)	Loop step a) to step c), for $\mathcal{C} - 1$ times, then let $\boldsymbol{TR_{nc}^{r}} = \boldsymbol{TR_{nc}^{*(r+1)}}$.

Table 3.1 Whole procedure of procedure Π

After applying one iteration from step a) to step c) of procedure **Π** (Table 1.) on $\boldsymbol{TR_{nc}^{*r}}$, a new set of sub-tableaus is obtained as follows:

$$\boldsymbol{TR_{nc}^{*(r+1)}} = \left\{TR_{nc}^{*(r+1),1}, TR_{nc}^{*(r+1),2}, \dots, TR_{nc}^{*(r+1),(m+1)^{(r+1)}}\right\} \tag{3.47}$$

Continuing iteration, the resulting set of sub-tableaus after r steps of the **Π** procedure is referred as:

$$\boldsymbol{TR_{nc}^{r}} = \left\{TR_{nc}^{r,1}, TR_{nc}^{r,2}, \dots, TR_{nc}^{r,(m+1)^{r}}\right\} \tag{3.48}$$

Thus, the constraints that define each element in $\boldsymbol{TR_{nc}^{r}}$, i.e. $TR_{nc}^{r,s}$, are of the following form:

$$-\boldsymbol{\alpha_{nc} b} \leq 0$$
$$\begin{bmatrix} \boldsymbol{\Lambda}_1 \cdot \boldsymbol{\Gamma}_1 \\ \boldsymbol{\Lambda}_2 \cdot \boldsymbol{\Gamma}_2 \\ \vdots \\ \boldsymbol{\Lambda}_{\mathcal{C}-1} \cdot \boldsymbol{\Gamma}_{\mathcal{C}-1} \end{bmatrix} \leq 0, \tag{3.49}$$

where:

$\boldsymbol{\Lambda}_{r} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$, $\boldsymbol{\Gamma}_{r} = \begin{bmatrix} \left(\boldsymbol{c_{nc}^{p}} \boldsymbol{\alpha_{nc}} - \boldsymbol{c_{ic}^{p}} \boldsymbol{\alpha_{ic}}\right)\boldsymbol{b} \\ -\boldsymbol{\alpha_{ic} b} \\ \boldsymbol{\rho_{km}} \cdot (\boldsymbol{\alpha_{ic} b}) \end{bmatrix}$, $ic = r$,and $\boldsymbol{\rho_{km}}$ is the function with element as in (3.46). It is evident that all the constraints in each $TR_{nc}^{r,s}$ are of the form (3.31) and thus, the polyhedron that $con(TR_{nc}^{r,s})$ defines is a convex cone as per Lemma 1.

Additionally, although $\boldsymbol{TR_{nc}^{r}}$ cannot be guaranteed to be convex, the pointwise maximum and minimum of $\boldsymbol{TR_{nc}^{r}}$ is convex. Since if f_1 and f_2 are convex, then pointwise maximum f, which is defined by $f(x) = \max\{f_1(x), f_2(x)\}$, is also convex (Stephen Boyd & Vandenberghe, 2004). That is: $\boldsymbol{TR}_{\boldsymbol{nc}}^{\boldsymbol{r}} = \max\left\{TR_{nc}^{r,1}, TR_{nc}^{r,2}, \dots, TR_{nc}^{r,(m+1)^{r}}\right\}$ is convex. □

Remark: Although this proof is based on a generic nonnegative orthant, i.e. $\boldsymbol{b} \geq \boldsymbol{0}$, Theorem 2 is still valid in each single orthant of the real number RHS space.

A map of RHS space can be generated by Theorem 2. In this map, the RHS space can be divided into different set of convex cones, which correspond to different tableaus.

The following section discusses an additional approach for generating an RHS map based on the 100 percent rule.

3.2.3 100 Percent Rule: Theory and Limitations

As discussed in Chapter 2, the 100 Percent Rule is based on formula (2.16). Based on the 100% rule for a specific tableau, the feasible range of change of each b_i can be computed by using the $\boldsymbol{B}^{-1}$ that is obtained in (2.16). The 100 Percent Rule states than when the sum of the percentage changes of all b_i is smaller than 1 (100%) the solution of the LP can be calculated with the current tableau. Thus, this method is generally used for analyzing the effect of simultaneous changes in RHS space. However, this rule is only providing a necessary condition but not sufficient one since it cannot provide a conclusive statement about the tableau if the sum does exceed 100 percent.

In this section, a method is proposed that can approximates the allowable feasibility region of b_i for each tableau based on the 100 Percent Rule. In this case the allowable feasibility region refers to the maximal range of simultaneous changes in parameters that are allowed for the LP to be solved by the same tableau. In this sense, the proposed method will expand upon the original 100% by approximating the entire region of feasibility rather than the limited region provided by the necessary but not sufficient 100% rule.

The argument and description of the original 100 Percent Rule on RHS was introduced in (Bradley, Hax, & Magnanti, 1977), where the Rule is formulated as:

$$\sum_{i=1}^{m} \frac{\Delta b_i}{\Delta \mathrm{b}_i^{max}} \leq 1 \tag{3.50}$$

where $\Delta \mathrm{b}_i^{max}$ is the calculated feasible range of b_i when only one of the b_i are allowed to change. For the application of the rule it is required that Δb_i and $\Delta \mathrm{b}_i^{max}$ must have the same sign. Thus, the 100 Percent Rule is actually generating a convex hull in the RHS space where the feasible solution is stable, i.e. it is obtained with the same tableau, with respect to changes of Δb_i. Then, this convex hull can be calculated a series of simplexes around the normal (nominal) point $\bar{\boldsymbol{b}}$.

Proof. To proof the mentioned property within a simple process, the changes in b_i are defined in terms of deviations with respect to the normal point $\bar{\boldsymbol{b}}$ where the latter is moved to the origin. Thus, in each orthant the signs of each coefficients in $\bar{\boldsymbol{b}}$ are constant. In any one of the orthant, the $\Delta \boldsymbol{b_{max}}$ defines a set of m points along the axes as:

$$\boldsymbol{\mathcal{p}_{set}} = \boldsymbol{I}\, \Delta \boldsymbol{b_{max}} \tag{3.51}$$

Each of these points is represented by row vector in $\boldsymbol{\mathcal{p}_{set}}$ and it is referred as $\boldsymbol{\mathcal{p}_i}$. Then, a simplex $\boldsymbol{Convs}$ (convex hull) in this orthant is generated along with the original point $\boldsymbol{O}$.

$$\boldsymbol{Convs} = \left\{ \theta_o \boldsymbol{O} + \sum_{i=1}^{m} \theta_i \boldsymbol{\mathcal{p}_i} \middle| \, \theta \geq 0, 1^T \theta = 1 \right\} \tag{3.52}$$

where $\theta \in \mathbb{R}^{(m+1)\times 1}$, θ_i is the fraction of point $\boldsymbol{\mathcal{p}_i}$ in the mixture convex combination of the points $\boldsymbol{\mathcal{p}_{set}}$. Since $\theta_o \boldsymbol{O} = 0$, and $0 \leq \theta_o \leq 1$, (3.52) can be converted into:

$$\boldsymbol{Convs} = \left\{ \sum_{i=1}^{m} \theta_i \boldsymbol{\mathcal{p}_i} \middle| \, \theta \geq 0, 1^T \theta \leq 1 \right\} \tag{3.53}$$

where $\theta \in \mathbb{R}^{m\times 1}$, (3.51) is a set of the points that within the simplex generated region. Substitute (3.51) into one point $\boldsymbol{b}$ of (3.53):

$$\boldsymbol{b} = \boldsymbol{\mathcal{p}_{set}} \theta \quad \theta \geq 0, 1^T \theta \leq 1 \tag{3.54}$$

Pre-multiply (3.54) with $\boldsymbol{\mathcal{p}}_{set}^{-1}$,then:

$$(\boldsymbol{I}\, \Delta \boldsymbol{b_{max}})^{-\boldsymbol{1}} \boldsymbol{b} = \boldsymbol{\mathcal{p}}_{set}^{-1} \boldsymbol{b} = \theta \quad \theta \geq 0, 1^T \theta \leq 1 \tag{3.55}$$

multiply (3.55) with 1^T,then:

$$\sum_{i=1}^{m} \frac{b_i}{\Delta \mathrm{b}_i^{max}} = \sum_{i=1}^{m} \theta_i = 1^T \theta \leq 1 \tag{3.56}$$

Since $\bar{\boldsymbol{b}}$ has already been moved to the origin $\boldsymbol{O}$, $\Delta \boldsymbol{b} = \boldsymbol{b}$, substitute it into (3.56):

$$\sum_{i=1}^{m} \frac{\Delta b_i}{\Delta \mathrm{b}_i^{max}} = \sum_{i=1}^{m} \theta_i \leq 1 \tag{3.57}$$

It is obvious from (3.57) that there is an equivalence between the definition of the orthant simplex in (3.52) and the 100 Percent Rule based region in the corresponding orthant as in (3.50). Thus, the simplex generated in each orthant by using the points $\boldsymbol{\wp}_{\boldsymbol{set}}$ defined with 100 Percent Rule with respect to the nominal point $\overline{\boldsymbol{b}}$ is actually the region of feasibility, i.e. solution is obtained with same tableau, when simultaneous changes in RHS space. Since all adjacent simplexes in this condition are sharing common vertices, the regions obtained from each single application of the 100 % rule can be joined (connected) together to form bigger regions of feasibility of the tableau. Thus, the convex hull generated by all these points $\boldsymbol{\wp}$ will also consist of simplexes. If a region is defined by the 100 Percent Rule, then this region is a convex polyhedron which belongs to a certain tableau. A conservative polyhedron that satisfied the 100 Percent Rule can be obtained by applying *convhulln* in MATLAB to generate a convex hull around the extreme vertices. □

Based on the previous discussion, the procedure of the 100 percent rule based method, which is referred heretofore as 100 Based Hull Method here, is as follows:

i. Select a point $\overline{\boldsymbol{b}}$ from the RHS space (RS), calculate the $\boldsymbol{x_B}$ and $\boldsymbol{B}^{-1}$ by using the Simplex method for Linear Programing (LP) Problem (The procedure of the Simplex algorithm was introduced in Section 2.4.1).

ii. Computing the maximal feasible range of b_i when only one of the b_i is allowed to change, i.e. $\Delta \mathrm{b}_i^{max}$, by using the formula that obtained from (2.16) such that $\boldsymbol{x_B} \geq \boldsymbol{0}$, which is:

$$\begin{aligned} &\boldsymbol{x_B} = \boldsymbol{B}^{-1}\left(\overline{\boldsymbol{b}} + \Delta \mathrm{b}_i^{max}\right) \\ &s.t. \\ &\boldsymbol{x_B} \geq \boldsymbol{0} \end{aligned} \tag{3.58}$$

iii. From (3.53), a series of allowable range of Δb_i are obtained, i.e. $\Delta \mathrm{b}_i^{max}$. Generating a matrix of corresponding point from $\Delta\boldsymbol{b}$, by apply the range of $\Delta\boldsymbol{b}$ on $\overline{\boldsymbol{b}}$, which is referred as:

$$\boldsymbol{\wp} = \begin{bmatrix} \wp_{11} & \wp_{12} \\ \vdots & \vdots \\ \wp_{m1} & \wp_{m2} \end{bmatrix} \tag{3.59}$$

where $\wp_{i1} = \overline{\boldsymbol{b}} + (\Delta b_i)_{min}$ and $\wp_{i2} = \overline{\boldsymbol{b}} + (\Delta b_i)_{max}$.

iv. Applying *convhulln* in MATLAB to generate a convex polyhedron around the extreme vertices $\boldsymbol{\wp}$ as well as $\overline{\boldsymbol{b}}$, an expression of each plane of this polyhedron also can be obtained. This convex hull is a conservative set in a region of a specific tableau where the same tableau can still be used to obtain the LP solutions.

To illustrate the method a simple example is shown as follows.

An LP problem with 2 inequalities is defined as follows:

$$
\begin{aligned}
&max\ Z = \boldsymbol{cx} \\
&\text{s. t.} \\
&\quad \boldsymbol{Ax} \leq \boldsymbol{b} \\
&\quad \boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix},\ \boldsymbol{c} = [3 \quad 5],\ \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \geq 0,\ \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0
\end{aligned}
\tag{3.60}
$$

Convert (3.57) into a standard initial tableau form as follows:

$$
\begin{aligned}
&min\ Z = [-\boldsymbol{c} \quad \boldsymbol{0}] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} \\
&\text{s. t.} \\
&\quad [\boldsymbol{A} \quad \boldsymbol{I}] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} = \boldsymbol{b} \\
&\quad \boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \boldsymbol{x_s} = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \geq 0
\end{aligned}
\tag{3.61}
$$

Equations (3.62), also illustrated in Fig. 3.6, indicate that there are 3 different regions in the RHS that belongs to different tableaus and the edges $\boldsymbol{e}$ of each tableau regions, i.e. $\boldsymbol{TR_{nc}}$ as per the notation used in the previous section, are marked as two dot lines e_1 and e_2. The sub-indexes 10, 11, and 01 in $\boldsymbol{nc}$ of $\boldsymbol{TR_{nc}}$ are used to denote the 3 different regions for which either x_1, $\{x_1, x_2\}$, and x_2 are basic variables in each $\boldsymbol{TR_{nc}}$, respectively. It should be noticed that the regions $\boldsymbol{TR}$ as well as edges $\boldsymbol{e}$ are generated by using the method proposed in section 3.2.4 and 3.3.

$$
\begin{aligned}
e_1 &= \{3b_1 - b_2 = 0 | \boldsymbol{b} \geq 0\} \\
e_2 &= \{b_1 - b_2 = 0 | \boldsymbol{b} \geq 0\}
\end{aligned}
\tag{3.62}
$$

Now apply the proposed 100% Based Hull Method, the first initial point is $\boldsymbol{\bar{b}_1} = (12,9)$. After applying the Simplex Algorithm for this point, the final tableau is shown as:

$$
\begin{bmatrix} \boldsymbol{A}^* & \boldsymbol{S}^* & \boldsymbol{b}^* \\ \boldsymbol{z}^* - \boldsymbol{c} & \boldsymbol{y}^* & z^* \end{bmatrix} = \begin{bmatrix} -2 & 0 & 1 & -1 & 3 \\ 1.5 & 1 & 0 & 0.5 & 4.5 \\ 4.5 & 0 & 0 & 2.5 & 22.5 \end{bmatrix}
\tag{3.63}
$$

The detailed description of the Simplex Algorithm as well as the parameters in final tableau as in (3.63) is given elsewhere (Hillier, 2001). From the final tableau (3.60), the $\boldsymbol{B}^{-1}$ is given by:

$$\boldsymbol{B}^{-1} = \boldsymbol{S}^{*} = \begin{bmatrix} 1 & -1 \\ 0 & 0.5 \end{bmatrix}$$

A series of inequalities can be obtained by using (3.58) (step ii of the 100 Based Hull Method above).

$$\begin{aligned} \boldsymbol{B}^{-1}(\overline{\boldsymbol{b}} + \Delta b_1) &= \begin{bmatrix} 1 & -1 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 12 + \Delta b_1 \\ 9 \end{bmatrix} \geq 0 \\ \boldsymbol{B}^{-1}(\overline{\boldsymbol{b}} + \Delta b_2) &= \begin{bmatrix} 1 & -1 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 12 \\ 9 + \Delta b_2 \end{bmatrix} \geq 0 \end{aligned} \tag{3.64}$$

Then, obtaining the range of each Δb_i by computing (3.64) and the results are as following:

$$\begin{aligned} &-3 \leq \Delta b_1 \\ &-9 \leq \Delta b_2 \leq 3 \end{aligned} \tag{3.65}$$

By substitute (3.65) into (3.59) in step iii. a series of points will be generated as (since $(\Delta b_1)_{max}$ does not exist the point $\mathcal{p}_{12}$ will be eliminated):

$$\mathcal{p}_{11} = (9,9), \qquad \mathcal{p}_{21} = (12,0), \qquad \mathcal{p}_{22} = (12,12) \tag{3.66}$$

Then a convex hull $\boldsymbol{\mathcal{P}_1}$ can be generated around $\{\overline{\boldsymbol{b}}_{\mathbf{1}}, \mathcal{p}_{11}, \mathcal{p}_{21}, \mathcal{p}_{22}\}$ by using *convhulln* in MATLAB, and the edge of this polyhedron can also be obtained as the bold line around $\overline{\boldsymbol{b}}_{\mathbf{1}}$in Fig. 3.6.

Following the same procedure, another conservative polyhedron $\boldsymbol{\mathcal{P}_2}$ in tableau range $\boldsymbol{TR_{11}}$ can also be obtained by applying $\overline{\boldsymbol{b}}_{\mathbf{1}} = (6,8)$ into 100 Based Hull Method and the set of $\mathcal{p}$ is:

$$\mathcal{p}_{11} = (2.67,8), \qquad \mathcal{p}_{12} = (8,8), \qquad \mathcal{p}_{21} = (6,6), \qquad \mathcal{p}_{22} = (6,18) \tag{3.67}$$

And the edge of polyhedron $\boldsymbol{\mathcal{P}_2}$ is also marked as the bold line around $\overline{\boldsymbol{b}}_{\mathbf{2}}$ in Fig. 3.6.

The limitations of 100 % Based Hull Method are obvious for this example as shown in Fig. 3.6 since it can only provide a necessary condition but not sufficient. Hence, it cannot give an explicit judgement if the sum does exceed 100 percent. Thus, uncertainty propagation cannot be efficiently accomplished by using this single algorithm only as compared to the algorithm presented in section 3.2.1. It is worth noticing that while in 2 dimensional cases, i.e. 2 inequalities as in the example discussed here this method can generate some edges of the conservative polyhedrons that can cover the edges of the corresponding tableaus, such as the line generated by $\mathcal{p}_{11}$ and $\mathcal{p}_{22}$ is equal to e_2, in higher dimensions of the RHS space this method is found to be very inefficient.

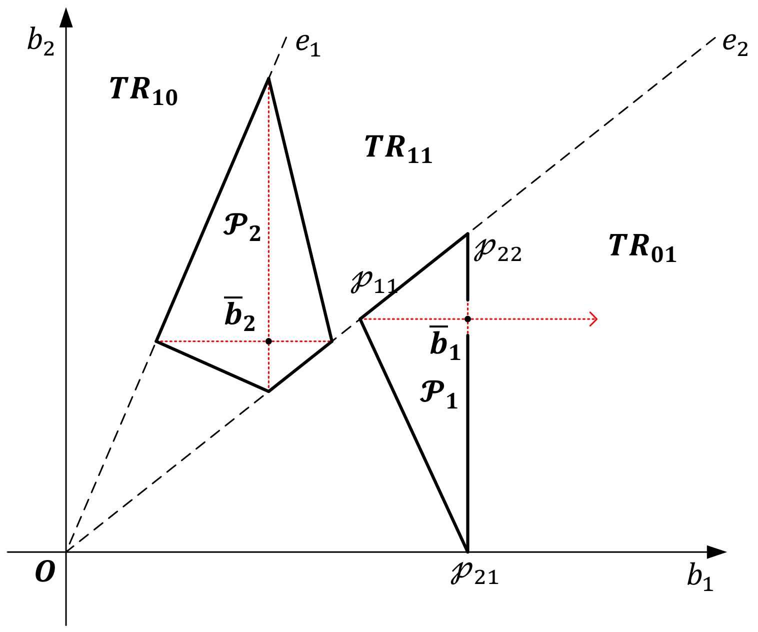


Figure 3.6 An example of RHS space by using 100 Based Hull Method

3.2.4 Convex Cone Method

Based on the discussion in 3.2.2, the main purpose of the RHS map generation algorithm is to find a set $\boldsymbol{TR_{nc}}$ where all of the convex cones TR_{nc}^{r} belongs to a specified tableau. However, in 3.2.3, the 100 Based Hull Method has been shown to be incapable of providing a complete description of the tableau distribution in RHS map in finite time. Therefore, an approach that addresses the limitations of 100 Based Hull Method is proposed in this section.

Since this method is based on the segmentation and allocation of the convex cones in RHS space, this method will be referred heretofore in this thesis as Convex Cone Method (CCM). The key ideas and basic procedures of CCM algorithm have already presented in 3.2.1 and 3.2.2, such as the dividing process of overlap regions in Fig. 3.4. Thus, this algorithm will be described with the same symbols and defined functions which have been introduced in the previous sections:

i. Compute a set $\boldsymbol{\mathcal{T}}$ where consist all possible $\boldsymbol{x}_{nc}^{p}$'s combinations of tableaus T_{nc} by using function f_{nck}, $nc = 1, 2, \dots, \mathcal{C}$, $\mathcal{C} = C_{m+n}^{m}$. This function can be formulated as following form:

$$T_{nc} = (\boldsymbol{\mathcal{T}})_{nc} = \left(f_{nck}(m, \boldsymbol{k})\right)_{nc} \tag{3.68}$$

where f_{nck} is a function that can generate a set $\boldsymbol{\mathcal{T}}$ where consists all of the possible combinations (T_{nc}) by choosing m elements from vector $\boldsymbol{k}$, $\boldsymbol{k} = 1,2,\dots,(m+n)$. This function can be accomplished by applying the function *nchoosek* in MATLAB.

ii. Calculate the basic solution vector $\boldsymbol{x}_{nc}^{p}$ corresponding to each T_{nc} by using $\boldsymbol{x}_{nc}^{p} = \left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{x_s}\end{bmatrix}\right)_{T_{nc}}$, where $\boldsymbol{x}_{nc}^{p}$ is the number i element from $\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{x_s}\end{bmatrix}$, where $i \in T_{nc}$. $\boldsymbol{A}_{nc}^{p} \in \mathbb{R}^{m\times m}$ is the square matrix corresponding to $\boldsymbol{x}_{nc}^{p}$, such that $\boldsymbol{A}_{nc}^{p} = ([\boldsymbol{A} \quad \boldsymbol{I}])_{T_{nc}}$, where the subscript T_{nc} indicates the number i columns from $[\boldsymbol{A} \quad \boldsymbol{I}]$, where $i \in T_{nc}$. $\boldsymbol{c}_{nc}^{p} = ([-\boldsymbol{c} \quad \boldsymbol{0}])_{T_{nc}}$, where the subscript T_{nc} indicates the number i element from $[-\boldsymbol{c} \quad \boldsymbol{0}]$, where $i \in T_{nc}$.

iii. Obtain each element x_i of the solution vector $\boldsymbol{x}_{nc}^{p}$ with respects to $\boldsymbol{b}$ by using (3.26) (Cramer's rule).

$$x_i = \frac{\det\left(\boldsymbol{A}_{nc,i}^{p}\right)}{\det\left(\boldsymbol{A}_{nc}^{p}\right)} \tag{3.26}$$

where $\boldsymbol{x}_{nc}^{p} = [x_1, x_2, \dots, x_m]^T = \left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{x_s}\end{bmatrix}\right)_{T_{nc}}$, $\boldsymbol{A}_{nc,i}^{p}$ is the matrix formed by replacing the number i column of $\boldsymbol{A}_{nc}^{p}$ by the column vector $\boldsymbol{b}$. Thus, $\boldsymbol{A}_{nc,i}^{p}$ is a function of $\boldsymbol{A}_{nc}^{p}$ and $\boldsymbol{b}$. Then the matrix $\boldsymbol{\alpha_{nc}}$ is defined as follows:

$$\boldsymbol{x}_{nc}^{p} = \boldsymbol{\alpha_{nc} b}$$

$$\boldsymbol{\alpha_{nc}} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1m} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mm} \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial b_1} & \frac{\partial x_1}{\partial b_2} & \cdots & \frac{\partial x_1}{\partial b_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial b_1} & \frac{\partial x_m}{\partial b_2} & \cdots & \frac{\partial x_m}{\partial b_m} \end{bmatrix} \tag{3.69}$$

Since $\boldsymbol{x}_{nc}^{p} = \boldsymbol{\alpha_{nc} b}$, and $\boldsymbol{x}_{nc}^{p} \geq 0$, the possible edges of tableau T_{nc} can be described as in (3.37).

$$-\boldsymbol{\alpha_{nc} b} \leq 0 \tag{3.37}$$

Meanwhile the cost function of T_{nc} is represent as in (3.38).

$$Z_{nc}^{p} = \boldsymbol{c}_{nc}^{p}\boldsymbol{\alpha_{nc} b} \tag{3.38}$$

From this step polynomial expression of edges and cost function with respect to $\boldsymbol{b}$ for every T_{nc} in $\boldsymbol{\mathcal{T}}$ are obtained. The region in defined by (3.37) is referred as TR_{nc}^{0}, and the resulting polyhedrons are convex cones (the proof of this is in 3.2.2).

iv. Select tableau T_{nc} from $\boldsymbol{\mathcal{T}}$, where the set of remaining tableaus are referred to as $\boldsymbol{\mathcal{T}_{ic}}$, $ic = 1,2,\dots,\mathcal{C}; ic \neq nc$. At this stage if there are tableaus for which is known a priori that they cannot be possible in the RHS space (RS) , then these tableaus can be eliminated from $\boldsymbol{\mathcal{T}}$ in advance. For example, in some case the tableau T_{nc} where the basic solutions are $\boldsymbol{x_B} = \begin{bmatrix}\boldsymbol{0}\\\boldsymbol{1}\end{bmatrix} \cdot \begin{bmatrix}\boldsymbol{x}\\\boldsymbol{x_s}\end{bmatrix}$ would only contribute the cost function $Z_{nc}^{p} = 0$ since $\forall \boldsymbol{x} = \boldsymbol{0}$. It is generally impossible for this type of tableaus to produce a competitive cost function solution than other tableaus where the cost functions $Z_{nc}^{p} \leq 0$.

v. Select a tableau T_{ic} from $\boldsymbol{\mathcal{T}_{ic}}$. Define a set of the convex cones that compose the region of T_{nc} ($\boldsymbol{TR_{nc}^{*}}$) as $\boldsymbol{TR_{nc}^{*}} = \{TR_{nc}^{*1}, TR_{nc}^{*2}, \dots, TR_{nc}^{*\mathcal{K}}\}$, where symbol $*$ means that this polyhedron has not been divided or further processed by further iterations, $\mathcal{K} \in \mathbb{N}$. At the start of these iterations, $\mathcal{K} = 1, \boldsymbol{TR_{nc}^{*}} = \{TR_{nc}^{*1}\} = TR_{nc}^{0}$. For each convex cone TR_{nc}^{*k} $(k = 1,2,\dots,\mathcal{K})$ in $\boldsymbol{TR_{nc}^{*}}$, a subset $\boldsymbol{TR_{nc}^{k}}$ is generated by adding a series of constrains, and each element in $\boldsymbol{TR_{nc}^{k}}$ generated in each iteration are referred as $TR_{nc}^{k,km}$:

$$TR_{nc}^{k,km} = \begin{cases} \begin{bmatrix} con(TR_{nc}^{*k}) \\ \left(\boldsymbol{c_{nc}^{p}\alpha_{nc}} - \boldsymbol{c_{ic}^{p}\alpha_{ic}}\right)\boldsymbol{b} \\ -\boldsymbol{\alpha_{ic}b} \end{bmatrix} \leq 0, & km = 1 \\ \begin{bmatrix} con(TR_{nc}^{*k}) \\ \boldsymbol{\rho_{km}} \cdot (\boldsymbol{\alpha_{ic}b}) \end{bmatrix} \leq 0, & 2 \leq km \leq m+1 \end{cases} \tag{3.70}$$

where $km = 1,2,\dots,m+1$, $\boldsymbol{\rho_{km}}$ is a column vector defined in (3.46):

$$\boldsymbol{\rho_{km}} = (\boldsymbol{\rho})_{km-1}, \qquad \boldsymbol{\rho} = \begin{bmatrix} 1 & & \boldsymbol{-1} \\ & \ddots & \\ \boldsymbol{0} & & 1 \end{bmatrix} \tag{3.46}$$

$con(TR)$ means all of the constraints that may contribute to define the edge of TR and these constraints can be extracted as a property belonging to the corresponding objects TR. After the calculation in (3.65), the subset $\boldsymbol{TR_{nc}^{k}}$ is obtained:

$$\boldsymbol{TR_{nc}^{k}} = \left\{TR_{nc}^{k,1}, TR_{nc}^{k,2}, \dots, TR_{nc}^{k,m+1}\right\} \tag{3.71}$$

After all of the $\boldsymbol{TR_{nc}^{k}}$ have been manipulated by (3.70), the set of $\boldsymbol{TR_{nc}^{*}}$ is obtained as:

$$\boldsymbol{TR}_{\boldsymbol{nc}}^{*} = \begin{Bmatrix} \boldsymbol{TR}_{\boldsymbol{nc}}^{\boldsymbol{1}}, \\ \boldsymbol{TR}_{\boldsymbol{nc}}^{\boldsymbol{2}}, \\ \dots, \\ \boldsymbol{TR}_{\boldsymbol{nc}}^{\boldsymbol{\mathcal{K}}} \end{Bmatrix} = \begin{Bmatrix} TR_{nc}^{1,1}, TR_{nc}^{1,2}, \dots, TR_{nc}^{1,m+1}, \\ TR_{nc}^{2,1}, TR_{nc}^{2,2}, \dots, TR_{nc}^{2,m+1}, \\ \dots, \\ TR_{nc}^{\mathcal{K},1}, TR_{nc}^{\mathcal{K},2}, \dots, TR_{nc}^{\mathcal{K},m+1} \end{Bmatrix} \tag{3.72}$$

Renumber the convex cone elements $TR_{nc}^{k,km}$ in $\boldsymbol{TR}_{\boldsymbol{nc}}^{*}$ such that:

$$\begin{gathered} \boldsymbol{TR}_{\boldsymbol{nc}}^{*} = \left\{TR_{nc}^{1}, TR_{nc}^{2}, \dots, TR_{nc}^{kn}, \dots, TR_{nc}^{(\mathcal{K}-1)(m+1)+km}\right\} \\ TR_{nc}^{kn} = TR_{nc}^{k,km}, \qquad kn = (k-1)(m+1) + km \end{gathered} \tag{3.73}$$

Save the set in (3.73) as a property belonging to the corresponding object $\boldsymbol{TR}_{\boldsymbol{nc}}^{*}$ and the $\mathcal{K}$ in this set is reassigned such that $\mathcal{K} = (\mathcal{K}-1)(m+1) + km$ as well as $k = kn = 1,2,\dots,\mathcal{K}$.

vi. Eliminate empty TR_{nc}^{k} as well as the redundant constraints in each convex cone TR_{nc}^{k} from $\boldsymbol{TR}_{\boldsymbol{nc}}^{*}$. This procedure can be accomplished by using the *noredund* algorithm from *lcon2vert* package in MATLAB (Matt, 2017). Avoiding the inclusion of the origin is important when eliminating redundant constraints by using these mentioned methods. Since all of the tableau region TR_{nc}^{k} have a vertex on the origin, then all the constrains are weakly redundant at that point. Thus, it is impossible to distinguish the redundant constraints from the no redundant constrains when the origin is included. Therefore, some additional constraints should be initially added in the constraints set of every TR_{nc}^{k}, such that:

$$(TR_{nc}^{k})_{nr} = \begin{bmatrix} con(TR_{nc}^{k}) \\ -1^{T}\boldsymbol{b} + \Phi_{min} \\ 1^{T}\boldsymbol{b} - \Phi_{max} \end{bmatrix} \leq \boldsymbol{0}, \qquad 0 < \Phi_{min} < \Phi_{max} \tag{3.74}$$

where $\Phi \in \mathbb{R}$. The function $-1^{T}\boldsymbol{b} + \Phi_{min} \leq 0$ ensures that the neighborhood of the origin is not considered when assessing redundancy and the function $1^{T}\boldsymbol{b} - \Phi_{max} \leq 0$ ensures that the polyhedron of $(TR_{nc}^{k})_{nr}$ exists in a finite space thus can be recognized and eliminated by most redundant constraints eliminating algorithms. These additional constraints that related with Φ are referred as $\boldsymbol{f}_{\boldsymbol{ac}}$.

vii. Renumber the TR_{nc}^{k} that still remain after the elimination of non-redundant constraints (step vi) and let the number $\mathcal{K}$ equals to the amount of the existing convex cones. Thus, $\boldsymbol{TR}_{\boldsymbol{nc}}^{*} = \{TR_{nc}^{*1}, TR_{nc}^{*2}, \dots, TR_{nc}^{*\mathcal{K}}\}$. Then, select another tableau region T_{ic} from $\boldsymbol{\mathcal{T}}_{\boldsymbol{ic}}$ that have not been manipulated yet, return to step v and iteration step v. vi. vii. until all of the T_{ic} have been used for checking the edge of the tableau region set $\boldsymbol{TR}_{nc}^{*}$.

viii. After all the T_{ic} have been used for checking the edge of the tableau region set $\boldsymbol{TR}_{nc}^{*}$ the final set of a certain tableau T_{nc}'s region $\boldsymbol{TR}_{nc}$ that consist of a series of convex cones is obtained, such that:

$$\boldsymbol{TR}_{nc} = \{TR_{nc}^{1}, TR_{nc}^{2}, \dots, TR_{nc}^{\mathcal{K}_{nc}}\} \tag{3.75}$$

where $\mathcal{K}_{nc}$ is the number of convex cones that contains in the final tableau region $\boldsymbol{TR}_{nc}$.

ix. Select another tableau T_{nc} from $\boldsymbol{\mathcal{T}}$ and iterate step v. to step viii. until the final tableau region $\boldsymbol{TR}_{nc}$ is computed.

x. The profile of RHS map, i.e. RS, is obtained once all of the final tableau region $\boldsymbol{TR}_{nc}$ in terms of T_{nc} in $\boldsymbol{\mathcal{T}}$ have been generated as follows:

$$RS = \begin{Bmatrix} \boldsymbol{TR}_1, \\ \boldsymbol{TR}_2, \\ \dots, \\ \boldsymbol{TR}_{\mathcal{K}_{RS}} \end{Bmatrix} = \begin{Bmatrix} TR_1^1, TR_1^2, \dots, TR_1^{\mathcal{K}_1}, \\ TR_2^1, TR_2^2, \dots, TR_2^{\mathcal{K}_2}, \\ \dots, \\ TR_{\mathcal{K}_{RS}}^1, TR_{\mathcal{K}_{RS}}^2, \dots, TR_{\mathcal{K}_{RS}}^{\mathcal{K}_{\mathcal{K}_{RS}}} \end{Bmatrix} \tag{3.76}$$

Since a considerable number of $\boldsymbol{TR}_{nc}$ may have already eliminated during the previous procedures $\mathcal{K}_{RS}$, the $\boldsymbol{TR}_{nc}$ that still exist in the RS, might be much smaller than C, such that $nc = 1,2, \dots, \mathcal{K}_{RS}$, $\mathcal{K}_{RS} \le C$.

The step i. to step x. is the main procedure of generating an RHS map part of the CCM algorithm. It should be noticed that in some cases a tableau does not involve any slack value $\boldsymbol{x_s}$. In those cases, it is necessary to assess whether the basic solution $\boldsymbol{x_B}$ is identical in $\boldsymbol{TR}_{nc}$ to another one without accounting for the slack values $\boldsymbol{x_s}$ part.

3.2.5 Sensitivity Analysis Based on CCM Algorithm

Two levels of tableaus are considered, main tableaus and sub-tableaus.

Main-tableaus, are referred to as $\boldsymbol{MR}_{mc}$ where $mc = 1,2, \dots, \mathcal{K}_{MC}$, $\mathcal{K}_{MC}$ is the amount of the main tableaus that are active in RS. A main tableau is a one for which the basic and non-basic variables of the non-slack value part $\boldsymbol{x}$ remain the same, however the $\boldsymbol{x_s}$ is not.

The second level is the sub-tableaus level. These *sub-tableaus* are the tableaus that are discussed in the previous sections and referred as $\boldsymbol{TR_{nc}}$. Thus, each $\boldsymbol{MR_{mc}}$ may contains several $\boldsymbol{TR_{nc}}$ regions.

The sensitivity analysis procedure of CCM algorithm is straightforward based on the RHS map as follows:

i. Compute the uncertainty range of $\boldsymbol{b}$, such that:

$$\Delta \boldsymbol{b} = \begin{bmatrix} \Delta \boldsymbol{b_1} \\ \Delta \boldsymbol{b_2} \\ \vdots \\ \Delta \boldsymbol{b_m} \end{bmatrix} = [\boldsymbol{b_{min}} \quad \boldsymbol{b_{max}}] = \begin{bmatrix} b_1^{min} & b_1^{max} \\ b_2^{min} & b_2^{max} \\ \vdots & \vdots \\ b_m^{min} & b_m^{max} \end{bmatrix} \tag{3.77}$$

$$con(\Delta \boldsymbol{b}) = \begin{bmatrix} \boldsymbol{b_{min}} - \boldsymbol{b} \\ \boldsymbol{b} - \boldsymbol{b_{max}} \end{bmatrix} \leq \boldsymbol{0} \tag{3.78}$$

The expressions in the RHS in the current thesis are assumed to be linear.

ii. Choose one sub-tableau region $\boldsymbol{TR_{nc}}$ from RS, calculate the region $TR_{nc}^{\mathcal{K}}\big|_{\Delta \boldsymbol{b}}$ as well as the vertices $VR_{nc}^{\mathcal{K}}\big|_{\Delta \boldsymbol{b}}$ of the convex cones $TR_{nc}^{\mathcal{K}}$ in $\boldsymbol{TR_{nc}}$ that may satisfy constraints from $\Delta \boldsymbol{b}$ (3.78) by using *lcon2vert* algorithm ($c2v$) in MATLAB. The result $TR_{nc}^{\mathcal{K}}\big|_{\Delta \boldsymbol{b}}$ is shown as follows:

$$VR_{nc}^{\mathcal{K}}\big|_{\Delta \boldsymbol{b}} = c2v\left(TR_{nc}^{\mathcal{K}}\big|_{\Delta \boldsymbol{b}}\right) = c2v\left(\begin{bmatrix} con(TR_{nc}^{\mathcal{K}}) \\ con(\Delta \boldsymbol{b}) \end{bmatrix} \leq \boldsymbol{0}\right) \tag{3.79}$$

After computing all of the convex cone regions that contained in $\boldsymbol{TR_{nc}}|_{\Delta \boldsymbol{b}}$, and the vertices set $\boldsymbol{VR_{nc}}|_{\Delta \boldsymbol{b}}$ also have been obtained as:

$$\boldsymbol{TR_{nc}}|_{\Delta \boldsymbol{b}} = \left\{TR_{nc}^{1}|_{\Delta \boldsymbol{b}}, TR_{nc}^{2}|_{\Delta \boldsymbol{b}}, \dots, TR_{nc}^{\mathcal{K}_{nc}}\big|_{\Delta \boldsymbol{b}}\right\} \tag{3.80}$$

$$\boldsymbol{VR_{nc}}|_{\Delta \boldsymbol{b}} = \left\{VR_{nc}^{1}|_{\Delta \boldsymbol{b}}, VR_{nc}^{2}|_{\Delta \boldsymbol{b}}, \dots, VR_{nc}^{\mathcal{K}_{nc}}\big|_{\Delta \boldsymbol{b}}\right\} \tag{3.81}$$

iii. A series of $\boldsymbol{x}_{nc}$ will be calculated by substituting $\boldsymbol{VR_{nc}}|_{\Delta \boldsymbol{b}}$ into (3.69), similarly a series of Z_{nc} will be obtained by substituting $\boldsymbol{VR_{nc}}|_{\Delta \boldsymbol{b}}$ into (3.38). Select the maximum and minimum values from$\boldsymbol{x}_{nc}$ and Z_{nc}, resulting in sensitivity ranges of $\boldsymbol{TR_{nc}}$ with uncertainty range of $\Delta \boldsymbol{b}$ to be:

$$\boldsymbol{SR_{nc}}|_{\Delta\boldsymbol{b}} = \begin{bmatrix} \Delta Z_{nc} \\ \Delta \boldsymbol{x}_{nc} \end{bmatrix}_{\Delta\boldsymbol{b}} = \begin{bmatrix} Z_{nc}^{min} & Z_{nc}^{max} \\ \boldsymbol{x}_{nc}^{min} & \boldsymbol{x}_{nc}^{max} \end{bmatrix}_{\Delta\boldsymbol{b}} = \begin{bmatrix} Z_{nc}^{min} & Z_{nc}^{max} \\ x_{nc,1}^{min} & x_{nc,1}^{max} \\ \vdots & \vdots \\ x_{nc,m}^{min} & x_{nc,m}^{max} \end{bmatrix}_{\Delta\boldsymbol{b}} \tag{3.82}$$

iv. Calculate the volume of $\boldsymbol{TR_{nc}}|_{\Delta\boldsymbol{b}}$ by substituting all vertices of $\boldsymbol{VR_{nc}}|_{\Delta\boldsymbol{b}}$ into *convhulln* in MATLAB. Then the volume is:

$$Vol_{nc}|_{\Delta\boldsymbol{b}} = convhulln(\boldsymbol{VR_{nc}}|_{\Delta\boldsymbol{b}}) \tag{3.83}$$

v. Chose another sub-tableau region $\boldsymbol{TR_{nc}}$ from RS and start repeat again from step ii. to step iv. Then the region RS with respect of $\Delta\boldsymbol{b}$, i.e. $RS|_{\Delta\boldsymbol{b}}$, the overall sensitivity range of $RS|_{\Delta\boldsymbol{b}}$, $SRS|_{\Delta\boldsymbol{b}}$, as well as the volumes of every sub-tableau in this region $Vol|_{\Delta\boldsymbol{b}}$ are given as follows:

$$RS|_{\Delta\boldsymbol{b}} = \left\{\boldsymbol{TR_1}|_{\Delta\boldsymbol{b}}, \boldsymbol{TR_2}|_{\Delta\boldsymbol{b}}, \dots, \boldsymbol{TR}_{\mathcal{K}_{RS}}\big|_{\Delta\boldsymbol{b}}\right\} \tag{3.84}$$

$$SRS|_{\Delta\boldsymbol{b}} = \begin{bmatrix} \Delta Z \\ \Delta X \end{bmatrix}_{\Delta\boldsymbol{b}} = \begin{bmatrix} Z^{min} & Z^{max} \\ X^{min} & X^{max} \end{bmatrix}_{\Delta\boldsymbol{b}} = \begin{bmatrix} Z^{min} & Z^{max} \\ X_1^{min} & X_1^{max} \\ \vdots & \vdots \\ X_m^{min} & X_m^{max} \end{bmatrix}_{\Delta\boldsymbol{b}} = \begin{Bmatrix} \boldsymbol{SR_1}, \\ \boldsymbol{SR_2}, \\ \dots, \\ \boldsymbol{SR}_{\mathcal{K}_{RS}} \end{Bmatrix}_{\Delta\boldsymbol{b}} \tag{3.85}$$

$$\begin{aligned} Vol|_{\Delta\boldsymbol{b}} &= sum\left(\{Vol_1|_{\Delta\boldsymbol{b}}, Vol_2|_{\Delta\boldsymbol{b}}, \dots, Vol_{\mathcal{K}_{RS}}\big|_{\Delta\boldsymbol{b}}\}\right) = \prod_{i=1}^{m} \Delta b_i \\ &= \prod_{i=1}^{m} \left(b_i^{max} - b_i^{min}\right) \end{aligned} \tag{3.86}$$

It is worth to note that the $\boldsymbol{TR_{nc}}|_{\Delta\boldsymbol{b}}$ might be non-feasible solution in the region of $\Delta\boldsymbol{b}$ and such regions can be eliminated from $RS|_{\Delta\boldsymbol{b}}$ during this process. Additionally, the vertices $VMR_{mc}|_{\Delta\boldsymbol{b}}$, sensitivity range $SMR_{mc}|_{\Delta\boldsymbol{b}}$ and volume $VolM_{mc}|_{\Delta\boldsymbol{b}}$of main tableau region $\boldsymbol{MR_{mc}}$ with respect of $\Delta\boldsymbol{b}$ also can be obtained by using the same method as in (3.84) to (3.86).

Another conclusion can be derivate from this section is that: In RHS space for same parameter $\bar{b}_i$ if $\bar{b}_{i,s} \ll \bar{b}_{i,g}$, where $\bar{b}_{i,s}$ and $\bar{b}_{i,g}$ are two nominal conditions of a same tableau, then for the same amount of change, $b_{i,s}$ will be more sensitive than $b_{i,g}$, however, if the ratio of change is same, then they are same sensitive. A more general description of this conclusion is: for an explicit given uncertainty range of RHS $RS|_{\Delta\boldsymbol{b}}$, if the nominal condition of $\bar{\boldsymbol{b}}_{\boldsymbol{s}}$ is much closer to original point than the nominal condition $\bar{\boldsymbol{b}}_{\boldsymbol{g}}$. Then, in most case, the number of tableaus in the uncertainty region of $\bar{\boldsymbol{b}}_{\boldsymbol{s}}$ is larger than which in the uncertainty region of $\bar{\boldsymbol{b}}_{\boldsymbol{g}}$. Since the conic cones' distribution of the sub-tableaus with respect to finite edges that are more compact in the region that close to original point. What can be found from this conclusion is that if some of the nominal parameters in RHS space is big enough, then they are negligible for some sensitivity analysis. And if they are too small, they might be very sensitive and generally hard to be investigated. This conclusion is useful when manipulating CCM algorithm in higher dimensional RHS space, since it provides a method to reduce the computation demanding by eliminate the negligible dimensions of the RHS space.

3.3 Case Study

Generally, in CCM algorithm, the RHS map are obtained from offline calculation, and the sensitivity analysis is used for generating the online uncertainty range. Therefore, the case study will also be based on this sequence.

3.3.1 2D RHS Map Generator

The example is the same as shown (3.60) and (3.61) in section 3.2.3. The example involves two inequalities and thus the RHS space is two-dimensional with respect to variables b_1 and b_2. The standard form of this LP problem is as following:

$$min\ Z = [-\boldsymbol{c} \quad \boldsymbol{0}]\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{x_s}\end{bmatrix}$$

s. t.

$$\begin{gathered}[\boldsymbol{A} \quad \boldsymbol{I}]\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{x_s}\end{bmatrix} = \boldsymbol{b}\\ \boldsymbol{A} = \begin{bmatrix}1 & 2\\ 3 & 2\end{bmatrix}, \qquad \boldsymbol{I} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \qquad \boldsymbol{c} = [3 \quad 5],\\ \boldsymbol{b} = \begin{bmatrix}b_1\\ b_2\end{bmatrix} \geq 0, \qquad \boldsymbol{x} = \begin{bmatrix}x_1\\ x_2\end{bmatrix} \geq 0, \qquad \boldsymbol{x_s} = \begin{bmatrix}x_3\\ x_4\end{bmatrix} \geq 0\end{gathered} \tag{3.61}$$

Step i: As discussed in section 3.2.4 of the CCM method is to compute the set $\boldsymbol{\mathcal{T}}$ by using *nchoosek* in MATLAB. In this problem, the amount of constraints $m = 2$, the amount of unknowns $n = 2$, $C = C(m, m + n) = 6$, which means that this problem have 6 possible combinations of X_{nc}^{p} with respect of each tableau T_{nc}. For instance, $T_1 = [1 \quad 2]$ means that x_1 and x_2 are both basic variables for this tableau. Thus, using this notation, the entire set $\boldsymbol{\mathcal{T}}$ is as following:

$$\boldsymbol{\mathcal{T}} = \begin{Bmatrix}T_1, T_2, T_3\\ T_4, T_5, T_6\end{Bmatrix} = \begin{Bmatrix}[1 \quad 2], [1 \quad 3], [1 \quad 4]\\ [2 \quad 3], [2 \quad 4], [3 \quad 4]\end{Bmatrix} \tag{3.87}$$

Step ii: After obtaining the set $\boldsymbol{\mathcal{T}}$, the parameters of each objective tableaus T_{nc}: $\boldsymbol{x}_{nc}^{p}$, $\boldsymbol{A}_{nc}^{p}$ and $\boldsymbol{c}_{nc}^{p}$ are calculated. T_1 is used as an example to illustrate the following several steps, and these steps are also applied on the other tableaus. Following (3.88), the $\boldsymbol{x}_1^p$, $\boldsymbol{A}_1^p$ and $\boldsymbol{c}_1^p$ corresponding to T_1 are:

$$\boldsymbol{x}_1^p = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \boldsymbol{A}_1^p = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}, \quad \boldsymbol{c}_1^p = -[3 \quad 5] \tag{3.88}$$

In T_1 each basic variable is calculated as follows:

$$\begin{aligned} x_1 &= \frac{\det(\boldsymbol{A}_{1,1}^{p})}{\det(\boldsymbol{A}_1^p)} = \frac{\det\left(\begin{bmatrix} b_1 & 2 \\ b_2 & 2 \end{bmatrix}\right)}{\det\left(\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}\right)} = \frac{1}{2}b_2 - \frac{1}{2}b_1 \\ x_2 &= \frac{\det(\boldsymbol{A}_{1,2}^{p})}{\det(\boldsymbol{A}_1^p)} = \frac{\det\left(\begin{bmatrix} 1 & b_1 \\ 3 & b_2 \end{bmatrix}\right)}{\det\left(\begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}\right)} = \frac{3}{4}b_1 - \frac{1}{4}b_2 \end{aligned} \tag{3.89}$$

Step iii: $\boldsymbol{\alpha_1}$ can be represented as:

$$\boldsymbol{\alpha_1} = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} = \begin{bmatrix} \frac{\partial X_1}{\partial b_1} & \frac{\partial X_1}{\partial b_2} \\ \frac{\partial X_2}{\partial b_1} & \frac{\partial X_2}{\partial b_2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} \end{bmatrix} \tag{3.90}$$

Since, $x_i \geq 0$ and $Z_{nc}^p = \boldsymbol{c}_{nc}^p \boldsymbol{\alpha_{nc}} \boldsymbol{b}$, applying (3.37) and (3.38), the initial edge $con(TR_1^{*1}) = -\boldsymbol{\alpha_1 b}$ as well as the cost function Z_1^p of T_1 are:

$$\begin{aligned} -\boldsymbol{\alpha_1 b} &= -\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \leq \boldsymbol{0} \\ Z_1^p &= \boldsymbol{c}_1^p \boldsymbol{\alpha_1 b} = [-3 \quad -5] \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = -\frac{9}{4}b_1 - \frac{1}{4}b_2 \end{aligned} \tag{3.91}$$

Step iv – v: The (3.91) is an initial property of objective tableau T_1 as well as an initial property of tableau region $\boldsymbol{TR_1^*}$, which is defined by the initial edges as well as the constrains of $\boldsymbol{b} \geq \boldsymbol{0}$, such that:

$$\frac{1}{2}b_1 - \frac{1}{2}b_2 \leq 0 \tag{3.92}$$

$$-b_1 + \frac{1}{3}b_2 \leq 0$$

$$b_1 \geq 0, \qquad b_2 \geq 0$$

Fig. 3.7 illustrates the initial region of $\boldsymbol{TR_1^*}$, which is enclosed in the red lines as well as a property of objective tableau T_1 with the additional constraints $\boldsymbol{f_{ac}}$ as in (3.93).

Step vi: The additional constraints $\boldsymbol{f_{ac}}$ in this example are:

$$\boldsymbol{f_{ac}} = \begin{bmatrix} -1^T\boldsymbol{b} + \Phi_{min} \\ b_1 - \Phi_{max} \\ b_2 - \Phi_{max} \end{bmatrix} \leq \mathbf{0}, \qquad \Phi_{min} = 10^{-6}, \qquad \Phi_{max} = 20 \tag{3.93}$$

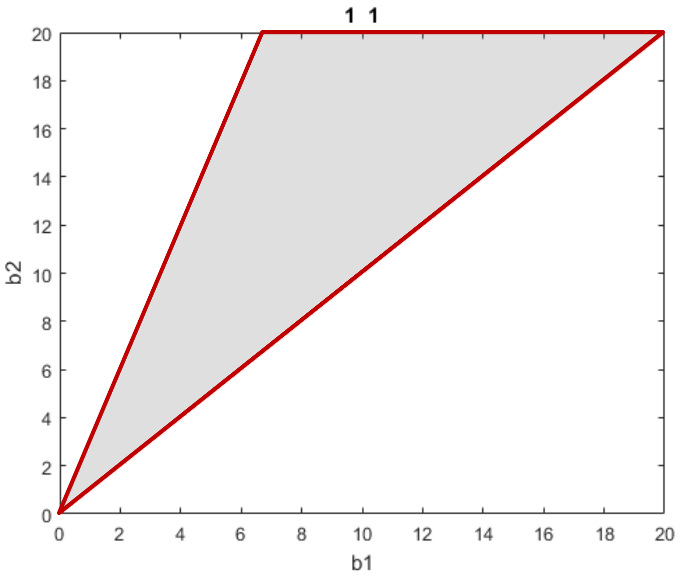


Figure 3.7 The initial region of $\boldsymbol{TR_1^*}$

Another tableau is introduced here, the tableau T_5, with the similar steps of generating $\boldsymbol{TR_1^*}$. The initial property for T_5 are:

$$\boldsymbol{x_5^p} = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} X_2 \\ X_4 \end{bmatrix}, \qquad \boldsymbol{A_5^p} = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix}, \qquad \boldsymbol{c_5^p} = -[5 \quad 0]$$

$$-\boldsymbol{\alpha_5 b} = -\begin{bmatrix} \frac{1}{2} & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \leq \mathbf{0} \tag{3.94}$$

$$Z_5^p = \boldsymbol{c_5^p \alpha_5 b} = [-5 \quad 0] \begin{bmatrix} \frac{1}{2} & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = -\frac{5}{2}b_1$$

Therefore, the initial regions of T_5, i.e. $\boldsymbol{TR_5^*}$, with constraints of $\boldsymbol{b} \geq \boldsymbol{0}$ and $\boldsymbol{f_{ac}}$ have been shown by the region enclosed within the red lines plane in Fig. 3.8. It is obvious that there is an overlap region of $\boldsymbol{TR_1^*}$ and $\boldsymbol{TR_5^*}$. Step v. in section 3.2.4 can be used to solve this overlap.

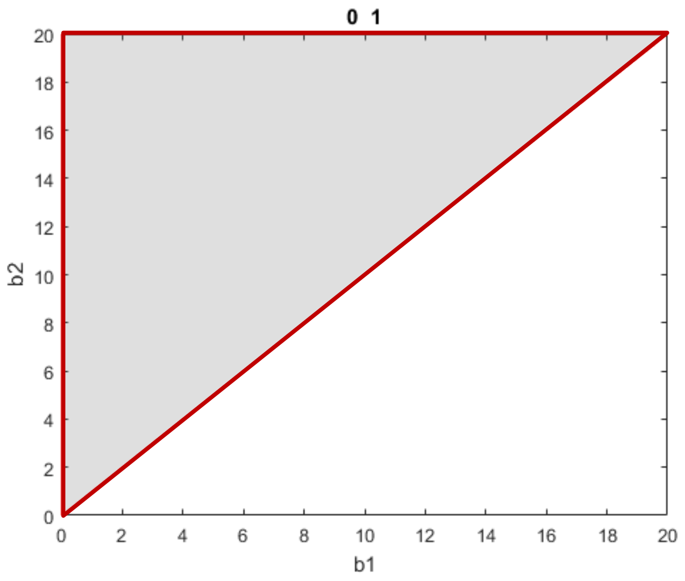


Figure 3.8 The initial region of $\boldsymbol{TR_5^*}$

Applying (3.70) for $\boldsymbol{TR_1^*}$ and $\boldsymbol{TR_5^*}$, since in the condition of this example, $\boldsymbol{TR_1^*} = \{TR_1^{*1}\}$, $\mathcal{K} = 1$, $km = 1$:

$$TR_1^{1,1} = \begin{bmatrix} con(TR_1^{*1}) \\ (\boldsymbol{c}_1^p\boldsymbol{\alpha_1} - \boldsymbol{c}_5^p\boldsymbol{\alpha}_5)\boldsymbol{b} \\ -\boldsymbol{\alpha}_5\boldsymbol{b} \end{bmatrix} = \begin{bmatrix} con(TR_1^{*1}) \\ \frac{1}{4}b_1 - \frac{1}{4}b_2 \\ -\boldsymbol{\alpha}_5\boldsymbol{b} \end{bmatrix} \leq 0$$

$$TR_1^{1,2} = \begin{bmatrix} con(TR_1^{*1}) \\ \boldsymbol{\rho}_1 \cdot (\boldsymbol{\alpha}_5\boldsymbol{b}) \end{bmatrix} = \begin{bmatrix} con(TR_1^{*1}) \\ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \end{bmatrix} \leq 0 \tag{3.95}$$

$$TR_1^{1,3} = \begin{bmatrix} con(TR_1^{*1}) \\ \boldsymbol{\rho}_2 \cdot (\boldsymbol{\alpha}_5\boldsymbol{b}) \end{bmatrix} = \begin{bmatrix} con(TR_1^{*1}) \\ \begin{bmatrix} -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \end{bmatrix} \leq 0$$

Thus $\boldsymbol{TR_1^*}$ has been divided into 3 convex cones: $\boldsymbol{TR_1^*} = \{TR_1^{1,1}, TR_1^{1,2}, TR_1^{1,3}\}$. Then these 3 convex cones are checked for feasibility and redundancy of constraints by using *noredund* algorithm from *lcon2vert* package in MATLAB (Matt, 2017) along with additional constrains $\boldsymbol{f_{ac}}$.

The result of this test is that only $TR_1^{1,1}$ has a feasible solution and $\frac{1}{4}b_1 - \frac{1}{4}b_2 \leq 0$ as well as $-\boldsymbol{\alpha_5 b} \leq \mathbf{0}$ are redundant constrains. After eliminating the constraints, $\boldsymbol{TR_1} = \{TR_1^{1,1}\}$**.** Similar procedure has to be applied between Tableau T_1 and the remaining tableaus in $\boldsymbol{\mathcal{T}}$. In the end of this iteration, the result of the tableau T_1 is still $TR_1^{1,1}$.

Similar procedure will be applied as done above for tableau T_1 has to be applied to the other tableaus. After checking all of the 6 tableaus, the final results are only 3 tableaus are active in the RHS map space (*RS*), they are: T_1, T_3 and T_4. The region of each tableau are $\boldsymbol{TR_1}$, $\boldsymbol{TR_3}$ and $\boldsymbol{TR_4}$, respectively. The tableau distribution of this RHS map is illustrated in Fig. 3.9.

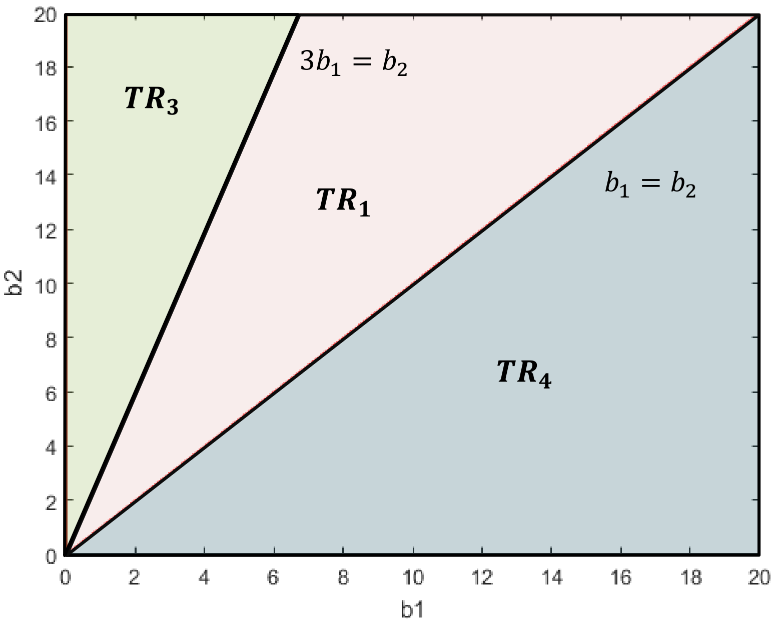


Figure 3.9 The tableau distribution and main edges of the mentioned RHS space

3.3.2 Sensitivity Analysis in 2D RHS Space

After obtaining the RHS map in Fig. 3.9, the properties of every active tableau region are calculated. For instance, the parameters of TR_1 are:

$$-\boldsymbol{\alpha_1 b} = -\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \leq \mathbf{0} \tag{3.96}$$

$$Z_1^p = \boldsymbol{c}_1^p \boldsymbol{\alpha_1} \boldsymbol{b} = [-3 \quad -5] \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = -\frac{9}{4} b_1 - \frac{1}{4} b_2$$

$$\boldsymbol{e_1} \triangleq \begin{Bmatrix} -3b_1 + b_2 \leq 0 \\ b_1 - b_2 \leq 0 \end{Bmatrix}$$

If the uncertainty ranges of the RHS are $\Delta\boldsymbol{b}$, and the nominal condition of RHS is $\bar{\boldsymbol{b}}$, then the constraints of the uncertainty region can be generated by using (3.77) and (3.78). For instance, for particular numerical values of $\bar{\boldsymbol{b}}$ and $\Delta\boldsymbol{b}$, the $con(\Delta\boldsymbol{b})$ is represented as:

$$\bar{\boldsymbol{b}} + \Delta\boldsymbol{b} = \begin{bmatrix} 6 \\ 8 \end{bmatrix} + \begin{bmatrix} -4 & 2 \\ -6 & 6 \end{bmatrix} = [\boldsymbol{b_{min}} \quad \boldsymbol{b_{max}}] = \begin{bmatrix} 2 & 8 \\ 2 & 14 \end{bmatrix}$$

$$con(\Delta\boldsymbol{b}) = \begin{bmatrix} \boldsymbol{b_{min}} - \boldsymbol{b} \\ \boldsymbol{b} - \boldsymbol{b_{max}} \end{bmatrix} = \begin{bmatrix} 2 - b_1 \\ 2 - b_2 \\ b_1 - 8 \\ b_2 - 14 \end{bmatrix} \leq \boldsymbol{0} \tag{3.97}$$

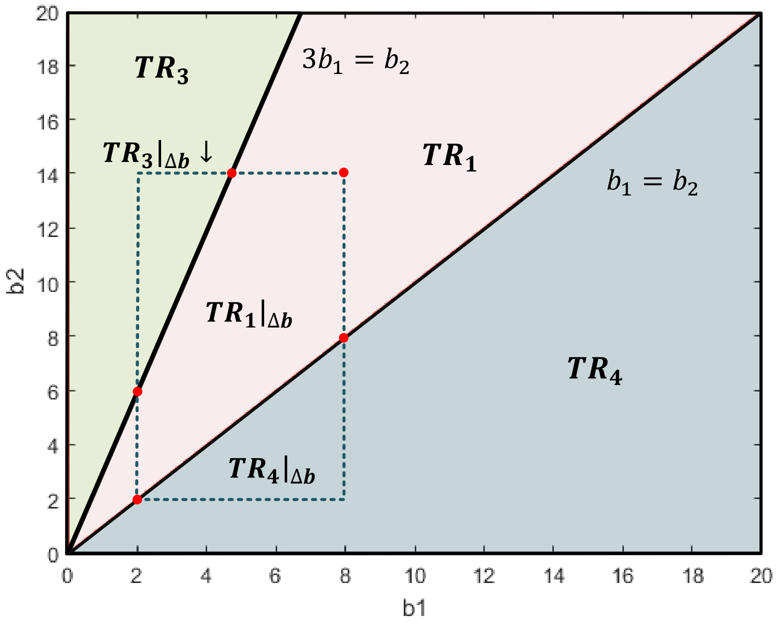


Figure 3.10 The region of uncertainty in RHS space

The region described by $con(\Delta\boldsymbol{b})$ is illustrated in Fig. 3.10. It is obvious that the uncertainty region, i.e. $RS|_{\Delta\boldsymbol{b}}$, enclosed within the dotted lines in the figure, can be divided into 3 regions corresponding to the 3 different tableaus. For example, $TR_1|_{\Delta b}$ is used to describe the procedure of sensitivity analysis in the CCM algorithm.

The vertices of $TR_1|_{\Delta b}$ can be computed by using *lcon2vert* in MATLAB (Matt, 2017) following (3.79) such that:

$$VR_1|_{\Delta \boldsymbol{b}} = c2v(TR_1|_{\Delta b}) = c2v\left(\begin{bmatrix} con(TR_1|_{\Delta b}) \\ con(\Delta \boldsymbol{b}) \end{bmatrix} \leq \boldsymbol{0}\right) = c2v\left(\begin{bmatrix} \boldsymbol{e_1} \\ con(\Delta \boldsymbol{b}) \end{bmatrix} \leq \boldsymbol{0}\right) \quad (3.98)$$

The resulting vertices, shown as red dots in Fig. 3.10, are:

$VR_1|_{\Delta \boldsymbol{b}} = \{(4.67,14)\ (8,14)\ (8,8)\ (2,2)\ (2,6)\}$

Then by applying (3.82) with $VR_1|_{\Delta \boldsymbol{b}}$, the maximum and minimum values of each parameter can be obtained. It should be noticed that if the dimensions of RHS space are too large, there may be too many vertices that need to be calculated. In that case, in order to decrease the calculation effort, the function *linprog* in MATLAB can be applied to (3.98) to reduce the computation load. For each point in $VR_1|_{\Delta \boldsymbol{b}}$ the results of for each variable are:

$$\begin{bmatrix} Z_1 \\ x_1 \\ x_2 \end{bmatrix}_{\Delta \boldsymbol{b}} = \begin{bmatrix} -14 & -21.5 & -20 & -5 & -6 \\ 4.67 & 3 & 0 & 0 & 2 \\ 0 & 2.5 & 4 & 1 & 0 \end{bmatrix} \quad (3.99)$$

After selecting the maximum and minimum values of each variable in (3.99), the corresponding sensitivity range of $\boldsymbol{TR_{nc}}$ under uncertainty of $\Delta \boldsymbol{b}$, i.e. $\boldsymbol{SR_1}|_{\Delta \boldsymbol{b}}$, can be summarized as follows:

$$\boldsymbol{SR_1}|_{\Delta \boldsymbol{b}} = \begin{bmatrix} \Delta Z_1 \\ \Delta \boldsymbol{x}_1 \end{bmatrix}_{\Delta \boldsymbol{b}} = \begin{bmatrix} -Z_1^{min} & -Z_1^{max} \\ x_{1,1}^{min} & x_{1,1}^{max} \\ x_{1,2}^{min} & x_{1,2}^{max} \end{bmatrix}_{\Delta \boldsymbol{b}} = \begin{bmatrix} 5 & 21.5 \\ 0 & 4.67 \\ 0 & 4 \end{bmatrix}_{\Delta \boldsymbol{b}}$$

The volume of $TR_1|_{\Delta b}$ can be obtained by using *convhulln* in MATLAB as shown in (3.83):

$Vol_1|_{\Delta \boldsymbol{b}} = convhulln(VR_1|_{\Delta \boldsymbol{b}}) = 43.32$

Following the same approach, the sensitivity range of the other 2 tableaus can also be computed by using the previous methods. Since the whole volume of uncertainty region $RS|_{\Delta \boldsymbol{b}}$ is:

$$Vol|_{\Delta \boldsymbol{b}} = sum(\{Vol_1|_{\Delta \boldsymbol{b}}, Vol_3|_{\Delta \boldsymbol{b}}, Vol_4|_{\Delta \boldsymbol{b}}\}) = \prod_{i=1}^{2} \Delta b_i = 72$$

Therefore, if the probability distribution of occurrence of parameters RHS are assumed to be uniform, then the possibility of $TR_1|_{\Delta b}$ region to be active under the uncertainty range of $\Delta \boldsymbol{b}$ is:

$$P(TR_1|_{\Delta b}) = \frac{Vol_1|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = \frac{43.32}{72} = 60.17\%$$

The possibility of the other 2 tableaus also can obtained along with this method and are:

$$P(TR_3|_{\Delta b}) = \frac{Vol_3|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = \frac{10.68}{72} = 14.83\%$$

$$P(TR_4|_{\Delta b}) = \frac{Vol_4|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = \frac{18}{72} = 25\%$$

The mathematical complexity of the approach obviously increases with the dimensions as shown for a 3D case study in Chapter 4.

3.4 Conclusions

In this chapter, two algorithms referred to as the 100 Percent Rule Based Method and the Convex Cone Method (CCM) were proposed in order to propagate uncertainty into the solutions of an LP problem. It was shown that the 100 Percent Rule Based Method can provide a necessary but not sufficient polyhedron region based bounds around a nominal point but it cannot generate the entire region of uncertainty. To address this problem, the CCM algorithm is proposed that it is able to generate the entire uncertainty regions. A series of lemmas and theorems were proved and served as the basis of the proposed CCM algorithm.

There are two main layers in the CCM algorithm, as shown in Fig. 3.11. The first layer is focusing on generating the RHS map. This layer is highly computationally demanding but it has to be performed only once for any values of the RHS of the inequalities of the LP problem. In this layer, after generating every possible combination of solutions, the feasible region of every possible tableau can be obtained. Then these tableaus will be divided and bounded by comparing with the intersection tableaus that have an overlap region with each other. The final feasible region of each tableau can be computed by using a series of overlap region allocation steps and by dividing the remaining polyhedron part of this tableau into many smaller convex cones. The final RHS map is then generated by gathering together all of these active tableaus. The second layer of this uncertainty propagation algorithm involves an on-line sensitivity analysis. By using the RHS map that is computed in the previous layer and based on the uncertainty region, which is assumed in this layer, the sensitivity ranges of each variable of the LP problem with respect to each of the tableaus can be calculated. Though this layer needs to be executed whenever the uncertainty bounds of the RHS changes, the computational effort is not as high as for the first layer. This fact makes the CCM an attractive approach for uncertainty propagation in an LP problem.

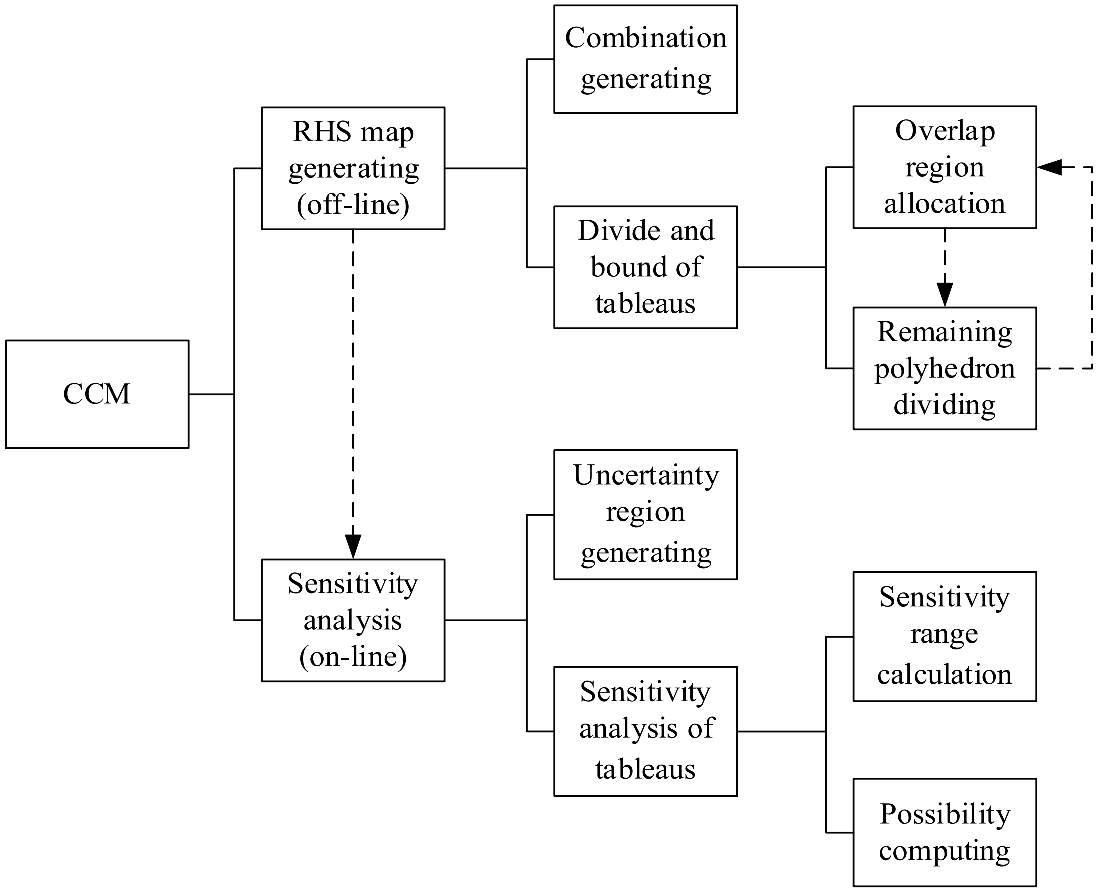

CCM
RHS map generating (off-line)
Sensitivity analysis (on-line)
Combination generating
Divide and bound of tableaus
Overlap region allocation
Remaining polyhedron dividing
Uncertainty region generating
Sensitivity analysis of tableaus
Sensitivity range calculation
Possibility computing

Figure 3.11 The basic structure of the CCM algorithm

Chapter 4

Robust Nonlinear MPC based on Convex Cone Method and Its Applications in Control of Bioreactors Based on Dynamic Metabolic Flux Balance Models

In this chapter, the controller algorithm based on the Convex Cone Method (CCM) proposed in Chapter 3 is applied to a bioreactor culture described by a Dynamic Flux Balance Model (DFBM). As shown in Chapter 3 the proposed CCM algorithm is significantly more efficient for generating the uncertainty regions, each corresponding to a different active set of basic solutions, as compared with the 100 Percent Rule Based Method. Thus, the CCM algorithm is adopted in this chapter for propagating the uncertainty along with the prediction horizon of the controller. The tree structure framework (Lucia et al., 2013) is applied in this work for implementing a robust Economic Model Predictive Control (EMPC) where the branches of the tree correspond to different uncertainty regions identified by the CCM method. It will be shown with simple calculations that the resulting EMPC controller is computationally more efficient for estimating control actions as compared to the currently used uncertainty propagation methods, such as Monte Carlo sampling (Kawohl et al., 2007) and Polynomial Chaos Expansions (PCE) (Ghanem & Spanos, 1990), in the presence of a large amount of model uncertainty parameters, e.g. more than 10 parameters.

The dynamic metabolic flux model (DMFM) that is used as internal models in the proposed EMPC algorithm of this Chapter is formulated as Linear Programming (LP) optimization. As discussed in Chapter 2, both academia and industrial practitioners are increasingly adopting DMFM to model numerous biotechnological processes models; hence, the DMFM based controller proposed in the current chapter is expected to have wide applicability. Moreover, although the presented application is specifically targeted to dynamic metabolic flux models, the controller with the novel uncertainty propagation method proposed in this chapter is applicable for not only biotechnological processes but also for other processes that may be modeled as LP's in the future.

This chapter is organized as follows. Section 4.1 proposes a preliminary illustration of the application procedure for CCM algorithm using a 3D case study method. The discussion of the prediction model development based on CCM and a tree structure algorithm as well as the propagation of the uncertainty onto the outputs' predictions based on robust NMPC are presented in section 4.2. A comparison of this approach with other algorithms is also discussed in this section. In section 4.3, the algorithm described in section 4.2 will be used to develop the robust-model predictive controller based on DFBM; this controller is then used to formulate a robust bioreactor control framework. The results from such implementation are presented in section 4.4. Concluding remarks are presented at the end.

4.1 Illustrative Case Study: CCM Algorithm

4.1.1 Introduction

In this section, the CCM algorithm is illustrated for a 3D case study to facilitate visualization of the method. This application reveals that when the size of the uncertain inequality constraints grows, which is generally referred as the growing of the dimension of the RHS space in this section, the number of convex cones as well as the computational complexity of the RHS space may tend to increase exponentially.

There are two main layers in the CCM algorithm, as shown in Fig. 3.11. The first layer focus on generating the RHS map. This layer is computationally demanding but it has to be performed only once for any values of the RHS of the inequalities of the LP problem. Thus, for the case of a controller application it can be executed off-line. In this layer, after generating every possible combination of solutions, the feasible region for every possible tableau can be obtained. Note that each tableau corresponds to a particular set of basic solutions. When overlap between tableaus occurs, they will be divided and bounded by comparing with the intersection tableaus that have an overlap region with each other. The final feasible region of each tableau can be computed by using a series of overlap region allocation steps and by ultimately dividing the remaining polyhedron part of this tableau into many smaller convex cones. The final RHS map is then generated by gathering together all the active tableaus.

The second layer of this uncertainty propagation algorithm involves an on-line sensitivity analysis calculation. By using the RHS map that is computed in the previous layer and based on the uncertainty region which is defined *a priori* in this layer, the sensitivity ranges of each variable of the LP problem with respect to each of the tableaus can be calculated. However, this layer needs to be executed only when the uncertainty bounds of the RHS changes; hence, the computational effort for the execution of this second layer is not as high as for the first layer. This fact makes the CCM an attractive approach for uncertainty propagation.

It will be shown that the computational complexity increases considerably in the 3D case as compared to the 2D case presented in Chapter 3, but the algorithm is still computationally attractive as compared to the Monte Carlo sampling.

4.1.2 A 3D Case Study

A 3 dimensional example is presented here to illustrate the application of the CCM method proposed in Chapter 3. The objective is to show how the methodology scales with an increase in the number of dimensions, i.e. the number of uncertain inequalities considered in the example. In particular, a comparison between the current 3D case with the 2D case presented in Chapter 3 will be conducted to evaluate the complexity of the proposed approach.

The LP problem under consideration is defined as follows:

$$min\ Z = \begin{bmatrix} -\boldsymbol{c} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix}$$

s. t.

$$\begin{gathered} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_s} \end{bmatrix} = \boldsymbol{b} \\ \boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 4 & 1 \end{bmatrix}, \quad \boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 2 & 1 \end{bmatrix}, \\ \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \geq 0, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0, \quad \boldsymbol{x_s} = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq 0 \end{gathered} \tag{4.1}$$

The CCM algorithm proposed in Chapter 3 is applied here to calculate the 3D RHS map for this problem. The space is divided into 3 different main tableaus:

$$\mathcal{T} = \{\boldsymbol{MR_1}, \boldsymbol{MR_2}, \boldsymbol{MR_3}\} \tag{4.2}$$

where $\boldsymbol{MR_1}$ includes all tableaus where x_1 is basic and x_2 is non-basic; $\boldsymbol{MR_2}$ includes all tableaus where both x_1 and x_2 are basic; $\boldsymbol{MR_3}$ contains all tableaus where x_2 is basic and x_1 is non-basic. For example, as illustrated in Fig. 4.1, tableau $\boldsymbol{MR_1}$ contains 3 convex cones shown in different colors with 2 additional constrains $\boldsymbol{f_{ac}}$ where $\Phi_{min} = 10$ and $\Phi_{max} = 20$ as in function (3.71). The relationship among these tableaus can be represented as follows:

$$\boldsymbol{MR_1} = \{\boldsymbol{TR_5}, \boldsymbol{TR_6}\}$$
$$\boldsymbol{TR_5} = \{TR_5^1, TR_5^2\}, \qquad \boldsymbol{TR_6} = \{TR_6^1\} \tag{4.3}$$

where the convex cones in $\boldsymbol{TR_5}$ are in terms of $T_5 = [1 \quad 3 \quad 5]$ for which the corresponding basic variables are $\boldsymbol{x_B} = [x_1 \quad x_3 \quad x_5]^T$; the convex cones in $\boldsymbol{TR_6}$ are in terms of $T_6 = [1 \quad 4 \quad 5]$ with the corresponding basic variables $\boldsymbol{x_B} = [x_1 \quad x_4 \quad x_5]^T$. The polyhedron of TR_5^1, TR_5^2 and TR_6^1 are colored in Figure 4.1 in red, green and blue, respectively. It is evident from Fig. 4.1 that the polyhedron of $\boldsymbol{MR_1}$ is nonconvex. Other regions of $\boldsymbol{MR_2}$ and $\boldsymbol{MR_3}$ as well as their convex cones can also obtained by using CCM algorithm as illustrated in Fig. 4.2 where each main tableau is shown in different color.

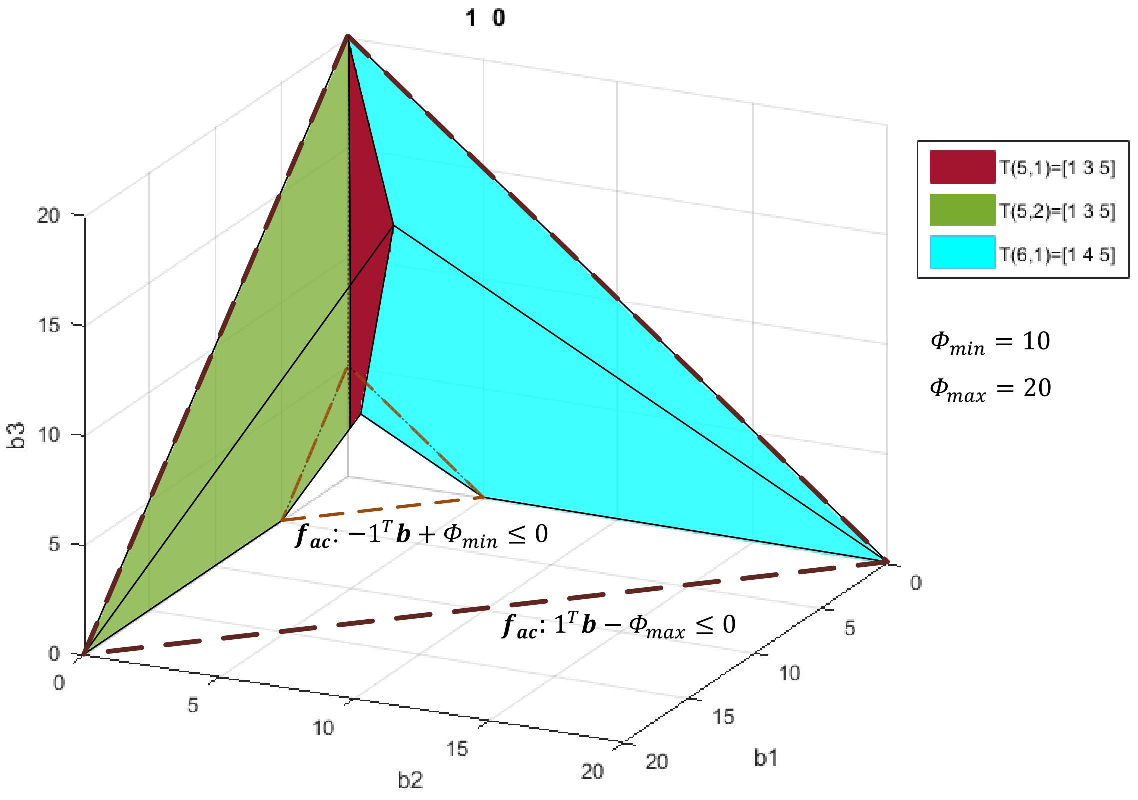


Figure 4.1 Tableaus distribution of $\boldsymbol{MR_1}$ in RHS map

Once the 3D RHS map has been identified, the sensitivity analysis also can be performed with the method introduced in section 3.3.2. The uncertainty range in this case is defined as follows:

$$\bar{\boldsymbol{b}} = \begin{bmatrix} 3 \\ 4 \\ 12 \end{bmatrix}, \quad \Delta\boldsymbol{b} = [\boldsymbol{b_{min}} \quad \boldsymbol{b_{max}}] = [0.5 \quad 2] \cdot \bar{\boldsymbol{b}} = \begin{bmatrix} 1.5 & 6 \\ 2 & 8 \\ 6 & 24 \end{bmatrix}$$

Thus, the constraints describing the uncertain ranges of all parameters $con(\Delta \boldsymbol{b})$ can be represented as follows:

$$con(\Delta \boldsymbol{b}) = \begin{bmatrix} \boldsymbol{b}_{\boldsymbol{min}} - \boldsymbol{b} \\ \boldsymbol{b} - \boldsymbol{b}_{\boldsymbol{max}} \end{bmatrix} = \begin{bmatrix} 1.5 - b_1 \\ 2 - b_2 \\ 2 - b_3 \\ b_1 - 6 \\ b_2 - 8 \\ b_3 - 24 \end{bmatrix} \leq \boldsymbol{0}$$

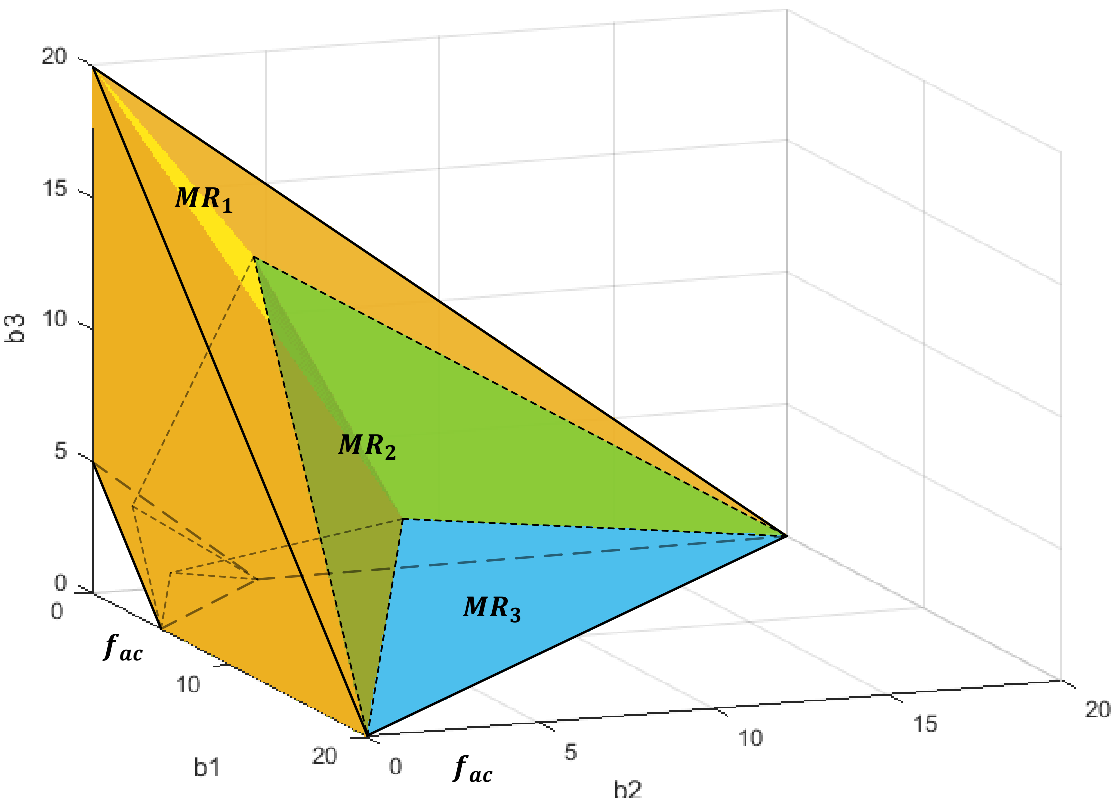


Figure 4.2 Tableaus distribution of MR_1, MR_2 and MR_3 in RHS map

After intersecting the regions of $con(\Delta \boldsymbol{b})$ with the main tableaus calculated in the RHS map by eliminating the redundant constraints, the sensitivity region for each of the uncertain model parameters for each main tableau can be obtained. After this procedure, it was found that 2 main tableaus, $\boldsymbol{MR_1}$ and $\boldsymbol{MR_2}$, have active tableaus in the uncertainty region. For instance, the region $\boldsymbol{MR_1}$ maintains 2 different tableaus $\{\boldsymbol{TR_5}, \boldsymbol{TR_6}\}$, i.e. 3 polyhedrons $\{TR_5^1, TR_5^2, TR_6^1\}$, in this region which are indicated as bounded by the dash-dotted line in Fig. 4.3. The sensitivity region of $\boldsymbol{MR_1}$ ($\boldsymbol{SMR_1}$) can be obtained by analyzing the sensitivity region of each of the polyhedrons and select the maximum and minimum values of each parameter within each polyhedron.

Then, the cost function values, referred to as sensitivity ranges, for each polyhedron are as follows:

$Z_5^1|_{\Delta \boldsymbol{b}} = [4 \quad 12]_{\Delta \boldsymbol{b}}, \qquad Z_5^2|_{\Delta \boldsymbol{b}} = [4 \quad 6]_{\Delta \boldsymbol{b}}, \qquad Z_6^1|_{\Delta \boldsymbol{b}} = [3 \quad 12]_{\Delta \boldsymbol{b}}$

accordingly, the sensitivity ranges of the cost function in $\boldsymbol{SMR_1}$ is as follows:

$Z_{\boldsymbol{SMR_1}}\big|_{\Delta \boldsymbol{b}} = [\min\{Z_5^1|_{\Delta \boldsymbol{b}}, Z_5^2|_{\Delta \boldsymbol{b}}, Z_6^1|_{\Delta \boldsymbol{b}}\} \ \max\{Z_5^1|_{\Delta \boldsymbol{b}}, Z_5^2|_{\Delta \boldsymbol{b}}, Z_6^1|_{\Delta \boldsymbol{b}}\}] = [3 \quad 12]_{\Delta \boldsymbol{b}}$

The volume of each polyhedron in $\boldsymbol{MR_1}$ are:

$Vol_5^1|_{\Delta \boldsymbol{b}} = 70.67, \qquad Vol_5^2|_{\Delta \boldsymbol{b}} = 14.67, \qquad Vol_6^1|_{\Delta \boldsymbol{b}} = 200.33$

hence, the complete volume of $RS|_{\Delta \boldsymbol{b}}$, which is the sum of each polyhedron in $\boldsymbol{MR_1}$, is as follows:

$$Vol|_{\Delta \boldsymbol{b}} = \prod_{i=1}^{3} \Delta b_i = 486$$

Thus, the probability of occurrence of each tableau in the uncertainty region under consideration can be determined as follows:

$$P(\boldsymbol{TR_5}|_{\Delta b}) = \frac{Vol_5^1|_{\Delta \boldsymbol{b}} + Vol_5^2|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = 17.56\%$$

$$P(\boldsymbol{TR_6}|_{\Delta b}) = \frac{Vol_6^1|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = 41.22\%$$

The possibility of $\boldsymbol{MR_1}$ can also be obtained as follows:

$P(\boldsymbol{MR_1}|_{\Delta b}) = P(\boldsymbol{TR_5}|_{\Delta b}) + P(\boldsymbol{TR_6}|_{\Delta b}) = 58.78\%$

The sensitivity range for another main tableau $\boldsymbol{MR_2}$ that is active in the overall uncertainty region also can be computed using the CCM algorithm. The results for $\boldsymbol{MR_2}$ are illustrated in Fig. 4.4 with different colors for the different polyhedrons. In this graph, the $\boldsymbol{MR_2}$ maintains 2 different tableaus $\{\boldsymbol{TR_1}, \boldsymbol{TR_2}\}$, i.e. 3 polyhedrons $\{TR_1^1, TR_1^2, TR_2^1\}$, in the uncertainty region shown in Fig. 4.4 as the regions bounded by the dash-dotted lines.The volume of each polyhedron composing $\boldsymbol{MR_2}$ are:

$Vol_1^1|_{\Delta \boldsymbol{b}} = 3.33, \qquad Vol_1^2|_{\Delta \boldsymbol{b}} = 20.73, \qquad Vol_2^1|_{\Delta \boldsymbol{b}} = 176.27$

Thus, the probability of occurrence of each tableau in the uncertainty region under consideration with the assumption that the probability distribution of this RHS space are uniform is as follows:

$$P(\boldsymbol{TR_1}|_{\Delta b}) = \frac{Vol_1^1|_{\Delta \boldsymbol{b}} + Vol_1^2|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = 4.95\%$$

$$P(\boldsymbol{TR_2}|_{\Delta b}) = \frac{Vol_2^1|_{\Delta \boldsymbol{b}}}{Vol|_{\Delta \boldsymbol{b}}} = 36.27\%$$

The probabilities for $\boldsymbol{MR_2}$ can also be obtained as follows:

$$P(\boldsymbol{MR_2}|_{\Delta b}) = P(\boldsymbol{TR_1}|_{\Delta b}) + P(\boldsymbol{TR_2}|_{\Delta b}) = 41.22\%$$

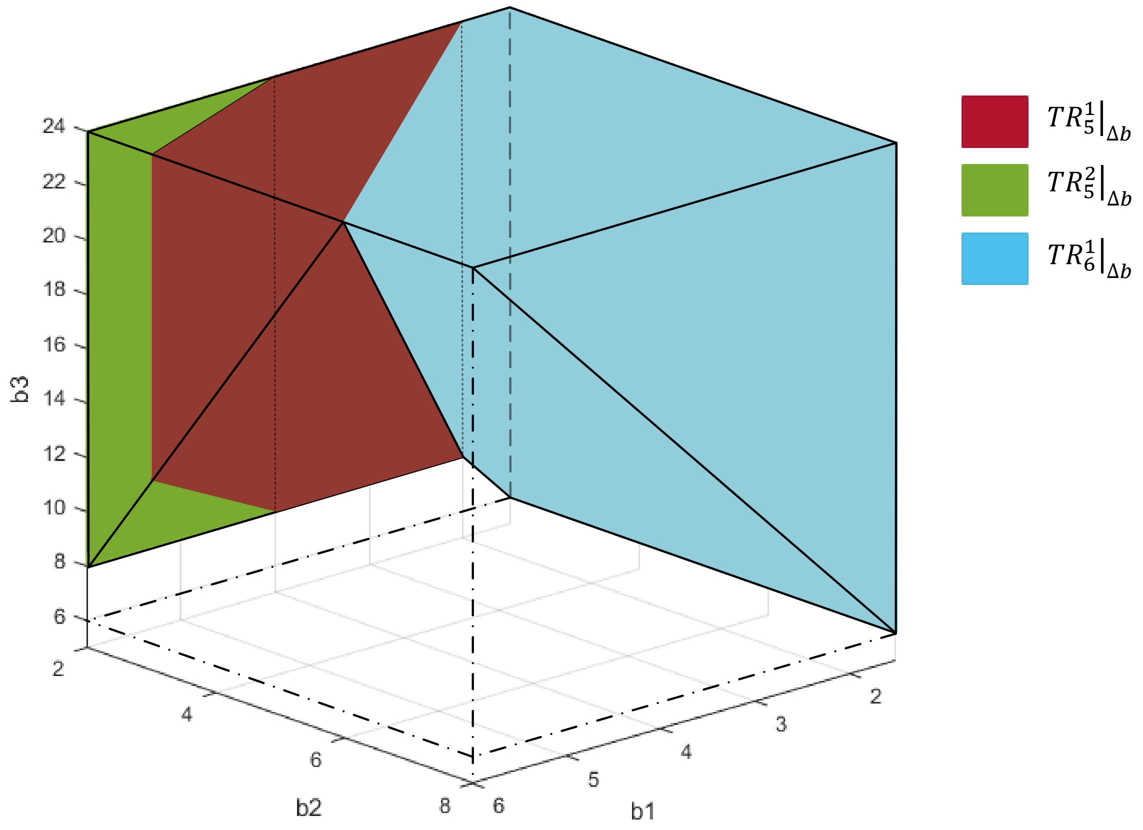


Figure 4.3 Region of $\mathrm{MR_1}$ that enclosed within the uncertainty region $\boldsymbol{con}(\Delta \boldsymbol{b})$

Note that the sum of $P(\boldsymbol{MR_1}|_{\Delta b})$ and $P(\boldsymbol{MR_2}|_{\Delta b})$ is 100% as expected. To assess the progressive increase in computational complexity as the number of uncertain parameters increases, the 3D case study discussed above is compared to the 2D case study presented in Chapter 3 (see section 3.3). It is evident from the comparison that for the 3D case, the main tableaus are generally containing more sub-tableaus as well as convex cones that compose each of them. Thus, the tableaus are less likely to be convex in 3D RHS space. In most cases, as the dimension of the RHS increases, the complexity of the tableau distribution also increases with respect to both the number of different active tableaus and the convex cones composing each one of them. However, this increase in complexity is mitigated to some extent, since generally only few of the tableaus are active in a certain RHS space. A typical example of higher dimension sensitivity analysis (6D RHS space case study) will be discussed below in a case study of a robust EMPC controller algorithm for a bioreactor which is the main focus of this thesis.

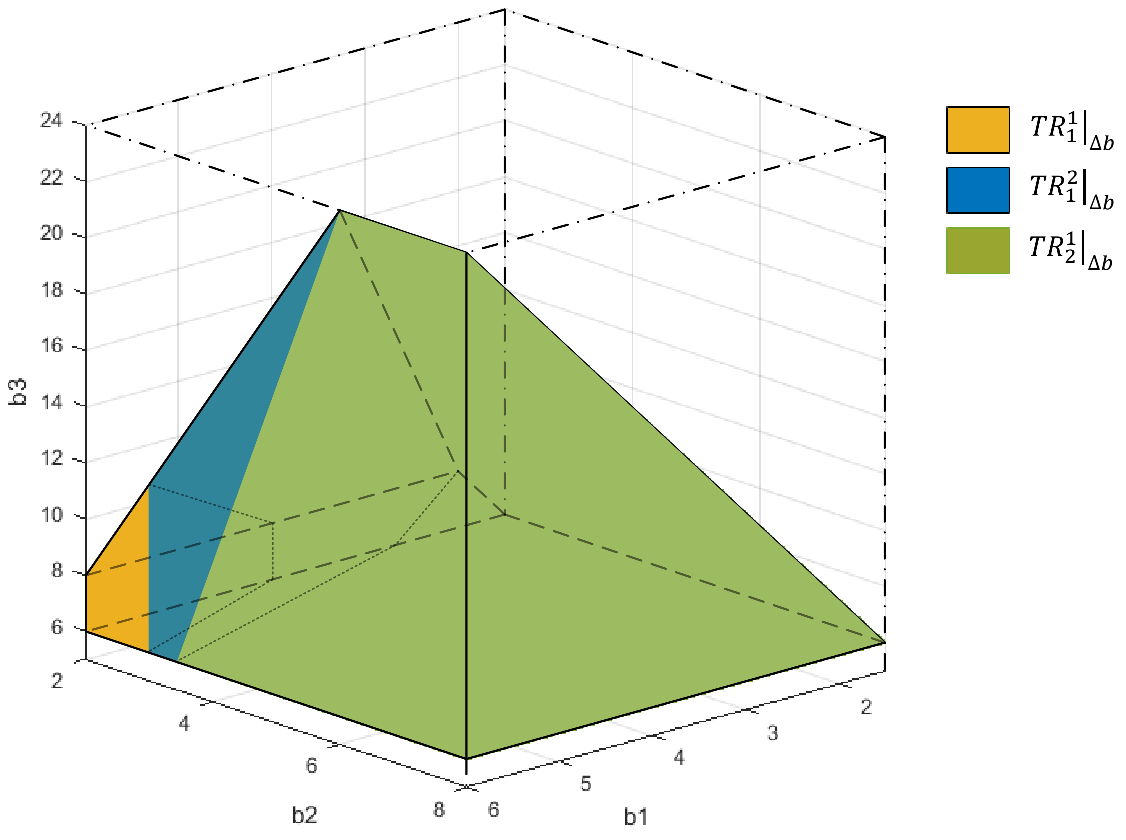


Figure 4.4 Region of MR_2 that enclosed within the uncertainty region $con(\Delta b)$

4.2 Robust NMPC Controller Formulation

As discussed in Chapter 3, the purpose of this novel algorithm is to propagate the uncertainty in model parameters onto a cost of an LP problem. An EMPC algorithm is proposed in this section which is made robust by using the CCM for uncertainty propagation along the prediction horizon. More specifically, the CCM algorithm in this work is tailored to the robust EMPC procedure where the biochemical processes are described by a dynamic metabolic flux model (DMFM) that is based on the solution of an LP problem at each time step. The DMFM is used to describe a batch bioreactor process and the objective is to maximize a cost at the end of the batch.

The general formulation of an EMPC controller involves the minimization or maximization of an economic terminal cost/penalty V_f function as earlier presented in equations (3.1) to (3.5). The bi-level optimization to be solved for the EMPC was introduced in (3.6) to (3.11). Additionally, the uncertain parameters on the Right Hand Side (RHS) ***b*** as well as the linearly supremum bounding procedure with respect to the uncertain parameters of the RHS were discussed in equations (3.12) to (3.15). The robust controller formulation of the current chapter is based on these earlier definitions.

Another important theoretical basis of the Tree (scenario) based structure of NMPC was discussed in section 2.3.4. As shown schematically in Fig. 4.5, the main idea of this method as reported in the recent literature is that the time trajectories corresponding to different parameter uncertainty or disturbance realizations can be represented by a tree composed of discrete scenarios. Each node of the branches in this tree structure is generated by uncertainty, i.e. parameter or disturbance related uncertainty. To avoid the increased computational cost expected with the exponential growth nature of the tree structure, branching with respect to different uncertainty realizations is only done for the initial time intervals of the prediction horizon (2 first intervals in Fig. 4.5) and then the uncertainty realization in each trajectory is assumed to remain constant until the end of the horizon (intervals 3 and 4 in Fig. 4.5).

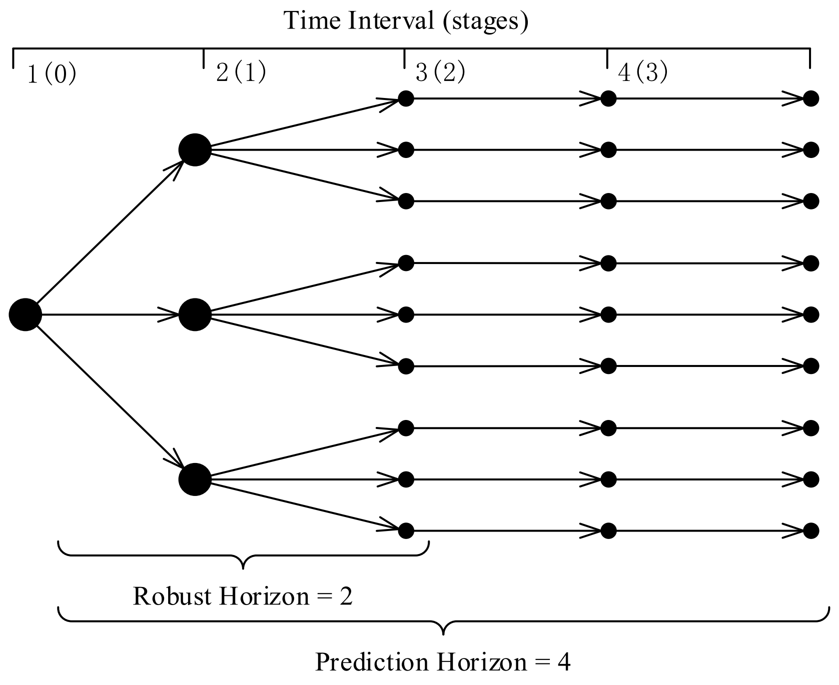


Figure 4.5 Uncertainty evolution with robust horizon represented by scenario tree structure

The most challenging problem of the scenario tree structure is how to generate a reasonable tree structure that maintains a balance between an accurate estimation of uncertainty and an acceptable size of the robust horizon. Many methods have been proposed to address this issue, such as the Monte Carlo simulations (Shapiro, 2003), the deterministic moment matching method (Høyland et al., 2003) of a probability distribution, the minimization of a certain probability matrix (de Oliveira et al., 2010) or machine learning techniques (Defourny, 2010). Several versions of the tree structure based MPC have been reported (Lucia, Finkler and Engell, 2013; Lindhorst et al., 2016). However, those frameworks are only acceptable for online implementation with a short prediction horizon or a limited number of uncertainties of the dynamic enzyme-cost Flux Balance Analysis (deFBA) model.

Considering these limitations, a novel tree structure based algorithm is proposed here in order to exploit the particular nature of the dynamic metabolic flux model describing the system, i.e. linear programming based model. The expectation is that the proposed approach would reduce the computational effort to estimate control actions in the presence of model uncertainty.

4.2.1 Tree Structure of Different Tableaus

The main assumption of the proposed EMPC algorithm is that the space of the outputs that are generated from the uncertainty parameters can be divided into a series of polyhedrons; for each polyhedron, max-min of outputs can be easily obtained based on linearity. The CCM procedure is used to find ranges of output values for each one of the Tableaus identified at each time step along the prediction horizon. Using the minimum and maximum values calculated at each time interval k, the ranges of outputs in a following time interval $k+1$ can be calculated using the CCM method together with the differential equations describing the mass balances in the DMFM model, the distribution of these time intervals has been shown as in Fig. 4.6 (b). Therefore, a tree structure uncertainty propagating method can be obtained by this approach where the tree expands along the prediction horizon from the current time interval, k. Thus, the novel controller algorithm proposed here will be referred to as a Tableau Based Tree (TBT) method. The schematic structure of the TBT algorithm is shown in Fig. 4.6 and Fig. 4.7. Each branch originating from a node represents a specific tableau corresponding to a particular range of uncertainty and for a particular input (control action or manipulated variable). In this structure, the path from the root node (initial state) x_0 to a leaf node is referred to as a scenario. The tree may branch at each stage where the branches originating from each node depends on the numbers of tableaus that are generated from the uncertain parameters. There are three different layers of sampling time interval in this prediction system, the basic sampling time interval is Δt, the instant where the new branches can be generated is referred as a new stage s and the time interval of each stage is Δs, the interval of the time where each controller inputs u can be manipulated is Δl. Generally, Δs can be equal or longer than Δt and Δl can be equal or longer than Δs, while the larger one of these values are generally be multiple times of the less one. For instance, as it has been shown in Fig. 4.6 (b), if the Δl is 2 times longer than Δs:

$$\Delta l = 2\Delta s$$

The control inputs of this tree structure are equal to each other every two stages, such as:

$$u_0^1 = u_1^1, \qquad u_2^1 = u_3^1$$

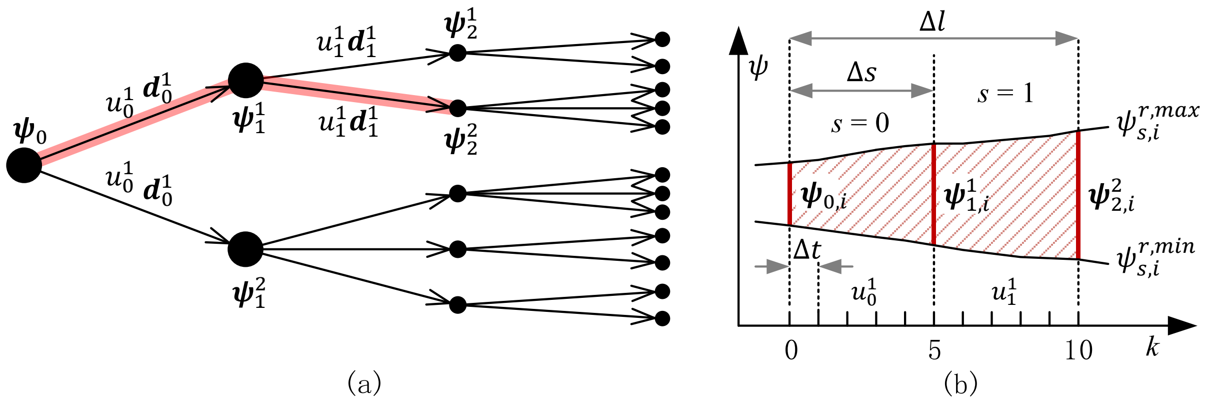


Figure 4.6 (a) Uncertainty evolution with robust horizon represented by scenario tree structure in TBT; (b) Uncertainty evolution of a specific parameter $\boldsymbol{\psi_i}$ in the first and second stages of an individual TBT scenario (the red bold line when $\boldsymbol{k = 0, 5, 10}$ is the range of $\boldsymbol{\psi_{0,i}}$, $\boldsymbol{\psi_{1,i}^1}$ and $\boldsymbol{\psi_{2,i}^2}$ corresponding to the nodes in the red scenario of figure 4.6 (a), respectively)

Additionally, from the CCM algorithm, the probability for each tableau to occur relative to the other tableaus can be calculated and thus, with this information, some of the leaves can be eliminated if the probability for them to occur is smaller than a certain user-defined threshold. Although some parameters, such as controller inputs, plants states and fluxes etc., may remain constant along different stages, they are denoted by different stage number s. For example, the values of uncertain parameters might remain constant during a certain scenario; however, they are marked as different superscripts in order to clearly identify their relative positions in the tree structure. Furthermore, the probability distribution of occurrence of parameters of the RHS of the constraints of the LP problem solved at each stage are assumed to be uniform in this work in order to simplify the problem, if not otherwise specified. Also, it is assumed that all of the possible states $\boldsymbol{S}_{\boldsymbol{\psi}_s^r}$ that are generated from each node $\boldsymbol{\psi}_s^r$ with the possibility $P_{s+1}^{*r(q)}$ of each states $\boldsymbol{\psi}_{s+1}^{r(q)}$ to occur within the tree scenario can be calculated from an uncertain nonlinear system that is represented as follows:

$$\boldsymbol{S}_{\boldsymbol{\psi}_s^r} = \begin{bmatrix} \left(\boldsymbol{\psi}_{s+1}^{r_1(1)}\right)^T & P_{s+1}^{*r(1)} \\ \left(\boldsymbol{\psi}_{s+1}^{r(2)}\right)^T & P_{s+1}^{*r(2)} \\ \vdots & \vdots \\ \left(\boldsymbol{\psi}_{s+1}^{r(k_T)}\right)^T & P_{s+1}^{*r(k_T)} \end{bmatrix} = f_{ns}(\boldsymbol{\psi}_s^r, P_s^r, u_s^r, \boldsymbol{d}_s^r) \tag{4.4}$$

where s is the identifier of different stages, r and $r(q)$ is the identifier of the leaves in one specific stage with $r \in [1,2, \dots, k_s]$ and $q \in [1,2, \dots, k_T]$, k_s is the amount of scenario in the stage s, k_T is the amount of the branches/tableaus that can be generated from node $\boldsymbol{\psi}_s^r$ with the controller inputs u_s^r and the uncertainty region $\boldsymbol{d}_s^r$, $\boldsymbol{\psi}_s^r$ is a matrix that contains two column vectors that illustrate the maximum and minimum states of the corresponding tableaus respectively:

$$\boldsymbol{\psi}_s^r = \begin{bmatrix} \boldsymbol{\psi}_s^{r,min} & \boldsymbol{\psi}_s^{r,max} \end{bmatrix} \tag{4.5}$$

the uncertainty region is defined as follows:

$$\boldsymbol{d}_s^r = \begin{bmatrix} \boldsymbol{d}_s^{r,min} & \boldsymbol{d}_s^{r,max} \end{bmatrix} \tag{4.6}$$

It is worth to note that the probability that obtained form (4.4) is not the actual probability P_{s+1}^r of each tableaus $(T|\boldsymbol{\psi}_{s+1}^r)$ for each of the states $\boldsymbol{\psi}_{s+1}^r$, a $*$ sign is used in the superscript of such probability symbols such as P_{s+1}^{*r} for avoiding the confusion of these two probabilities. The actual probability P_{s+1}^r can be computed by using the probability P_{s+1}^{*r} in (4.4) and the concept of conditional probability as follows:

$$P_{s+1}^r = P_{s+1}^{*r} P_s^r \tag{4.7}$$

In this work f_{ns} is an uncertain nonlinear system that is formulated by using the CCM algorithm combined with Euler integrations of the equation defined in (3.10). Therefore, each of the states $\boldsymbol{\psi}_{s+1}^{r(q)}$ and their corresponding probability $P_{s+1}^{*r(q)}$ in (4.4) are dependent on the current vector of states $\boldsymbol{\psi}_s^r$, the corresponding probability P_s^r, the corresponding control input u_s^r and the corresponding uncertainty region $\boldsymbol{d}_s^r$ at stage s and realization (scenario) r. For example, using Fig. 4.7, all the states $\boldsymbol{S}_{\boldsymbol{\psi}_2^3}$ that are generated from $\boldsymbol{\psi}_2^3$ can be represented as:

$$\boldsymbol{S}_{\boldsymbol{\psi}_2^3} = \begin{bmatrix} \boldsymbol{\psi}_3^5 & P_3^{*5} \\ \boldsymbol{\psi}_3^6 & P_3^{*6} \end{bmatrix} = f_{ns}(\boldsymbol{\psi}_2^3, P_2^3, u_2^3, \boldsymbol{d}_2^3)$$

where in this particular case, the previous stage is $s = 2$, the previous state identifier $r = 3$ and the two new generating branches are the 5th and 6th scenario of current stage, thus $r(1) = 5$ and $r(2) = 6$. Using CCM algorithm, the number of active tableaus within the uncertain region $\boldsymbol{d}_s^r$ is obtained as $\mathcal{K}_{RS} = 2$; thus, the number of branches k_T of the node $\boldsymbol{\psi}_2^3$ is:

$$k_T = \mathcal{K}_{RS} = 2 \tag{4.8}$$

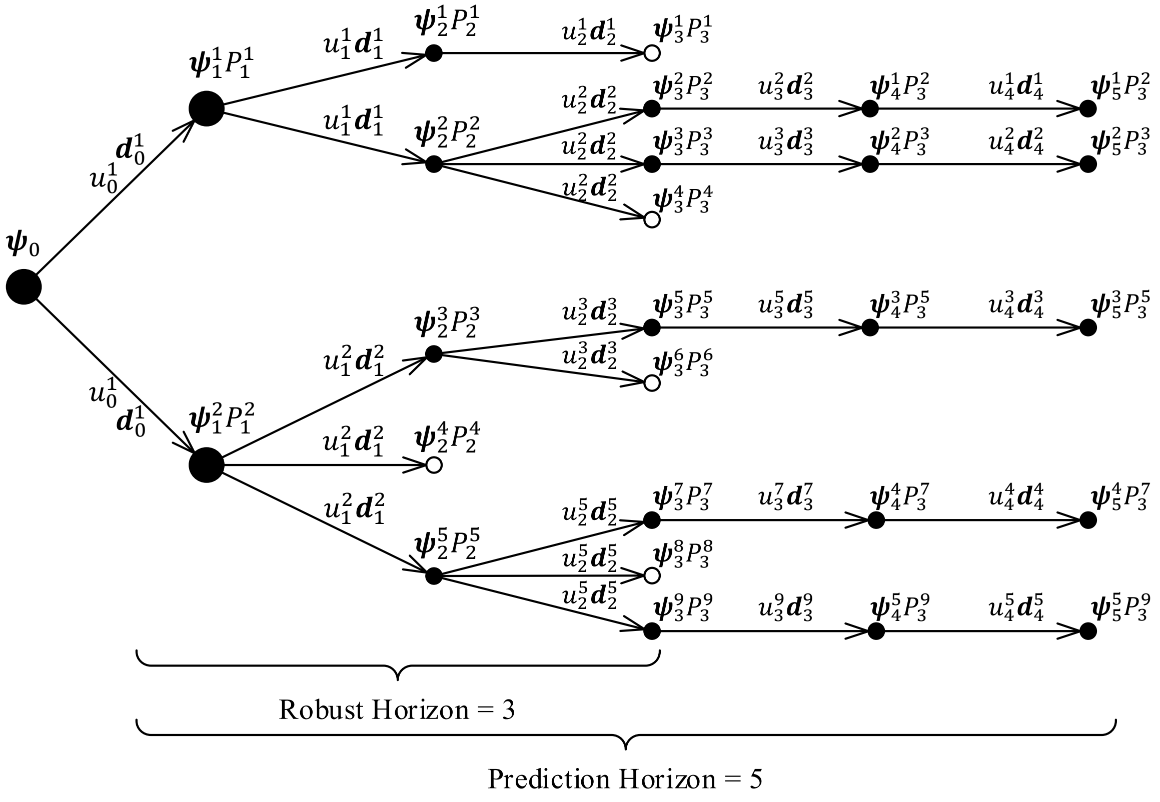

$\boldsymbol{\psi}_0$
$\boldsymbol{\psi}_1^1 P_1^1$
$\boldsymbol{\psi}_1^2 P_1^2$
$u_0^1 \boldsymbol{d}_0^1$
$u_1^1 \boldsymbol{d}_1^1$
$u_1^2 \boldsymbol{d}_1^2$
$\boldsymbol{\psi}_2^1 P_2^1$
$\boldsymbol{\psi}_2^2 P_2^2$
$\boldsymbol{\psi}_2^3 P_2^3$
$\boldsymbol{\psi}_2^4 P_2^4$
$\boldsymbol{\psi}_2^5 P_2^5$
$u_2^1 \boldsymbol{d}_2^1$
$u_2^2 \boldsymbol{d}_2^2$
$u_2^3 \boldsymbol{d}_2^3$
$u_2^5 \boldsymbol{d}_2^5$
$\boldsymbol{\psi}_3^1 P_3^1$
$\boldsymbol{\psi}_3^2 P_3^2$
$\boldsymbol{\psi}_3^3 P_3^3$
$\boldsymbol{\psi}_3^4 P_3^4$
$\boldsymbol{\psi}_3^5 P_3^5$
$\boldsymbol{\psi}_3^6 P_3^6$
$\boldsymbol{\psi}_3^7 P_3^7$
$\boldsymbol{\psi}_3^8 P_3^8$
$\boldsymbol{\psi}_3^9 P_3^9$
$u_3^2 \boldsymbol{d}_3^2$
$u_3^3 \boldsymbol{d}_3^3$
$u_3^5 \boldsymbol{d}_3^5$
$u_3^7 \boldsymbol{d}_3^7$
$u_3^9 \boldsymbol{d}_3^9$
$\boldsymbol{\psi}_4^1 P_3^2$
$\boldsymbol{\psi}_4^2 P_3^3$
$\boldsymbol{\psi}_4^3 P_3^5$
$\boldsymbol{\psi}_4^4 P_3^7$
$\boldsymbol{\psi}_4^5 P_3^9$
$u_4^1 \boldsymbol{d}_4^1$
$u_4^2 \boldsymbol{d}_4^2$
$u_4^3 \boldsymbol{d}_4^3$
$u_4^4 \boldsymbol{d}_4^4$
$u_4^5 \boldsymbol{d}_4^5$
$\boldsymbol{\psi}_5^1 P_3^2$
$\boldsymbol{\psi}_5^2 P_3^3$
$\boldsymbol{\psi}_5^3 P_3^5$
$\boldsymbol{\psi}_5^4 P_3^7$
$\boldsymbol{\psi}_5^5 P_3^9$
Robust Horizon = 3
Prediction Horizon = 5

Figure 4.7 A typical uncertainty evolution represented by scenario tree structure in TBT

For simplicity, the set of possible indices of branches $\boldsymbol{I}_B(s, r)$ in the scenario tree is referred to as $\boldsymbol{I}_B$. Additionally, if the probability P_{s+1}^{*r} of a new branch is less than a criterion P_{min}^* that is defined a priori, then this branch is set as a non-active branch $\boldsymbol{I}_B^{na}(s + 1, r)$. Thus, this branch will be eliminated from the scenario tree $\boldsymbol{I}_B$ and will not be used as a node in the branch generating process in the next stage. The set of non-active branches is denoted as $\boldsymbol{I}_B^{na}$ and are determined as per the following criterion:

$$P_{s+1}^{*r} \leq P_{min}^* \tag{4.9}$$

where P_{min}^{*} is the largest probability that determines one branch from the possible states set $\boldsymbol{S}_{\boldsymbol{\psi}_s^r}$ can be eliminated. Meanwhile, the number of active branches from the parent node, which means the node that has generated these branches, is denoted as r_{act}. The set of active branches is denoted as $\boldsymbol{I}_B^{act}$. Therefore, the probability P_{s+1}^r of active branch $\boldsymbol{I}_B^{act}(s+1,r)$ will be re-normalized by using (4.10) to compensate for the inactive branches that were eliminated:

$$P_{s+1}^{r} = \frac{P_{s+1}^{*r} P_s^r}{\sum\left(P^{*} | \boldsymbol{I}_B(s+1,r)\right)} \quad \left(\boldsymbol{I}_B \in \boldsymbol{I}_B^{act}\right) \tag{4.10}$$

where P_{s+1}^{*r} and $\left(P^{*}|\boldsymbol{I}_B(s+1,r)\right)$ means the corresponding probability of the tableau that is obtained from (4.4), i.e. the branch $\boldsymbol{I}_B^{act}(s+1,r)$. For instance, in Fig. 4.7 if the $\boldsymbol{I}_B(2,4)$ is a non-active branch, the state of this node $\boldsymbol{\psi}_2^4$ is denoted by a white dot, and the probability of the branch that originates from the same parent node $\boldsymbol{I}_B(1,2)$ with (4.4) are P_2^{*3} and P_2^{*5}. Then by using (4.10) the probability of the node $\boldsymbol{I}_B(2,3)$ that will be further used in the tree scenario is:

$$P_2^3 = \frac{P_2^{*3} P_1^2}{P_2^{*3} + P_2^{*5}} \quad \left(\boldsymbol{I}_B(2,3) \in \boldsymbol{I}_B^{act}, \qquad \boldsymbol{I}_B(2,5) \in \boldsymbol{I}_B^{act}\right)$$

Similarly, the probability of the node $\boldsymbol{I}_B(2,5)$ is:

$$P_2^5 = \frac{P_2^{*5} P_1^2}{P_2^{*3} + P_2^{*5}} \quad \left(\boldsymbol{I}_B(2,3) \in \boldsymbol{I}_B^{act}, \qquad \boldsymbol{I}_B(2,5) \in \boldsymbol{I}_B^{act}\right)$$

Similar to the tree scenario work previously reported in the literature (Lindhorst et al., 2016; Lucia et al., 2014), for simplicity, the branching with respect to different tableaus is only done for the initial 3 first time intervals as shown in Fig. 4.7 corresponding to the last scenario $\boldsymbol{I}_B(5,5)$. Then the tableaus in each trajectory are assumed to remain constant at the tableau (branch) that has the largest probability among the possible branches (generally this branch is at the nominal value) until the end of the horizon (intervals 4 and 5 in Fig. 4.7 for the last scenario $\boldsymbol{I}_B(5,5)$). This approach can be justified as follows: as the branching procedure progresses over the prediction horizon, it is expected that the sum of the probabilities $\sum\left(P_{s+1}^r | \boldsymbol{I}_B^{act}(s+1,r)\right)$ of all tableaus that are generated from the same parent node is smaller than the threshold P_{min} as defined in (4.11). Then, the number of branches in the rest of time intervals will remaining constant as shown by condition (4.12). Furthermore, the probability of each of the remaining branches in the rest of time intervals until the end of the prediction horizon l_P is assigned the last calculated probability for that branch calculated at the end of the robust horizon l_R as shown by condition (4.13). Therefore, the resulting lengths of the robust horizon l_R, which is defined in (4.14) by substituting (4.10) into (4.11), are different with respect to different scenario in this tree structure, where P_{s-1}^{r} is the probability of the parent node of $\boldsymbol{I}_B^{act}(s,r)$. Additionally, the prediction horizon is fixed in every tree structure, as $l_P = 5$ in the example shown in Fig. 4.7. As a merit of this method, the scenario with lager probability maintains longer robust horizon, which can provide a reasonable allocation for the computation effort.

$$\sum \left(P_{s+1}^{r} | \boldsymbol{I}_B^{act}(s+1,r)\right) \le P_{min} \tag{4.11}$$

$$\boldsymbol{\psi}_{s+1}^{r} = (\boldsymbol{\psi}_{s+1}^{r} | P_{s+1}^{*r} \ge \forall P_{s+1}^{*r}) \tag{4.12}$$

$$P_{s+1}^{r} = P_{l_R}^{r} \tag{4.13}$$

$$\left\{ l_R = s | \sum \left(P_s^r | \boldsymbol{I}_B^{act}(s,r)\right) \ge P_{min}, \qquad \forall P_s^r \le P_{min}, P_{s-1}^{r} \ge P_{min} \right\} \tag{4.14}$$

For instance, based on Fig. 4.7, if the sum of the probability of the active branches originating from parent node $\boldsymbol{I}_B(1{,}2)$ are:

$$(P_2^3 + P_2^5) \ge P_{min}$$

and the probability of $\boldsymbol{I}_B(2{,}3)$ is less than P_{min}, then the length of the robust horizon $l_R = 3$ for scenario $\boldsymbol{I}_B(3{,}5)$. If there are 2 tableaus that can be generated from $\boldsymbol{I}_B(2{,}3)$, which are $\boldsymbol{I}_B(3{,}5)$ and $\boldsymbol{I}_B(3{,}6)$, the probability of $\boldsymbol{I}_B(3{,}5)$ is larger than $\boldsymbol{I}_B(3{,}6)$, then by using (4.12):

$$\boldsymbol{\psi}_3^5 = (\boldsymbol{\psi}_3^5 | P_3^{*5} \ge P_3^{*6})$$

and the probability of this tableau/scenario $\boldsymbol{I}_B(3{,}5)$ is $P_3^5 = P_2^3$ and remains at that value beyond the end of the robut horizon until the end of the prediction horizon.

4.2.2 Mathematical Formulation

A tree based scenario optimization problem to accomplish economic predictive control can be formulated using the Tableau Based Tree (TBT) branch-pruning strategy presented in the previous section as follows:

$$\min_{u_s^r, \forall (s,r) \in \boldsymbol{I}_B^{act}} \hat{J} \tag{4.15}$$

s.t.

$$\boldsymbol{\psi}_{s+1}^{r} = f(\boldsymbol{\psi}_s^r, u_s^r, \boldsymbol{d}_s^r), \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.16}$$

$$P_{s+1}^{r} = g(\boldsymbol{\psi}_s^r, P_s^r, u_s^r, \boldsymbol{d}_s^r), \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.17}$$

$$\boldsymbol{\psi}_s^r \in \boldsymbol{\Psi_f}, \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.18}$$

$$u_s^r \in \mathbb{U}_{\boldsymbol{f}}, \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.19}$$

where $\hat{J}$ is defined as follows:

$$\hat{J} = V_f\left(\sum_{r=1}^{N_s} P_{s_f}^r J_{s_f}^r\right), \qquad \forall (s_f, r) \in \boldsymbol{I}_B^{act} \tag{4.20}$$

$J_{s_f}^r$ is the cost function of each one of the scenario $\boldsymbol{I}_B^{act}(s_f, r)$ at the final stage s_f, $P_{s_f}^r$ is the probability of $J_{s_f}^r$. N_s is the total number of branches in the s stage. V_f is the function that quantifies the economic terminal cost/penalty. The states $\boldsymbol{\psi}_s^r$ and controller actions u_s^r must be within their pre-specified constraints' sets $\boldsymbol{\Psi_f}$ and $\mathbb{U}_{\boldsymbol{f}}$ as defined in (4.18) and (4.19), respectively. To achieve robustness, the cost function $J_{s_f}^r$ of each one of r scenarios can be defined as follows:

$$J_{s_f}^r = L_f \boldsymbol{\psi}_{s_f}^{r,min}, \qquad \forall \psi_{s_f,i}^{r,min} \in \boldsymbol{\psi}_{s_f}^{r,min}, \forall i \in \mathbb{N}_{obj}, \forall (s_f, r) \in \boldsymbol{I}_B^{act} \tag{4.21}$$

where $\psi_{s_f,i}^{r,min}$ is the minimum value of the i-th element in terminal branch $\boldsymbol{I}_B^{act}(s_f, r)$, which can be obtained by using (4.5), and the set $\mathbb{N}_{obj}$ is the identifier of the elements in $\boldsymbol{\psi}$ that are being optimized. L_f is a function that reflect the significance of each $\psi_{s_f,i}^{r,min}$ into the cost function in (4.21). In the current work, for our particular bioreactor study only one of the elements in $\boldsymbol{\psi}$ needs to be optimized, i.e. the one corresponding to the growth rate. In this case, this specific element is denoted as $\psi_{s_f,obj}^{r,min}$; thus, the cost function can be defined as in (4.22).

$$J_{s_f}^r = \psi_{s_f,obj}^{r,min}, \qquad \forall (s_f, r) \in \boldsymbol{I}_B^{act} \tag{4.22}$$

It is necessary to establish a time-discrete model since discretization is the nature of the scenario tree structure model. However, most of the process models are formulated by using a continuous time domain since they describe continuous balances by sets of ODE's, i.e.

$$\dot{\psi} = \Omega(\psi, u, d) \tag{4.23}$$

Additionally, the NMPC algorithm that is applied in the current work involves both the control actions as well as the model states as decision variables of the optimization problem. Hence, the states described by ODEs need to be discretized thus resulting in a nonlinear optimization (NLP) problem. To address this problem, the ODE's are discretized and calculated over time by using the

Euler discrete integrations of the DMFM model (equation 3.10). The resulting discretized model can be written as follows:

$$\boldsymbol{\psi}_s^r(t+1) = \boldsymbol{\psi}_s^r(t) + \Delta t\big(\Omega(\boldsymbol{\psi}_s^r, u_s^r, \boldsymbol{d}_s^r)\big), \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.24}$$

where Δt is the duration of each time interval and each of the stages s may contain several sampling time intervals t and generally Δs needs to be multiple times of Δt. Two main options can be implemented to improve the accuracy of (4.24): i) reduce the time intervals of Δt and Δs and ii) use a different algorithm from Euler integrations for time integration of the differential equations of the model. For instance, the forward differentiation formula (FDF) of 1st order was used in the current study to integrate eq. (4.24). To further improve the accuracy of the integration, higher order methods explicit *Runge-Kutta* methods such as ode45 in MATLAB can be adopted.

It is worth noticing that the control inputs u_s^r should be changed every α_u ($\alpha_u \in \mathbb{Z}$) times for each stage, the actual time interval between each time of manipulation l is Δl. Also, the uncertainty range $\boldsymbol{d}_s^r$ is always assumed to involve percentages of the region $\boldsymbol{\delta_d}$ with respect to the nominal parameters $\overline{\boldsymbol{d}}$ that result in feasible solutions of the optimization problem. Thus, to ensure feasibility two additional constraints are added to the optimization problem as follows:

$$u_s^r = u_l^r, \qquad \boldsymbol{I}(s-1,r) \to \boldsymbol{I}(s,r), s \leq l\alpha_u < s + \alpha_u, \forall (s,r) \in \boldsymbol{I}_B^{act}, \forall u_l^r \in \boldsymbol{U}_L \tag{4.25}$$

$$\boldsymbol{d}_s^r = \boldsymbol{\delta_d} \cdot \overline{\boldsymbol{d}}, \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.26}$$

where $\boldsymbol{U}_L$ is the set of the input values u of this model and l is a subscript identifying each of the elements u_l^r in $\boldsymbol{U}_L$. The constrains in (4.25) are also introduced to ensure that decisions (control inputs) anticipating the future are avoided. These are generally referred as the *non-anticipatively* constraints. That is, the branches generated from same parents nodes share same control inputs in every α_u stage, just as what has been shown in Fig. 4.7 where $\alpha_u = 1$, $\boldsymbol{I}(1,1)$ and $\boldsymbol{I}(1,2)$ are computed by using same control inputs u_0^1. Thus, different scenario in TBT method maintains different profiles of control inputs. In some cases, the size of the set $\boldsymbol{U}_L$ might be unstable during optimization. This problem can be partially addressed by introducing a fixed number of active scenarios $\alpha_{I,s}$ for each stage s. For example, in Fig. 4.7, for the 0-2 stages:

$$\boldsymbol{U}_L = \{u_0^1, u_1^1, u_1^2, u_2^1, u_2^2, u_2^3, u_2^5\}$$

the u_2^4 is not in this set since the corresponding scenario $\boldsymbol{I}(2,4)$ has been eliminated with $P_2^4 \leq P_{min}$. However, if the value's changing of u_0^1 during optimization result in $P_2^4 > P_{min}$, the u_2^4

should be introduced in $\boldsymbol{U}_L$ and thus the size of $\boldsymbol{U}_L$ is unstable. If α_I is available in this stage as $\alpha_{I,2} = 4$, which means the active amount of scenario of the 2nd stage is fixed to 4, and:

$P_2^4 \leq \forall P_2^r$

then the scenario $\boldsymbol{I}(2,4)$ still not be active thus the size of $\boldsymbol{U}_L$ is maintained stable. Generally, the $\alpha_{I,s}$ of each stage s are different according to different models and can be obtained by analyzing the normal condition of the TBT branching structure.

Based on the above, the overall NLP optimization problem to be solved for calculating control actions is based on a two layers' optimization that has been previously partially introduced in $(3.6) - (3.11)$. This problem is now re-formulated as an EMPC (Economic Model Predictive Control Problem) by using the proposed tree structure as follows:

$$\min_{\boldsymbol{U}_L} \; V_f\left(\sum_{r=1}^{N_s} P_{s_f}^r \psi_{s_f,obj}^{r,min}(t_f)\right), \qquad \forall(s_f, r) \in \boldsymbol{I}_B^{act} \tag{4.27}$$

$$\begin{aligned}
\text{s.t.}\quad & Eqs. (4.4) - (4.14) \\
& \boldsymbol{\psi}_s^r(t+1) = \boldsymbol{\psi}_s^r(t) + \Delta t\big(\Omega(\boldsymbol{\psi}_s^r, u_s^r, \boldsymbol{d}_s^r)\big), \qquad \forall(s,r) \in \boldsymbol{I}_B^{act} && (4.28) \\
& \boldsymbol{\psi}_{s+1}^r = \boldsymbol{\psi}_s^r(t+\Delta s), \qquad \forall(s,r) \in \boldsymbol{I}_B^{act} && (4.29) \\
& \boldsymbol{\psi}_s^r \in \boldsymbol{\Psi}_{\boldsymbol{f}}, \qquad \forall(s,r) \in \boldsymbol{I}_B^{act} && (4.30) \\
& u_s^r \in \mathbb{U}_{\boldsymbol{f}}, \qquad \forall(s,r) \in \boldsymbol{I}_B^{act} && (4.31) \\
& Eqs. (4.25) - (4.26)
\end{aligned}$$

where the economic terminal cost/penalty V_f in (4.27) includes the sum of the minimum condition at the end of each scenario, in which the objective function is $\psi_{s_f,obj}^{r,min}$ at the terminal time t_f of the final stage s_f, multiplied by its corresponding probability $P_{s_f}^r$. The probability of each node P_s^r at each time interval can be obtained by using $Eqs. (4.4) - (4.14)$, i.e. the equations defining the TBT strategy outlined before for creating the tree structure and for pruning the branches with low probability. The function Ω represents the discretized dynamics (4.28). This function is formulated via the CCM algorithm that was introduced in section 3.2.5 and the uncertainty propagation procedure that was presented in 3.1. The initial states $\boldsymbol{\psi}_{s+1}^r$ of each stage is equal to the final states $\boldsymbol{\psi}_s^{r,min}(t+\Delta s)$ of the previous stage. Additional constraints for this model are bounds on the states (4.30) and the inputs (4.31). Equations $(4.25) - (4.26)$ provide the non-anticipatively constraints and the uncertainty constraints, respectively.

4.2.3 Theoretical Comparison of Computation Effort for the TBT Approach and the Monte Carlo Based Approach for Uncertainty Propagation

In this section, the TBT method that forms the basis of the EMPC algorithm is compared on a theoretical basis with other methods such as Monte Carlo sampling and PCEs.

Generally, the computational costs of Monte Carlo simulation and PCEs highly depends on the number of uncertain parameters that are being considered. For instance, for Monte Carlo simulation, if the number of uncertain parameters in the system is ζ, then the number N_M of the repeating random sampling in each time interval is as in (4.32) needs to be large enough so as to let the distribution of the sample points approaching to become uniform.

$$N_M = \mathcal{E}^{\zeta} \tag{4.32}$$

where $\mathcal{E}$ is a tuning parameter that is generally related to the sampling size of the input space. For instance, when $\mathcal{E}$ is increasing, the accuracy of the Monte Carlo algorithm is also increasing. Thus, $\mathcal{E}$ cannot be very small so as to avoid prohibitive computational effort. For a simple comparison, let the Monte Carlo simulation be applied in the tree scenario structure method illustrated schematically in Fig. 2.4. If the robust horizon of this model is r_μ, then the number of calculations for sampled values k_{TM} for this particular scenario in this tree structure can be calculated. Since each sample of N_M would generate N_M samples in the next time interval then the corresponding number of sampled values is determined as follows:

$$k_{TM} = \mathcal{E}^{r_\mu \zeta} \tag{4.33}$$

If the time required for each scenario is τ_c, then time T_M that is required for the procedure can be calculated as follows:

$$T_M = k_{TM}\tau_c \tag{4.34}$$

For comparison, for the TBT approach, let assume that the space of the nonlinear system is divided into N_T polyhedrons. Since the model for each polyhedrons of the process are linear, then the maximum and minimum of the variable of interest within each of these polyhedrons can be obtained by two calculations, i.e. minimum and maximum bounds, for each N_T. Thus, the amount k_T of braches that are generated in every time interval for each node is as follows:

$$k_T = 2N_T \tag{4.35}$$

Without initial consideration of the probability of each branch, the robust horizon of this model is r_μ , the number of k_{TT} calculations of this scenario in this tree structure is:

$$k_{TT} = (2N_T)^{r_\mu} \tag{4.36}$$

In this case, if the computational time consumed for each scenario is τ_c, then the total time T_T that consumed for the TBT procedure will be as follows:

$$T_T = k_{TT}\tau_c \tag{4.37}$$

Without consideration of other details of the algorithm that might further affect the computational effort of either the MC approach or our proposed TBT approach, such as the calculation of the vertices generating algorithm and the volume obtaining algorithm of each polyhedron, substituting (4.33) and (4.36) into (4.34) and (4.37) provides a ratio of the computational costs of the two methods, i.e.

$$\frac{T_M}{T_T} = \frac{\mathcal{E}^{r_\mu \zeta}}{(2N_T)^{r_\mu}} \tag{4.38}$$

Generally, $\mathcal{E}$ and N_T are of the same order of magnitude, and each symbol in (4.38) are positive integers, thus, the time required by the Monte Carlo based method is approximately $(\mathcal{E}^{r_\mu})^{\zeta-1}$ times of that required by the present TBT method. For instance, for $\mathcal{E} = 10$, $r_\mu = 3$ and $\zeta = 7$, the computational time of the Monte Carlo based method is expected to be 1×10^{18} times the computational time of the TBT method.

From this comparison, it can be concluded that the TBT method have the potential to save considerable computational costs. The explanations for this computational advantage are: i) the computational time in the TBT method is cubic, for a robust horizon of 3 time intervals, with respect to the number of polyhedrons N_T, created by the intersection of the constraints of the LP problem whereas for the computation in MC is exponential with respect to the number of uncertain parameters ζ, as shown in (4.38); ii) the TBT method also offers an efficient approach for calculating probability for each node of the tree structure; thus, the scenario with less probability will be eliminated in time thus reducing the computational costs.

On the one hand, it should be pointed out that the ability to save computational cost by pruning branches of low associated probability depends on the simplifying assumption that the probability distribution of occurrence of parameters in each polyhedron is uniform. In principle, the probability distribution of each node could be identified from experimental data and assuming that the data would be corrupted by sensor noise, the resulting probability distribution of each node would obey the central limit theorem as generally assumed in the Monte Carlo method. On the other hand, it can be shown that due to the propagation of uncertainty along the robust horizon, the assumption of uniform probability may not be very restrictive since the probability distribution of each node results from the concatenation of probabilities of previous branches to that node. For example, for any one of the parameters from the system states ψ_i, if the probability distribution in a parent node of ψ_i is denoted as $f_p(\psi_i)$, and the probability distribution in current node of ψ_i is referred to as $f_c(\psi_i)$. Assuming that $f_p(\psi_i)$ is uniform, the function $f_r(\psi_i)$ is represented as follows:

$$f_r(\psi_i) = \begin{cases} P_s^r\left(\psi_i^{max} - \psi_i^{min}\right)^{-1} & \psi_i^{min} \le \psi_i \le \psi_i^{max} \\ 0 & \psi_i \le \psi_i^{min}, \psi_i^{max} \le \psi_i \end{cases} \qquad \forall (s,r) \in \boldsymbol{I}_B^{act} \tag{4.39}$$

Then, the $f_c(\psi_i)$ can be calculated via the convolution of $f_p(\psi_i)$ and $f_r(\psi_i)$ as follows:

$$f_c(\psi_i) = \int_{-\infty}^{\infty} f_p(\psi) f_r(\psi_i - \psi) d\psi \tag{4.40}$$

In the first stage, the $f_p(\psi_i)$ is only an initial point, thus the distribution of $f_c(\psi_i)$ via (4.40) is a rectangular function as shown in Fig. 4.8(a). However, in the second stage, following (4.40) the distribution of $f_c(\psi_i)$ is a convolution of several rectangular functions thus resulting in a non-rectangular distribution of $f_c(\psi_i)$ as shown in Fig. 4.8(b). If uncertainty in $\boldsymbol{I}_B^{act}$ is propagated through several stages, it is expected that the distribution of $f_c(\psi_i)$ will approach a normal distribution as Fig. 4.8(c) in some extent by using the convolution operation represented by (4.40).

However, the differences between the probability distributions occurring between the nodes can impose additional demand for extra sampling if accurate calculations of the distribution are required. For instance, if the range of rectangular functions $f_r(\psi_i)$, which are generated from the probability distribution in the parent node $f_p(\psi_i)$, are not much different from each other the probability distribution in the current node $f_c(\psi_i)$ can be obtained without extra sampling. On the other hand, if $f_r(\psi_i)$ from the same parent node are very different from each other, additional sampling would be necessary. These distributions are not taken explicitly into account in the TBT method, since this method is mainly focused on the marginal property of each node/tableau, such as the maximum and minimum condition of the distribution set. However, if non-uniform probability assumption is made or if it is desired to calculate accurate probability distributions at each stage, further sampling is needed thus potentially increasing the computational effort. A study on these topics is left for future research studies.

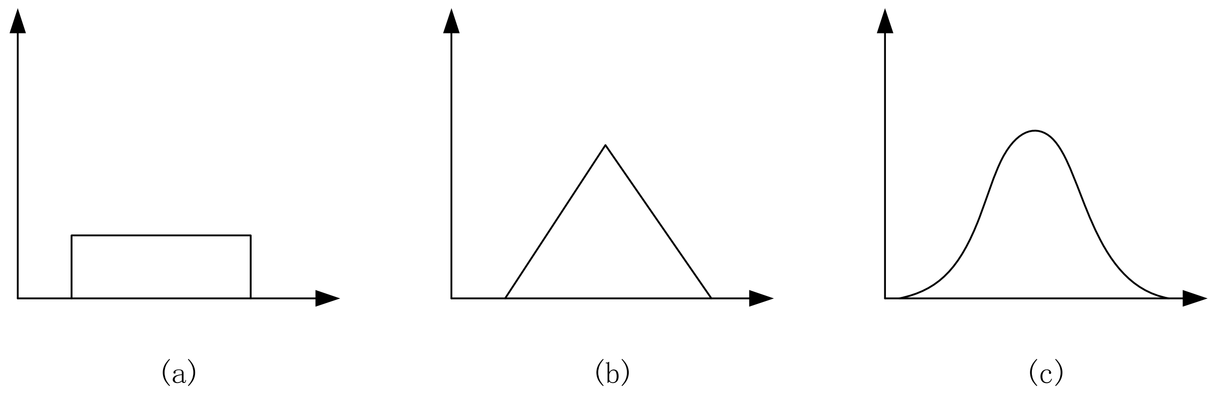


Figure 4.8 Probability distribution of one parameter at different stages

Finally, it is worth noticing that, just as for the parameter $\mathcal{E}$ in the Monte Carlo based method, the number of linearly polyhedrons N_T in solution space also can be adjusted in order to maintain a balance between the computational effort and the accuracy of the algorithm. In some cases, the polyhedrons with similar inherent linear coefficients can be considered as one linear polyhedron. Thus, the number of polyhedrons N_T can be controlled thus saving additional computational costs. One additional opportunity in this study for computational savings is by taking advantage of the use of the two levels of tableaus (main tableaus and sub-tableaus) that were discussed in section 3.2.5. If the differences between main tableaus is much larger than the differences among the sub-tableaus contained within each of them, then only the main tableaus could be considered for uncertainty propagation.

4.3 Robust Control Based on DFBM (Case Study)

In section 4.2, the TBT method and its advantages for propagating uncertainty were outlined. In this section, a robust controller based on the idea of Economic Model Predictive Controller that uses a DFBM as internal model is proposed. The robustness of the proposed algorithm is provided by using the TBT uncertainty propagation approach.

4.3.1 Dynamic Flux Balance Model (DFBM)

Dynamic Flux Balance Model (DFBM) is based on an a priori known network of n metabolites participating in m different reactions where the vector of metabolites' concentrations is represented by $\boldsymbol{z}_{\boldsymbol{n}\times\boldsymbol{1}}$. Each reaction in this model is associated to a flux, $\boldsymbol{\nu}_{\boldsymbol{n}\times\boldsymbol{1}}$ given in units of metabolite/hr/mM of cell. A stoichiometric matrix $\boldsymbol{\mathcal{A}}_{m\times n}$ is formulated to describe the stoichiometric relations between all metabolites according to the reactions considered in the metabolic network. As discussed in Section 2.6, the model assumes that the organism is optimal by allocating resources to maximize the growth rate at all time. This fundamental assumption indicates that the cells have adapted through natural evolution to act as an optimizer of resources. Based on this assumption, the DFBM model can be formulated as a Linear Programming (LP) problem, based on the flux balance equations and the defined stoichiometric matrix as follows:

$$\begin{aligned} &\max_{X,\boldsymbol{\nu}} \mu = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{\nu} \\ &s.t.\;\; \boldsymbol{\mathcal{A}\nu} \le \boldsymbol{b}, \qquad \frac{d\boldsymbol{z}}{dt} = \boldsymbol{\mathcal{A}\nu}X, \qquad \frac{dX}{dt} = \mu X \\ &\qquad |\dot{\boldsymbol{\nu}}| \le \dot{\boldsymbol{\nu}}_{max}, \qquad \boldsymbol{\nu}, \boldsymbol{z} \ge 0 \end{aligned} \tag{4.41}$$

where $\boldsymbol{b}_{m\times 1}$ represents a vector of consumption or production rate of extracellular metabolites such as nutrients and by-products, X is concentration of biomass, $\boldsymbol{z}$ is the current metabolites' concentrations, $\boldsymbol{w}$ is a vector that indicates the amounts of the growth precursors required per gram of biomass (Mahadevan et al., 2002). This dynamic flux model have been previously used by our research group for predictive control by (Kumar & Budman, 2017) based on a PCEs (Polynomial Chaos Expansion) method. Implementation of this technique resulted in computational demands that increased exponentially as a function of the uncertainty parameters of the model and consequently it was applied in that earlier study for a case study with only two uncertain parameters. In the current work, the TBT method is adopted to address this computational problem when considering a larger number of uncertain parameters.

4.3.2 Modeling with Uncertainty

The bioreactor study in the current work involves the development of a robust-EMPC based on a DFBM given in (4.41). The objective of the EMPC is to maximize the biomass at the end of the culture by manipulating the feeding rate of fresh media and the perfusion rate of supernatant. A schematic process flowsheet of this process has been shown in Fig. 4.9 with the indication of the main inputs and outputs. As for the perfusion operation it is assumed that only supernatant is perfused while all the cells are returned back to the bioreactor vessel. Accordingly, the dynamic mass balances that account for the feeding rate r_F, the perfusion rate r_P and the resulting changing reactor contents' volume V are formulated in terms of the fluxes $\boldsymbol{\nu}$, as per equations (4.42) -(4.44). Perfusion is implemented in the current work to ensure that the negative impact on growth by accumulation of high levels of some metabolites, such as acetate and glucose, is avoided (Mahadevan et al., 2002).

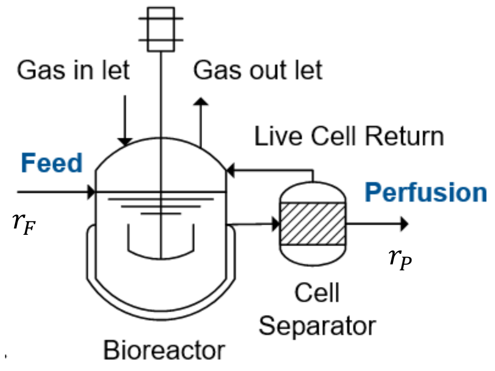


Figure 4.9 A schematic process flowsheet of the bioreactor with the feed and perfusion system

$$\frac{dV}{dt} = r_F - r_P \tag{4.42}$$

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{\mathcal{A}}\boldsymbol{\nu}X + \frac{r_F(\boldsymbol{z_{feed}} - \boldsymbol{z})}{V} = f(\boldsymbol{\mathcal{A}}, r_F, V, r_P, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{\nu}, X, \boldsymbol{z}) \tag{4.43}$$

$$\frac{dX}{dt} = \mu X - \frac{X(r_F - r_P)}{V} = g(\boldsymbol{\mathcal{A}}, r_F, V, r_P, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{\nu}, X, \boldsymbol{z}) \tag{4.44}$$

where $\boldsymbol{z_{feed}}$ is the concentration of metabolites in the feed and functions f and g represent the RHS of ODE's for the metabolites, i.e. $\boldsymbol{z}$ in (4.43), and biomass, i.e. X in (4.44). The DFBM model is then formulated as an LP that is solved at each time interval k as follows:

$$\begin{aligned}
&\max_{\boldsymbol{\nu}} \mu(k) = \boldsymbol{w^T}\boldsymbol{\nu}(k) \\
&s.t.\ \boldsymbol{\mathcal{A}}\boldsymbol{\nu}(k) \le \boldsymbol{b}(\boldsymbol{z}(k-1), \boldsymbol{\beta}) \\
&\qquad \left|\frac{\boldsymbol{\nu}(k) - \boldsymbol{\nu}(k-1)}{\Delta t}\right| \le \dot{\boldsymbol{\nu}}_{max} \\
&\qquad \boldsymbol{\nu}(k), \boldsymbol{z}(k), X(k) \ge 0
\end{aligned} \tag{4.45}$$

where the vector of uncertainty parameters $\boldsymbol{\beta}$ in this model is assumed as the main source of uncertainty $\boldsymbol{d}$, Δt is the time step, k used for discretization of the mass balances. The constraints in (4.45) consist of kinetic based bounds on flux rates and positivity constraints on fluxes and metabolites' concentrations; $\boldsymbol{z}$ and X are calculated from the discretized form of the process equations as given in (4.42) – (4.44). $\boldsymbol{b}$ is formulated as a function of a particular metabolite concentration. If there is no bound for the accumulation or depletion rates of a specific metabolite, then the corresponding $\boldsymbol{b}$ for this metabolite is equal to zero. The material balances of $\boldsymbol{z}$ and X (4.42) – (4.44) are discretized using an Euler discretization scheme and the resulting discrete equations are then used for the prediction until the batch end time t_f.

Since the EMPC method is adopted in the current work, the economic objective is to maximize the amount of biomass X at the end of the fed-batch $XV(t_f)$. It is also assumed that the biomass $X(k)$ and the main metabolite, glucose, can be measured online and can be used for the purpose of feedback. Therefore, the biomass $X(t_f)$ at the end of the batch can be predicted in 2 steps: i) solution of the LP in (4.45) at every time interval to obtain the fluxes $\boldsymbol{v}(k)$; and ii) calculation of output predictions of the ODE's in (4.42) – (4.44) by using the ode45 solver in MATLAB. These two steps constitute the inner problem of the EMPC controller as discussed in (3.7) – (3.11).

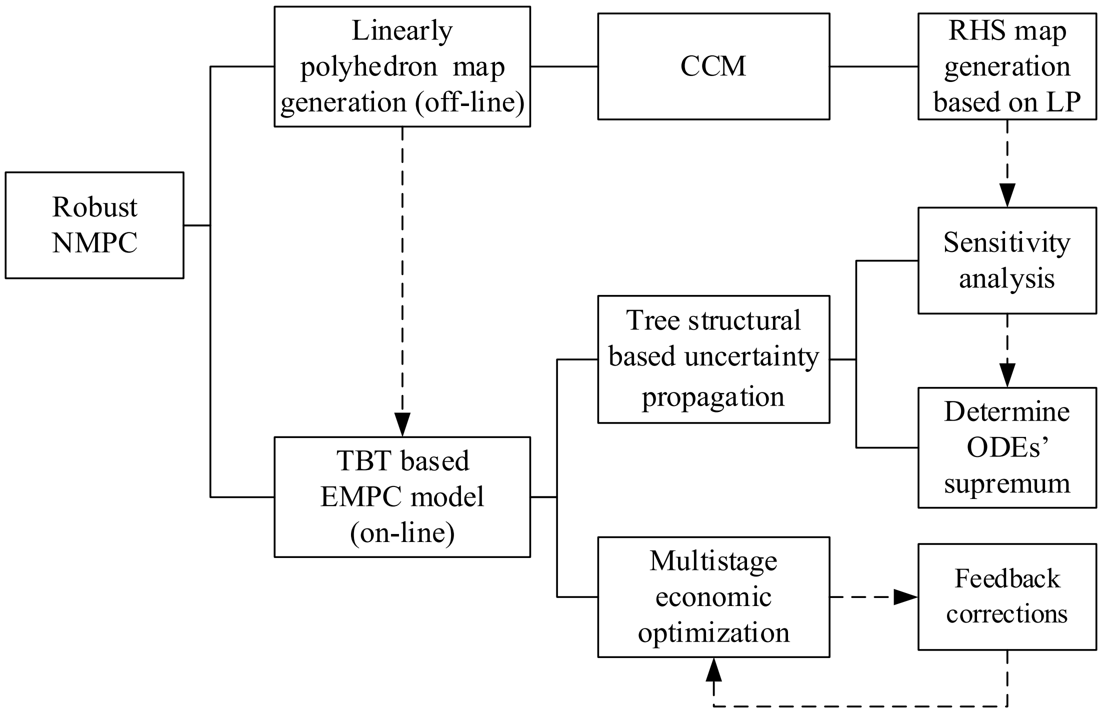
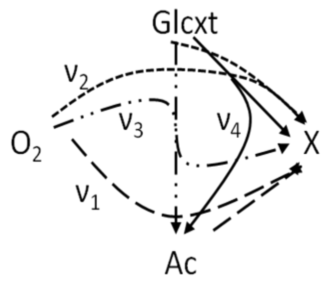

Robust NMPC
Linearly polyhedron map generation (off-line)
CCM
RHS map generation based on LP
Sensitivity analysis
Tree structural based uncertainty propagation
Determine ODEs' supremum
TBT based EMPC model (on-line)
Multistage economic optimization
Feedback corrections

Figure 4.10 The basic structure of the Robust NMPC formulation in this study

The uncertainty propagation TBT procedure is also based on the inner level problem given by (4.42) – (4.45). Accordingly, the effect of uncertain parameters $\boldsymbol{\beta}$ on the model predictions obtained with the discretized ODE's (4.42) – (4.44) coupled to the LP (4.45) is calculated using the TBT algorithm based on the following two steps: i) partition of the LP into many polyhedrons corresponding to different tableaus where for each polyhedron the maximal and minimal reaction fluxes $\boldsymbol{\nu}$ can be determined by using the sensitivity analysis of the CCM algorithm; and ii) propagation of the maximal and minimal bounds of the fluxes into corresponding the predictions of $\boldsymbol{z}$ and X as given by the discretized ODE's in (4.42) – (4.44). A schematic flowchart of the EMPC controller based on the TBT algorithm is shown in Fig. 4.10.

4.3.3 DFBM on the Growth of *E. coli*

A simplified DFBM model that is developed by Mahadevan et al., 2002 for illustrating the growth of *E. coli* on glucose is used in this work as a case study. Fig. 4.11 demonstrates the DFBM metabolic network with glucose ($Glcxt$), acetate (Ac) and oxygen (O_2) as input and the biomass (X) as the output.



ν_1	$39.43\,Ac + 35\,O_2 \rightarrow X$
ν_2	$9.46\,Glcxt + 12.92\,O_2 \rightarrow X$
ν_3	$9.84\,Glcxt + 12.73\,O_2 \rightarrow 1.24\,Ac + X$
ν_4	$19.23\,Glcxt \rightarrow 12.12\,Ac + X$

Figure 4.11 Simplified Metabolic Network for *E.coli* growth on Glucose: Flux balances and stoichiometric coefficients (Mahadevan et al., 2002)

This network contains 4 fluxes given by the vector $\boldsymbol{\nu} = [\nu_1, \nu_2, \nu_3, \nu_4]^T$ and 3 metabolites denoted by vector $\boldsymbol{z} = [z_{Gl}, z_{Ac}, z_{O_2}]$, the growth rate μ and the stoichiometric matrix $\boldsymbol{\mathcal{A}}_{\mathbf{3\times4}}$ that describes the stoichiometric relations among the 3 metabolites and the growth rate are as follows:

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} 0 & -9.46 & -9.84 & -19.23 \\ -35 & -12.92 & -12.73 & 0 \\ -39.43 & 0 & 1.24 & 12.12 \end{bmatrix}$$
$$\mu = \sum_{i=1}^{4} \nu_i \tag{4.46}$$

The dynamic mass balances of metabolites are given by (4.47) while problem (4.48) generates at each time interval the optimized fluxes $\boldsymbol{v}$ that are used in (4.47) at each instant. There are 3 distinct metabolic phases in the growth process of *E.coli* described by (4.48): i) Aerobic growth on Glucose, ii) Anaerobic growth on Glucose, and iii) Anaerobic growth on a second metabolite, acetate. These 3 phases can occur at the same time, but their occurrence is dependent on the concentration of O_2. The RHS map $\boldsymbol{M_{RHS}}$ of (4.48) is first generated with the procedure introduced in 3.2.4. Only 2 main active tableaus $\boldsymbol{MR_1}$ and $\boldsymbol{MR_2}$ were found which correspond to the basic fluxes $\{v_1, v_2\}$ or $\{v_2, v_4\}$. The detailed information of these two tableaus and the procedure and proofs are presented in Appendix A. Correspondingly, 2 different regions in RHS map, i.e. $\boldsymbol{MR_1}$ and $\boldsymbol{MR_2}$, can be considered where each region represents a distinct nutrient allocation strategy that the cell can adopt during the batch.

$$
\begin{aligned}
\frac{dz_{Gl}}{dt} &= \boldsymbol{\mathcal{A}}^{Gl}\boldsymbol{v}X + \frac{r_F\left(z_{Gl,in} - z_{Gl}\right)}{V} \\
\frac{dz_{O2}}{dt} &= \boldsymbol{\mathcal{A}}^{O2}\boldsymbol{v}X - \frac{r_F z_{O_2}}{V} + k_L a\left(0.21 - z_{O_2}\right) \\
\frac{dz_{Ac}}{dt} &= \boldsymbol{\mathcal{A}}^{Ac}\boldsymbol{v}X - \frac{r_F z_{Ac}}{V} \\
\frac{dX}{dt} &= \mu X - \frac{X(r_F - r_P)}{V} \\
\frac{dV}{dt} &= r_F - r_P
\end{aligned}
\tag{4.47}
$$

$$
\begin{aligned}
&\max_{X,v_i}\ \mu = \sum_{i=1}^{4} v_i \\
&\text{s.t.}\ \ z_i \geq 0,\ \forall i \in [1,3],\ v_i \geq 0,\ \forall i \in [1,4] \\
&\qquad |\boldsymbol{\mathcal{A}}^{Gl}\boldsymbol{v}| \leq \frac{GUR_{max}\, z_{Gl}}{K_m + z_{Gl}} \frac{mmol}{gdw - hr} \\
&\qquad -\boldsymbol{\mathcal{A}}^{O2}\boldsymbol{v} \leq OUR_{max} \\
&\qquad \boldsymbol{\mathcal{A}}^{Ac}\boldsymbol{v} \leq 100
\end{aligned}
\tag{4.48}
$$

Equations (4.47) and (4.48) are first used to formulate a nominal controller with the function proposed in 4.3.5. The uncertainty range of RHS $\boldsymbol{\beta}$ that is used in this model is assumed as in (4.26), i.e. $\boldsymbol{\beta} = \boldsymbol{d}$, where $\boldsymbol{d}$ is the uncertainty of the entire system, and thus $\boldsymbol{\beta} = \boldsymbol{\delta_d} \cdot \overline{\boldsymbol{d}}$ where the vector of nominal parameters is $\overline{\boldsymbol{d}}$. In this work, the maximum uptake rate constraints GUR_{max}, the maximum rate of oxygen uptake reaction OUR_{max} and the oxygen mass transfer coefficient $k_L a$ are chosen as the uncertain parameters. Therefore,

$$\overline{\boldsymbol{d}} = [GUR_{max} \quad OUR_{max} \quad k_L a]^{\boldsymbol{T}}$$

is the vector of nominal values of these parameters that is used in (4.26) to generate the uncertainty range $\boldsymbol{\beta}$. Finally, based on the nominal condition that have been developed and the uncertainty range $\boldsymbol{\beta}$, the robust controller can be designed based on the uncertainty propagation procedure (TBT) outlined in section 4.3.6.

4.3.4 Uncertainty Propagation

The detailed steps of the uncertainty propagation procedure for this case study are described below.

4.3.4.1 Generation of RHS Map based on the LP in (4.45) offline using the CCM in 3.2.4

The uncertainty parameter space of the LP (4.45), described by the uncertainty in the parameters of the LP model, has to be divided into a set of polyhedrons for the purpose of uncertainty propagation to be done with the TBT method. Thus, the CCM method is applied to generate the RHS map of the LP in (4.45) by following the procedure described in section 3.2.4. Since this process is often found to be computational demanding, part of the constraints that do not involve uncertainty can be eliminated from LP (4.45) during the RHS map generation process so that the computational effort is reduced. For instance, the constraints of $\boldsymbol{v}(k) \geq 0$ in LP (4.45) are not necessary in the current work. This is because the CCM method is based on the tableau analyses of the Simplex method, where the optimal solutions are forced to be positive. Also, following the outcomes established in section 3.2.5, if one of the coefficient $\overline{\boldsymbol{b}}_{\boldsymbol{i}}$ in the RHS is large enough so as the corresponding constraint will never become active during the duration of the culture, then this constraint can also be eliminated from the CCM search procedure. The generated map of RHS with respect to the LP (4.45) is denoted as $\boldsymbol{M_{RHS}}$ in this work.

4.3.4.2 Propagation of uncertainty onto fluxes (Sensitivity Analysis of CCM approach) using LP in (4.45)

Once the RHS map of LP (4.45 worst case for the LP) have been obtained, the robust NMPC based on TBT method can be formulated following the procedure outlined in 4.2.1. In this work, a scenario tree structure is generated along with the prediction horizon as illustrated in Fig. 4.12. For simplicity, most of the branches in this tree are eliminated in advance while only the nominal scenario (the green branch in Fig 4.12) and the worst case scenario (the red one in Fig 4.12) are left. Note that even though a worst-case scenario that is used in this tree structure represents the condition of the scenario that generated the worst case branch, it does not imply that only the worst case scenario is considered in this case study. Actually, this research is based on the computation of probability that the worst case of each scenario/branch may occur. The elimination can reduce the computational effort considerably while calculating the worst case of the current study. The reason of the application of this elimination is that only two main-tableaus $\boldsymbol{MR_{mc}}$ which interestingly correspond to two alternative cell strategies for its survival, i.e. either an aerobic or anaerobic respiration modes, can be active in the current work while the sub-tableaus $\boldsymbol{TR_{nc}}$ of each one of these two main-tableaus are not significantly different in terms of fluxes from the results obtained with the main tableaus. The detailed properties of the set of main tableaus $\boldsymbol{MR_{mc}}$ developed in this work is discussed in Appendix A. Thus, the RHS map is divided into 2 different polyhedrons in terms of 2 different main-tableaus' region $\boldsymbol{MR_{mc}}$. Since the production of biomass resulting from anaerobic respiration must be smaller than the rate under aerobic respiration, as long as the survival strategy of cells stay in the tableau region of the anaerobic respiration corresponding to one specific branch during the propagation process, then this branch consistently results in the minimum production of biomass which corresponds to the worst case. Thus, in this instance, the number of active scenarios $\alpha_{I,s}$ for each stage s is $\alpha_{I,1} = 1, \alpha_{I,i} = 2, i \in [2,3,\dots]$.

Therefore, the nominal branch and the worst case branch can be obtained based on (4.49) as follows:

$$\boldsymbol{S}_{\boldsymbol{z}_s^r} = \begin{bmatrix} \left(\boldsymbol{v}_{s+1}^{r(1)}\right)^T & P_{s+1}^{*r(1)} \\ \left(\boldsymbol{v}_{s+1}^{r(2)}\right)^T & P_{s+1}^{*r(2)} \end{bmatrix} = f_{CCM}(\boldsymbol{z}_s^r, X_s^r, P_s^r, u_s^r, \boldsymbol{\beta}_s^r, \boldsymbol{M_{RHS}}) \tag{4.49}$$

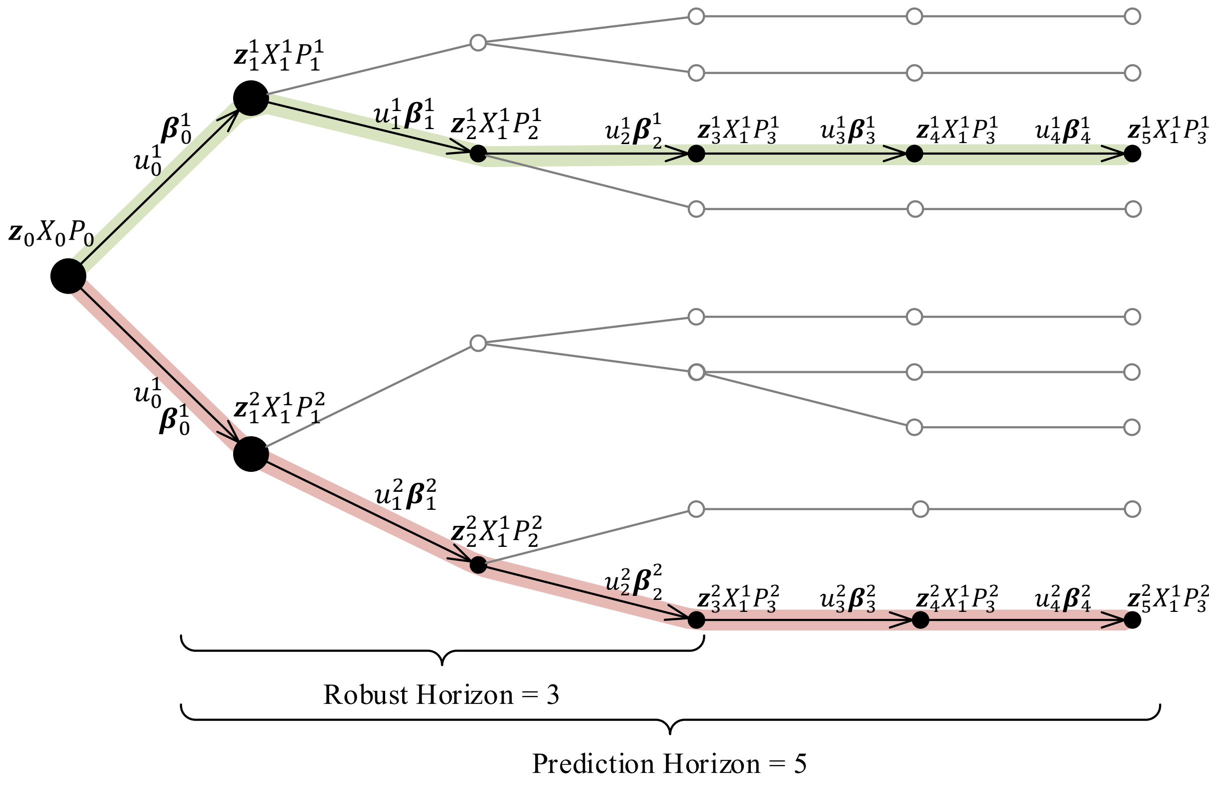
where the f_{CCM} is the function that calculates the minimal and maximal values of the fluxes as per the CCM algorithm, which have been introduced in section 3.2.5, using the *lcon2vert* algorithm in MATLAB. The elements in $\boldsymbol{z}_s^r, X_s^r$ and $\boldsymbol{\beta}_s^r$ are the corresponding maximum and minimum values as follows:

$$\boldsymbol{z}_s^r = \begin{bmatrix} \boldsymbol{z}_s^{r,min} & \boldsymbol{z}_s^{r,max} \end{bmatrix} \tag{4.50}$$

$$X_s^r = \begin{bmatrix} X_s^{r,min} & X_s^{r,max} \end{bmatrix} \tag{4.51}$$

where the uncertainty region is defined by:

$$\boldsymbol{\beta}_s^r = [\boldsymbol{\beta}_s^{r,min} \quad \boldsymbol{\beta}_s^{r,max}] \tag{4.52}$$



$\mathbf{z}_0 X_0 P_0$
u_0^1 $\boldsymbol{\beta}_0^1$
$\mathbf{z}_1^1 X_1^1 P_1^1$
$u_1^1\boldsymbol{\beta}_1^1$
$\mathbf{z}_2^1 X_1^1 P_2^1$
$u_2^1\boldsymbol{\beta}_2^1$
$\mathbf{z}_3^1 X_1^1 P_3^1$
$u_3^1\boldsymbol{\beta}_3^1$
$\mathbf{z}_4^1 X_1^1 P_3^1$
$u_4^1\boldsymbol{\beta}_4^1$
$\mathbf{z}_5^1 X_1^1 P_3^1$
u_0^1 $\boldsymbol{\beta}_0^1$
$\mathbf{z}_1^2 X_1^1 P_1^2$
$u_1^2\boldsymbol{\beta}_1^2$
$\mathbf{z}_2^2 X_1^1 P_2^2$
$u_2^2\boldsymbol{\beta}_2^2$
$\mathbf{z}_3^2 X_1^1 P_3^2$
$u_3^2\boldsymbol{\beta}_3^2$
$\mathbf{z}_4^2 X_1^1 P_3^2$
$u_4^2\boldsymbol{\beta}_4^2$
$\mathbf{z}_5^2 X_1^1 P_3^2$
Robust Horizon = 3
Prediction Horizon = 5

Figure 4.12 Uncertainty evolution with prediction horizon represented by scenario tree structure

Additionally, the $\Delta\boldsymbol{b}$ which are the variations in the RHS of the constraints of the LP problem that are used in f_{CCM} as in (3.77) with its constraints in (3.78) are obtained from (4.53). The supreme of $\boldsymbol{b}(\boldsymbol{z}_s^r, \boldsymbol{\beta}_s^r)$ can also be calculated as per equations (3.12) – (3.15) in section 3.1.

$$\Delta\boldsymbol{b} = [\boldsymbol{b}(\boldsymbol{z}_s^r, \boldsymbol{\beta}_s^r)_{\boldsymbol{min}} \quad \boldsymbol{b}(\boldsymbol{z}_s^r, \boldsymbol{\beta}_s^r)_{\boldsymbol{max}}] \tag{4.53}$$

The probability of each sub-tableau is derived by substituting (3.83) and (3.86) into (3.16):

$$P(TR_{nc}|_{\Delta b}) = \frac{Vol_{nc}|_{\Delta\boldsymbol{b}}}{Vol|_{\Delta\boldsymbol{b}}} \tag{4.54}$$

the probability of the 2 main-tableaus can be obtained as follows:

$$P_s^{*r(mc)} = \sum_{TR_{nc} \in \boldsymbol{MR_{mc}}} P(TR_{nc}|_{\Delta b}), \qquad mc \in [1,2] \tag{4.55}$$

where $mc \in [1,2]$ for current $\boldsymbol{M_{RHS}}$. Meanwhile, the optimized flux μ_s^r of biomass X_s^r can be calculated by substituting $\boldsymbol{v}_s^r$ into (4.45) as follows:

$$\mu_s^r = \boldsymbol{w^T v_s^r} \tag{4.56}$$

4.3.4.3 Propagation of uncertainty in fluxes of two scenarios in the TBT's tree structure into the predictions of $\boldsymbol{z}$ and X

As shown above, there are only two active scenarios $\boldsymbol{I}_B^{act}$ in the TBT's tree structure of this work, which are the nominal scenario $\boldsymbol{I}_B^n$ and the worst case scenario $\boldsymbol{I}_B^w$, respectively. Thus the probability P_s^w of worst case scenario $\boldsymbol{I}_B^w$ can be obtained at each stage with the functions of (4.9)–(4.14) while the probability P_s^n of the nominal scenario $\boldsymbol{I}_B^n$ can be obtained by using (4.57) for simplicity.

$$P_s^n = 1 - P_s^w \tag{4.57}$$

Correspondingly, the predictions of $\boldsymbol{z}$ and X at each stages can be obtained by using the functions (4.58)– (4.59) that were developed from (4.42)–(4.44).

$$\frac{d\boldsymbol{z}_{s+1}^r}{dt} = f_r\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) \tag{4.58}$$

$$\frac{dX_{s+1}^r}{dt} = g_r\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) \tag{4.59}$$

where the controller inputs $\boldsymbol{u}_s^r$ are the feeding rate $[r_F]_s^r$ and the perfusion rate $[r_P]_s^r$ of the corresponding stages:

$$\boldsymbol{u}_s^r = [\,[r_F]_s^r \quad [r_P]_s^r\,], \qquad \forall [r_F]_s^r \in \boldsymbol{r_F}, \qquad \forall [r_P]_s^r \in \boldsymbol{r_P} \tag{4.60}$$

Then, the vectors which elements are the control actions during the entire process are defined as follows:

$$\boldsymbol{u}_l^r = [\boldsymbol{r_F} \quad \boldsymbol{r_P}]_r, \qquad \forall \boldsymbol{u}_l^r \in \boldsymbol{U}_L \tag{4.61}$$

The functions f_r and g_r are formulated as follows:

$$f_r\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) = \begin{bmatrix} f_{min}\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) \\ f_{max}\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) \end{bmatrix}^T \tag{4.62}$$

$$g_r\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) = \begin{bmatrix} g_{min}\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) \\ g_{max}\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right) \end{bmatrix}^T \tag{4.63}$$

It is assumed that in each time interval, the maximum and minimum of function f and g can be obtained by calculating the partial derivatives or the Jacobian Matrix for each function and then using the maximum and minimum values of each parameter. For instance, within a specific time interval, if the partial derivative of f is expressed as:

$$\frac{\partial f}{\partial x_1} \leq 0, \qquad \frac{\partial f}{\partial x_2} \geq 0, \qquad \frac{\partial f}{\partial x_3} \leq 0$$

then the maximum value of this function at this instant is:

$$f_{max} = f\left(x_1^{min}, x_2^{max}, x_3^{min}\right)$$

Meanwhile, it is also assumed that the initial condition is known since it can be measured; thus, the uncertainty of the first stage is propagated from the uncertainty range of $\boldsymbol{v}_s^r$ from (4.49).

4.3.4.4 Prediction with uncertainty until the end of the batch by combining step 2 and step 3 above

At a given time interval $k,\ k \in \mathbb{N}$, when the vector of the manipulated variables $\boldsymbol{U}_L$ is available, then the feed $r_F(k+l|k)$ and perfusion $r_P(k+l|k)$ for each stage s are available, $\forall l \in \left(1,2,3,\dots,l_f\right)$. The end time t_f of the batch operation $t_f = k + \Delta l\left(l_f\right)$ is also defined as a function of the sampling intervals for each of the stages as $t_f = k + \Delta s\left(s_f\right)$, all of the different scale of time intervals, i.e. $\Delta l, \Delta s$ and Δt, are illustrated as in Fig. 4.6(b), $\tau = \{0,1,2,\dots,p\}$, $t_f = k + p$. Thus, the procedure for propagating uncertainty along the prediction horizon p is as follows:

1. Set the initial conditions, $i = 0$, $r_F(k+l|k)$ and $r_P(k+l|k)$ are known form $\boldsymbol{U}_L$, so are the $\boldsymbol{\beta}_0$, $\boldsymbol{v}_0$, $\boldsymbol{z}_0$, X_0, V_0. Let $l = 1$, $s = 0$, $\boldsymbol{v}(k-1) = \boldsymbol{v}_0$, $\boldsymbol{z}(k-1) = \boldsymbol{z}_0$, $X(k-1) = X_0$.

2. $i = i + 1$, calculate $\boldsymbol{v}_s^r(k+\tau)$, $\mu_s^r(k+\tau)$, $P_s^{*r}(k+\tau)$ and other relevant parameters by following the procedure in 4.3.4.2.

3. Determine $\boldsymbol{z}_s^r(k+\tau)$, $X_s^r(k+\tau)$, $P_s^r(k+\tau)$ using $\boldsymbol{v}_s^r(k+\tau)$, $\mu_s^r(k+\tau)$, $P_s^{*r}(k+\tau)$ from step 2 and $r_F(k+l|k)$ as well as $r_P(k+l|k)$ based on the instructions in 4.3.4.3.

4. If $\exists\alpha \in \mathbb{N}$, $\tau = \alpha\Delta l$, then $l = l + 1$, update $r_F(k+l|k)$ and $r_P(k+l|k)$ form $\boldsymbol{U}_L$; if $\exists\alpha \in \mathbb{N}$, $\tau = \alpha\Delta s$, then $s = s + 1$; if $\tau = p$, then break this uncertainty propagation procedure; else go to Step 2 and 3.

A flowchart of this section is demonstrated in Fig. 4.13.

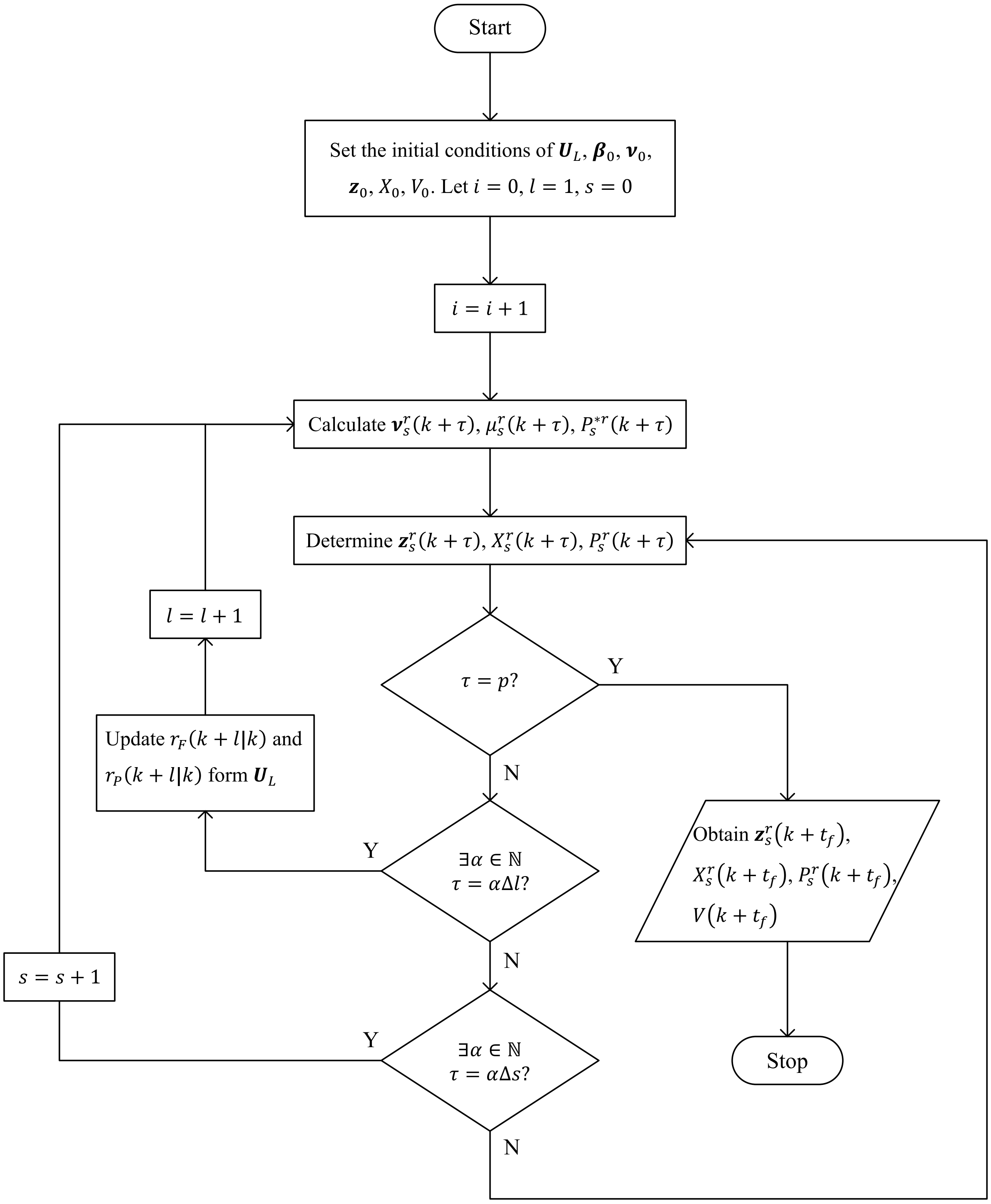

Start
Set the initial conditions of $\boldsymbol{U}_L$, $\boldsymbol{\beta}_0$, $\boldsymbol{\nu}_0$, $\boldsymbol{z}_0$, X_0, V_0. Let $i = 0$, $l = 1$, $s = 0$
$i = i + 1$
Calculate $\boldsymbol{\nu}_s^r(k+\tau)$, $\mu_s^r(k+\tau)$, $P_s^{*r}(k+\tau)$
Determine $\boldsymbol{z}_s^r(k+\tau)$, $X_s^r(k+\tau)$, $P_s^r(k+\tau)$
$\tau = p?$
Y
N
Obtain $\boldsymbol{z}_s^r(k+t_f)$, $X_s^r(k+t_f)$, $P_s^r(k+t_f)$, $V(k+t_f)$
Stop
$l = l + 1$
Update $r_F(k+l|k)$ and $r_P(k+l|k)$ form $\boldsymbol{U}_L$
Y
$\exists \alpha \in \mathbb{N}$ $\tau = \alpha \Delta l?$
N
$s = s + 1$
Y
$\exists \alpha \in \mathbb{N}$ $\tau = \alpha \Delta s?$
N

Figure 4.13 Flowchart of the procedure for propagating uncertainty along the prediction horizon $\boldsymbol{p}$

4.3.5 Nominal Controller Formulation

The nominal controller of the current study is used for comparison with the robust controller. Since the strategy of the controller here is EMPC, an economic objective function is chosen as the biomass at the end of the batch $X(t_f = k + p)V(t_f)$, which represents the amount of the biomass at the end of the batch. Meanwhile, it is assumed that the biomass $X_0 = X(k)$ and the nutrient $\boldsymbol{z}_0 = \boldsymbol{z}(k)$ can be measured as $X^{ms}(k)$ and $\boldsymbol{z}^{ms}(k)$ at the current sampling interval. The measured data is used to reinitialize the prediction procedure by using $X_0 = X^{ms}(k)$ and $\boldsymbol{z}_0 = \boldsymbol{z}^{ms}(k)$. Thus, the open-loop prediction model can be updated with the feedback: $f_{bk} = X^{ms}(k) - X(k|k-1)$ by using $X(k+p) = g(k+p) + f_{bk}$. Therefore, the nominal control problem, that is, the controller that does not consider the model uncertainty, can be formulated as in (4.64) by a two-level optimization problem. The inner level (Problem (4.45)) involves the calculation of fluxes in order to maximize growth rate at each instant. Then, the outer optimization calculates the control actions $\boldsymbol{U}_L$, i.e. the optimal feeding and perfusion rates, needed to maximize the biomass amount at the end of the batch. In the current work, the outer level is solved using *interior point* methods within *fmincon* in MATLAB, and the inner level (Problem (4.45)) is generally solved with *dual simplex method* in *linprog* in MATLAB (Kumar & Budman, 2017).

$$
\begin{aligned}
&\max_{\boldsymbol{U}_L} \; X(t_f|k)V(t_f|k) \qquad (4.64)\\
&\text{s.t.} \;\; Eqs.(4.42) - (4.45)\\
&\qquad f_{bk} = X^{ms}(k) - X(k|k-1)\\
&\qquad X(t_f|k) = g(t_f|k) + f_{bk}\\
&\qquad \boldsymbol{z}(k|k-1) = \boldsymbol{z}^{ms}(k), \qquad \exists \alpha \in \mathbb{N}, k = \alpha\Delta l\\
&\qquad X(k|k-1) = X^{ms}(k), \qquad \exists \alpha \in \mathbb{N}, k = \alpha\Delta l\\
&\qquad \boldsymbol{u_l} = [r_F(k+l|k) \quad r_P(k+l|k)\,], \qquad \forall \boldsymbol{u_l} \in \boldsymbol{U}_L
\end{aligned}
$$

4.3.6 Robust Controller Formulation

Once the uncertainty propagation procedure as described in section 4.2 has been implemented, the robust EMPC model of current work can be formulated based on the Problem (4.27) and the relevant constraints that have been discussed in section 4.2.2. To propagate uncertainty, first the offline procedure in 4.3.4.1 is used to generate the RHS map $\boldsymbol{M_{RHS}}$ as per problem (4.45). Then, the online EMPC calculations are done as per problem (4.65). Similar to the feedback corrections that are used in problem (4.64) of the nominal controller EMPC formulation, feedback corrections in the robust controller are used $f_{bk} = X^{ms}(k) - X(k|k-1)$ for updating the concentration of the biomass prediction, such as $X(k+p) = g(k+p) + f_{bk}$. Meanwhile, the current concentration of each nutrient at the root node (current time interval) are also updated by using the measurements, such as $\boldsymbol{z}_0(k|k-1) = \boldsymbol{z}^{ms}(k),\ X_0(k|k-1) = X^{ms}(k),\ \exists\alpha \in \mathbb{N}, k = \alpha\Delta l.$ Additionally, since there are only two scenarios in the tree structure of the current work, i.e. the nominal scenario and the worst case scenario, the number of active scenarios is $N_s = 2$ for problem (4.65). Finally, as for the nominal controller case, problem (4.65) is also solved using *fmincon* and *linprog* in MATLAB for solving the bi-level optimization.

$$\max_{\boldsymbol{U}_L} V_{s_f}(t_f|k)\left(\sum_{r=1}^{N_s} P_{s_f}^r X_{s_f}^{r,min}(t_f|k)\right), \qquad \forall(s_f, r) \in \boldsymbol{I}_B^{act} \tag{4.65}$$

$$\text{s.t. } Eqs. (4.49) - (4.63)$$

$$\boldsymbol{z}_s^r(\tau+1|k) = \boldsymbol{z}_s^r(\tau|k) + \Delta t\left(f_r\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right)\right) \tag{4.66}$$

$$X_s^r(\tau+1|k) = X_s^r(\tau|k) + \Delta t\left(g_r\left(\boldsymbol{\mathcal{A}}, \boldsymbol{u}_s^r, V_s^r, \boldsymbol{w}, \boldsymbol{z_{feed}}, \boldsymbol{v}_s^r, X_s^r, \boldsymbol{z}_s^r\right)\right)$$

$$\forall(s, r) \in \boldsymbol{I}_B^{act}, \qquad \tau \in [0,1,2,\dots,p]$$

$$f_{bk} = X^{ms}(k) - X_s^{r,min}(k|k-1)$$

$$X_{s_f}^{r,min}(t_f|k) = X_{s_f}^{r,min}(\tau = t_f|k) from (4.66) + f_{bk}$$

$$\boldsymbol{z}_0(k|k-1) = \boldsymbol{z}^{ms}(k), \qquad \exists\alpha \in \mathbb{N}, k = \alpha\Delta l$$

$$X_0(k|k-1) = X^{ms}(k), \qquad \exists\alpha \in \mathbb{N}, k = \alpha\Delta l$$

4.4 Results

This section aims to compare the performance of the nominal and robust optimization formulations in terms of robustness to uncertainty. In the first part (section 4.4.1), a simple case study is presented to illustrate the uncertainty propagation in a pure batch operation, i.e. without feed and perfusion, and the predictions with the nominal and robust model predictions are compared. In a second case study, 8 combinations of 2 different mean values for k_La, OUR_{max} and GUR_{max} are used to generate different disturbance conditions as in Table 4.2 and the results of the robust and nominal controller are compared in section 4.4.2. Finally, the detailed performance of one specific simulated controller in Section 4.4.2 is analyzed in Section 4.4.3.

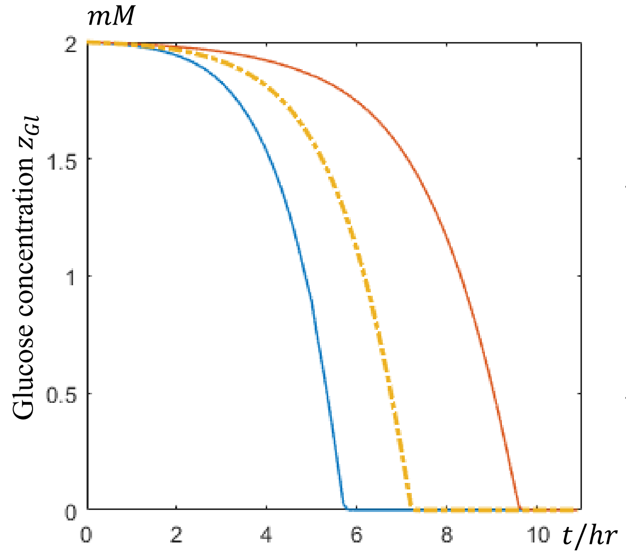
4.4.1 Comparison of Nominal and Robust Model Predictions for Pure Batch Operation

In this section, a simple case study of the uncertainty propagation based on $Eqs. (4.47)$ and the problem(4.48) is done by following the procedure illustrated in section 4.3.4. Then, the result of this propagation procedure onto the model predictions is compared with nominal predictions. In this section, feed and perfusion are not considered; similarly, the prediction of the robust model is the worst case scenario $\boldsymbol{I}_B^w$ from the start of the batch process. Also, an initially higher glucose concentration $z_{Gl,0}$ than the one presented in Table 4.1 is used in order to ensure the occurrence of all the possible phases of cell growth mentioned before. Thus, the parameter values used in the current study are as in Table 4.1.

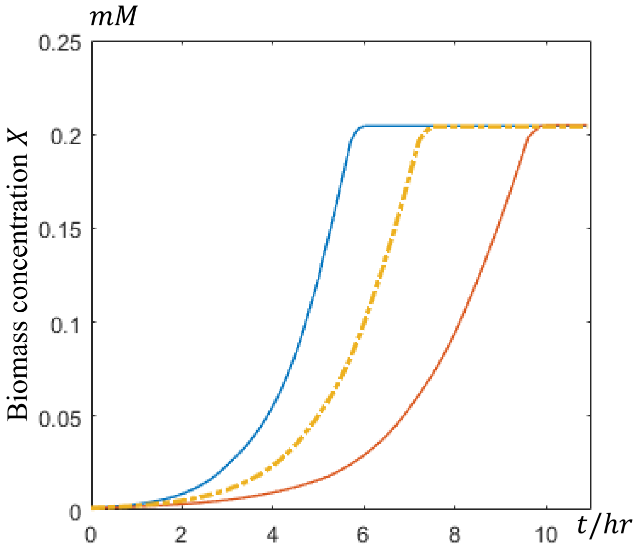
Table 4.1 Process parameters for *E. coli* growth on glucose and acetate used for uncertainty propagation

Name	Value
$k_L a$	$4\ hr^{-1}$
OUR_{max}	$12.0\ mM/g - dw/hr$
GUR_{max}	$6.5\ mM/g$
$\boldsymbol{\delta_d}$	$1 + [-0.2 \quad 0.2]$
K_m	$0.015\ mM$
$[t_f, \Delta t]$	$[10.0,0.1]\ hr$
$[s_f, \Delta s]$	$[20,0.5]\ hr$
Δl	$1.0\ hr$
$[r_F, r_P]$	$[0,0]\ L/hr$
$[z_{Gl,0}, z_{O_2,0}, z_{Ac,0}]$	$[2.0,0.21,0.20]\ mM$
$[V_{min}, V_{max}]$	$[0.2,0.4]\ L$
$[X_0, V_0]$	$[1 \times 10^{-3}\ mM, 0.3\ L]$
$[P_0, P_{min}^*, P_{min}]$	$[1,0.2,0.2]$

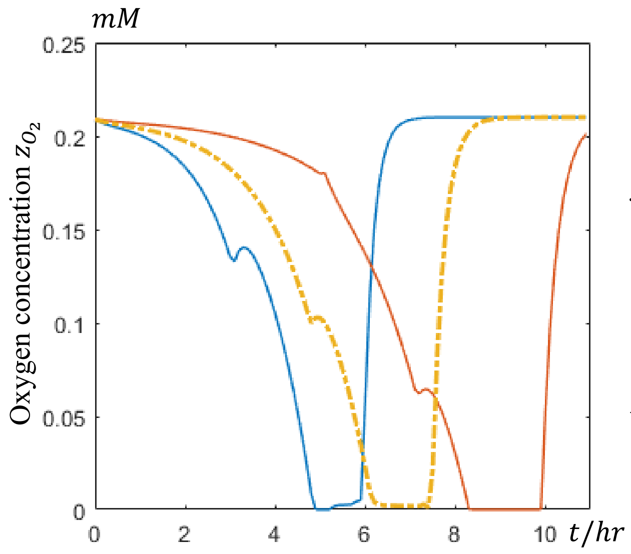
The worst case scenario for each metabolite, i.e. biomass (X), glucose ($Glcxt$), acetate (Ac) and oxygen (O_2), can be obtained from the tree structure shown in Fig. 4.14, where the difference between the red line $\psi_i^{r,min}$ and the blue line $\psi_i^{r,max}$ is due to the uncertainty.



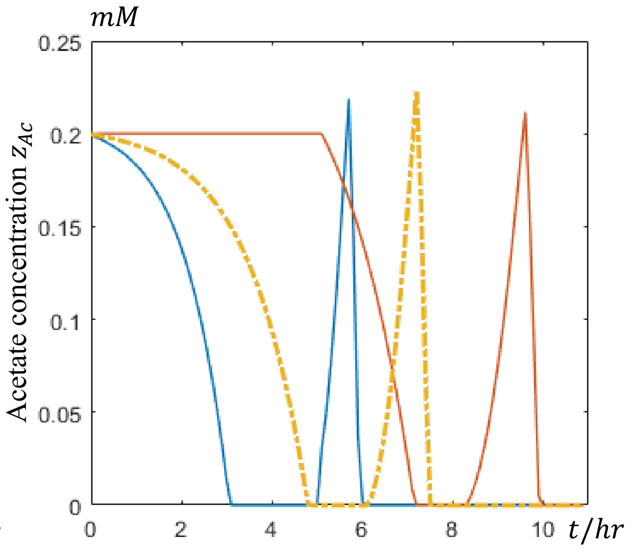
(a) Prediction of the glucose concentration z_{Gl}



(b) Prediction of the biomass concentration X



(c) Prediction of the oxygen concentration z_{O_2}



(d) Prediction of the acetate concentration z_{Ac}

Figure 4.14 Robust Prediction of Worst Case Scenario vs Nominal Prediction (blue line is the maximum bound $\psi_i^{r,max}$ of worst case scenario, red line $\psi_i^{r,min}$ is the minimum bound of the worst case scenario, the yellow dot line is the prediction of the nominal process)

It can be noticed that the prediction of the oxygen concentration z_{O_2} and the acetate concentration z_{Ac} in Fig. 4.14(c) and Fig. 4.14(d) exhibits highly nonlinear behavior while the uncertainty range is still predictable by using TBT method. The predictions of the nominal model are also shown in Fig. 4.14 by the yellow dot line. It is evident that the nominal predictions fall between the maximum and the minimum bounds of the robust model predictions. Also, it is evident that the final production of biomass is equal for both nominal and robust model predictions as expected from the mass balance since all the glucose is consumed towards biomass. These facts corroborate the ability of the TBT method to propagate the uncertainty onto the predictions while complying with the mass balances.

Note that the differences between the two branches of the robust predictions of biomass shown in Fig. 4.14(b) reflect the differences of the growth rates when reaching the maximal biomass value corresponding to the complete depletion of the initial glucose. Finally, from Fig. 4.14(c) and Fig. 4.14(d), it is obvious that at the time instant of 5 hr the minimum bound line (red line) $\psi_i^{r,min}$ has changed dramatically since at that instant for this scenario (worst case scenario), the robust horizon for the prediction of this first time interval is 5 hr. After this instant, the probability of the tableau that defines the minimum bound of this scenario is less than the minimum probability criteria $P_{min} = 0.2$. Thus, the tableau corresponding to this bound will always be set to the nominal tableau as discussed in section 4.2.1.

4.4.2 Comparison of Robust Case Studies

Two controllers are compared in this section: i- the robust EMPC controller based on equations (4.47) and the problem (4.48) using the TBT for uncertainty propagation as proposed in 4.3.5 and ii- the nominal EMPC controller based on equations (4.47) and the problem(4.48) with the formulation proposed in 4.3.6.

The nominal parameter values of the parameters that are used for the comparative case study of the robust and nominal EMPC controllers are presented in Table 4.2.

As shown in table 4.2, eight different set of parameters corresponding to the different combinations of upper and lower limits of $k_L a$, OUR_{max} and GUR_{max} are used to generate different disturbance conditions. Each of these parameters, i.e. $k_L a$, OUR_{max} and GUR_{max}, are kept constant during the run of the semi-batch operation, and the results of the robust and nominal controllers are compared in Table 4.2. Table 4.3 presents the amounts of biomass (cost) $V(t_f)X(t_f)$ that are produced at the end of the batch t_f using the robust and nominal controllers respectively.

Table 4.2 Process parameters for *E. coli* growth on glucose and acetate used for robust/nominal controller

Name	Value
$k_L a$	$4\ hr^{-1}$
OUR_{max}	$12.0\ mM/g-dw/hr$
GUR_{max}	$6.5\ mM/g$
$\boldsymbol{\delta_d}$	$1+[-0.2\quad 0.2]$
K_m	$0.015\ mM$
$[t_f, \Delta t]$	$[10.0,0.1]\ hr$
$[s_f, \Delta s]$	$[20,0.5]\ hr$
Δl	$1.0\ hr$
$[z_{Gl,in}, z_{O_2,in}, z_{Ac,in}]$	$[5.0,0,0]\ mM$
$[z_{Gl,0}, z_{O_2,0}, z_{Ac,0}]$	$[0.40,0.21,0.20]\ mM$
$[V_{min}, V_{max}]$	$[0.2,0.4]\ L$
$[X_0, V_0]$	$[1\times10^{-3}\ mM, 0.3\ L]$
$[P_0, P^*_{min}, P_{min}]$	$[1,0.2,0.2]$

From Table 4.3, it is obvious that the production of biomass is less sensitive to changes in $k_L a$ and more sensitive to changes in GUR_{max}. For this particular case study it was found that positive changes in the 3 uncertain parameters, i.e. $k_L a$, OUR_{max} and GUR_{max}, have positive effects on the biomass productivity value. Accordingly, when these 3 parameters are increasing the feeding of oxygen for tableau $\boldsymbol{MR_1}$ occurring is insufficient and thus the worst case strategy of tableau $\boldsymbol{MR_2}$ is more likely to happen. This is one of the reasons why the robust controller performance is improving when the uncertainty parameters are larger than the nominal condition, as shown in Table 4.3. Another case that supports this conclusion is that when $GUR_{max} = 9.1mM/g$ the resulting biomass productivity of the robust controller is 0.4905, which is about 4% better than that of the nominal controller (0.4733). However, the benefits of the robust controller are not significant when the uncertainty parameters are similar or slightly less than their corresponding nominal conditions, for which case the performance of the robust and nominal controllers are similar to each other. The explanation for this similarity stems from one or a combination of the following: i) the prediction interval of the nominal controller is $\Delta t = 0.1hr$ while for robust controller it is based on the intervals used for the stages $\Delta s = 0.5h$, which result in less precise prediction for the current study; ii) the nominal scenario of the TBT method in this case study was obtained by directly using the result of nominal prediction. Improvements regarding both of these items are left for future work. Nevertheless, if the uncertain parameter realizations are far lower than the nominal values, the performance of the robust controller can be also better that for the nominal controller as shown in the next subsection.

The CPU time of each controller is also presented in Table 4.3. The CPU time reported contains corresponds to the maximum time where the robust/nominal controller computes an optimized control action during one manipulating time interval (1 hour). As shown in this table, the average CPU time of the robust controller is approximately 30.4 times of the nominal controller. This results also indicates that the performance of the proposed TBT-based robust controller is more competitive than the sampling based methods, such as MC simulation and PCE, when the number of sampling points would require larger CPU times on each time interval.

Table 4.3 Robust controller and nominal controller performance

GUR_{max}	$OUR_{max} = 9.6$				OUR_{max} =14.4			
	$k_La = 3.2$		$k_La = 4.8$		$k_La = 3.2$		$k_La = 4.8$	
	Robust Cost	Nominal Cost	Robust Cost	Nominal Cost	Robust Cost	Nominal Cost	Robust Cost	Nominal Cost
5.2	0.0898	0.0900	0.0991	0.996	0.1140	0.1120	0.1276	0.1290
7.8	0.2656	0.2654	0.3085	0.3112	0.3389	0.3329	0.3868	0.3770
Ratio	1.00		0.99		1.02		1.03	
CPU Time	0.8062	0.0151	0.9737	0.0248	0.5944	0.0205	0.6080	0.0362

4.4.3 Comparison of a Specific Robust Case Study

Following the results presented in the previous section, one particular case is analyzed in the current section where the uncertainty parameters are significantly smaller than the corresponding nominal values. The 3 uncertain parameters considered here are set as $GUR_{max} = 4.6, OUR_{max} = 8.4, k_La = 2.8$ while the other parameter are as listed in Table 4.2. The productivity for the robust and nominal controllers that used the same methods as in section 4.4.2 are compared here as shown in Fig. 4.15.

As shown in Fig. 4.15(b), the terminal biomass productivity $V(t_f)X(t_f)$ of this robust controller system is 0.0527, which is about 4% better than that of the nominal controller of 0.0508. When analyzing the time trajectories, it can be seen that between time=0 to approximately $6hr$, though the feed and perfusion are slightly different at the initial several hours, the biomass growth occurs on glucose while the acetate profiles are similar for the robust and the nominal controllers. Accordingly, the nominal and robust controllers show similar performance during this time. In the time period $6hr \leq t \leq 9hr$, the concentration of acetate for both robust and nominal controllers have been exhausted while the supplementation of glucose for the robust controller is larger than for the nominal system. Since the controller action of the feeding and perfusion in the robust controller during this time interval is much more radical than for the nominal controller, especially within the period $6hr \leq t \leq 7hr$. However, the biomass growth is still similar for both controllers during this time period.

During the last phase of the batch, i.e. $9hr \leq t \leq 10hr$, the controller action differs significantly between the robust and the nominal controller, in particular with respect to the perfusion rates calculated by each of the controllers. As a result of these actions, the acetate concentration under the action of the nominal controller is growing much faster than for the robust controller. As a result of this increase, the biomass growth for the nominal controller is significantly lower than for the robust controller. However, the nominal controller cannot predict the occurrence of such worst case thoroughly, thus it cannot provide an appropriate controller action for reducing the probability for occurrence of the anaerobic process in advance. In conclusion, based on the performance comparison that have been discussed, the robust controller generally showed better final productivity (biomass level) than the nominal controller by 4% in higher disturbance or uncertainty conditions. Though it is a quite small improvement, the profit can be improved signigicantly for a high value product, a highly unstable system or a process with a more complex dynamic models.

The difference in the control actions of the robust and nominal controller in the last one hour also contributes to the difference in the production of byproducts, such as the acetate. For instance, in the last one hour, as it has been illustrated in Fig. 4.15(d), the growth of the acetate concentration using the nominal controller is significantly higher than that obtained with the robust controller.

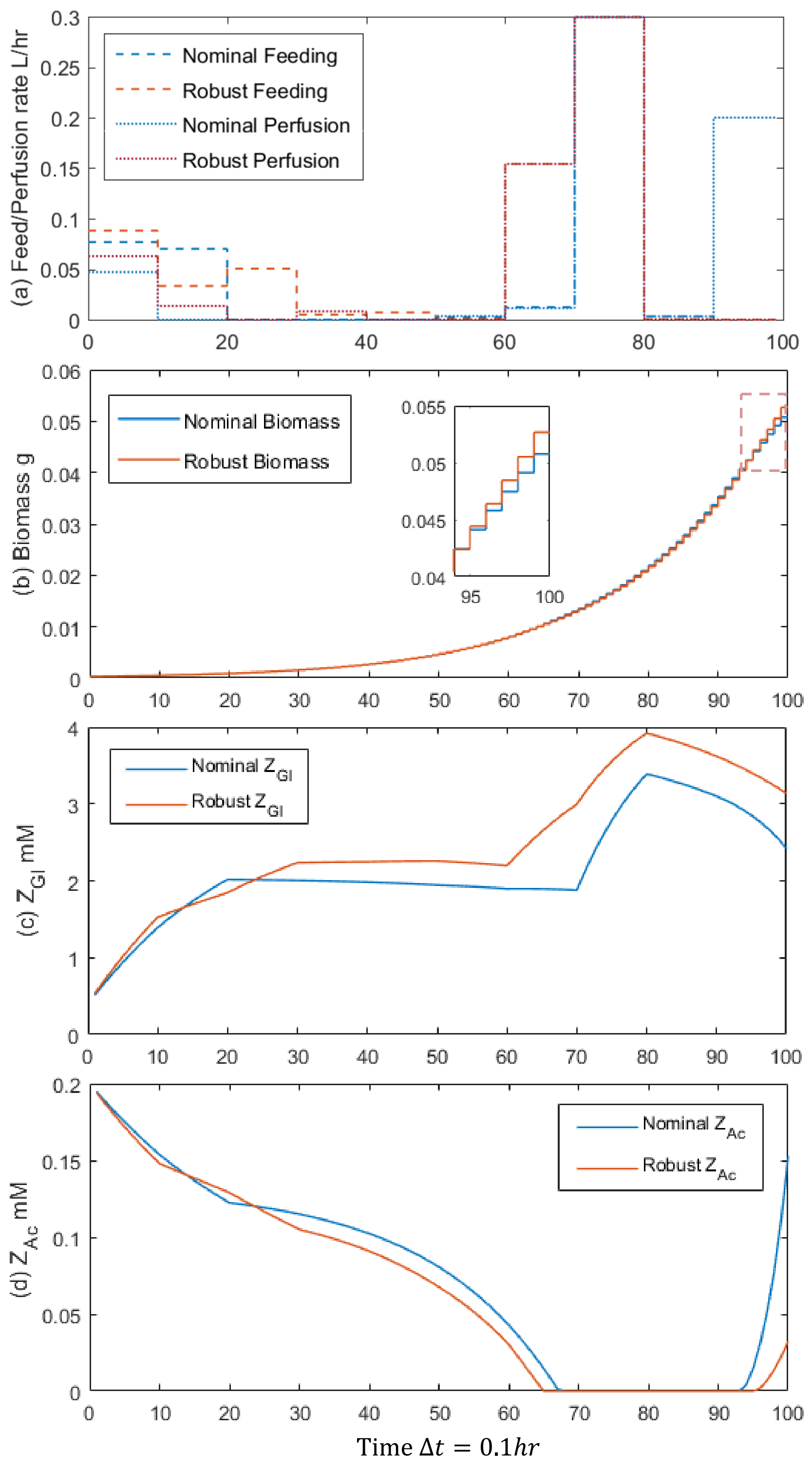
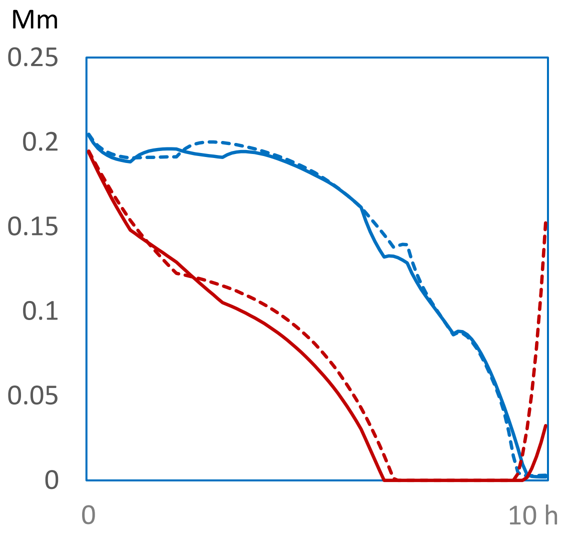
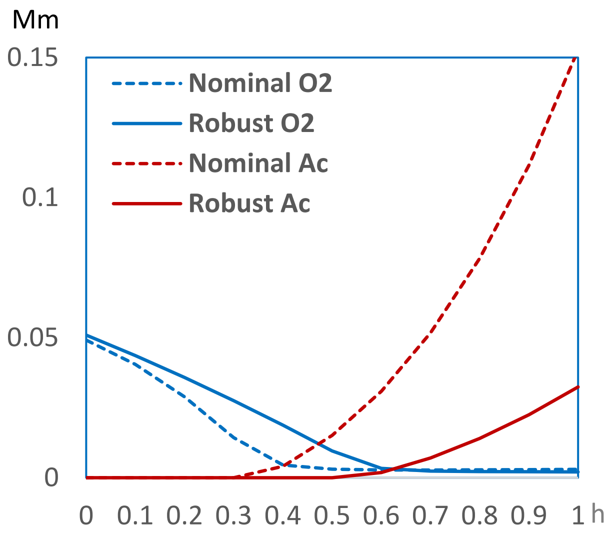


Figure 4.15 Robust vs. Nominal Controller: Feeding, Perfusion, Biomass, Glucose and Acetate



(a) The profile of Oxygen and Acetate concentration in 10 hours



(b) The profile of Oxygen and Acetate concentration in the last 1 hour

Figure 4.16 Robust vs. Nominal Controller: Oxygen and Acetate

Fig. 4.16 illustrates the oxygen and acetate profiles using the two controllers. Based on the concentration of oxygen, the production of the biomass can be divided into 2 phases, as it as been shown in Fig. 4.16(a), during the first 9 hours, cells mainly adopt aerobic growth on glucose and acetate, this is the aerobic phase with high efficiency in biomass production since byproducts is less. However, during the last one hour, when the oxygen is insufficient, the anaerobic growth on glucose will be the only choice, this is the worst case phase and the byproduct is acetate. Fig. 4.16(b) illustrated that in the last one hour, since the worst cases has already been predicted in the process of robust controller, the occurrence of the anaerobic growth was delayed significantly. Thus, less byproducts are obtained in the case of the robust controller; hence, the corresponding performance as well as the efficiency of the closed-loop process are improved considerably.

4.5 Summary

This chapter proposed a novel robust EMPC. In the current work a tree structure is used to propagate the uncertainty following different scenarios. The scenario based tree structure is based on the idea that the space described by the uncertain parameters can be sub-divided into a series of polyhedrons where the supremum can be obtained by linear calculations within each polyhedron region. Though there are many different ways to divide the uncertainty space, in the current work a novel algorithm referred to as the CCM algorithm presented in Chapter 2 has been adopted. Since the biochemical processes of the case study presented in this chapter is described by a dynamic metabolic flux model (DMFM) that is formulated by a constrained LP problem, the CCM method, which was particularly designed for this kind of applications, can be readily applied to the present case study. A tree structure multistage uncertainty propagation method is proposed that is referred to as tableaus based tree (TBT) method. Based on theoretical arguments it was shown that the TBT based propagation method is significantly more efficient than Monte Carlo or PCE method and thus it is particularly amenable for online implementation. Finally, a series of case studies were used to illustrate the proposed EMPC algorithm. It was shown that the robust controller is able to provide superior productivity as compared to the nominal controller for most cases.

Chapter 5
Conclusions and future work

5.1 Conclusions

This thesis presented a new robust EMPC algorithm based on a novel uncertainty propagation algorithm, which has been referred to as the Tableau Based Tree (TBT) method. This algorithm is shown to be less sensitive to the number of the uncertain parameters as compared to other robust algorithms that are based on sampling the uncertain parameters for uncertainty propagation. The method exploits the LP structure of the model used to describe the bioprocess considered for control.

5.1.1 CCM Algorithm in Sensitivity Analysis of RHS Space

The key goal of this study was to develop a computationally efficient robust EMPC algorithm. The computationally demanding step of any robust control algorithm is the propagation of uncertainty onto the variables involved in the control strategy.

The key idea proposed in the current work is to identify convex regions in the parameter space where each region corresponds to a Simplex tableau or part of it. It is assumed that the parameter space of the family of LP problems describing the uncertain system can be divided into a set of polyhedrons where for each polyhedron upper and lower bounds can be calculated by linear operations. In chapter 3, a series of lemmas are presented that provide the theoretical basis for the partitioning of the uncertain parameter space, i.e. the parameters used in the RHS of the constraints of the LP problem, into a finite number of polyhedrons. Since all of the polyhedrons in the RHS space can be illustrated as convex cones based on a standard simplex form of LP, the proposed partitioning method is referred to as the Convex Cone Method (CCM).

The 100 Percent Rule that has been widely used in LP problem to calculate local sensitivity of the LP solution has been found to be inefficient for uncertainty propagation in the context of robust control design. It was shown in this thesis that the 100 Percent Rule can only provide necessary but not sufficient conditions regarding the applicability of a particular Simplex tableau in the neighborhood of a nominal point but this method is not efficient for calculating all the Simplex tableaus that may exist for given ranges of uncertainty parameters' values. Beyond its ability to identify convex regions that correspond to different Simplex tableaus the CCM was also found useful for providing the probability of occurrence of each of these regions by following the procedure presented in chapter 3. These two aspects make the CCM an attractive basis for uncertainty propagation in an LP problem as compared to the 100 Percent Rule that is of limited applicability for our problem.

Though the CCM method can be adopted for multi-dimensional cases, from a series of case studies of the CCM approach, it was found that with the dimensionality of the LP problem increasing, the complexity of the problem also increases dramatically. Actually, without optimization, the time complexity of CCM algorithm off-line is approximately $O(n!)$ in the worst case, where n is the dimension of RHS. Fortunately, in most cases, this complexity mostly contributes on the CPU time of the first layer in CCM, the RHS map generation part, which has to be performed only once off-line. Although the second sensitivity analysis layer needs to be executed online, the computational effort is not as high as for the first layer. Thus, the negative impact of the time complexity in this algorithm can be relatively mitigated.

5.1.2 TBT-Based Robust EMPC

Following the identification of the convex polyhedrons by the CCM method the TBT method was developed to propagate the uncertainty in parameters onto the variables used in the control strategy.

Based on the CCM, the RHS map can be obtained where each polyhedron is referred to as tableaus or tableau regions in this thesis. With the intention of demonstrating the conditions of each tableau in any specific time instant during the process, the main idea of TBT method is introduced. In general, the TBT method is a tree structure based uncertainty propagating method where the branches correspond to each one of the convex cones calculated by the CCM algorithm. This approach is found to reduce the conservativeness of the uncertainty propagation step considerably, since not only all of the present uncertainty but also the influence of the previous uncertain parameters and control actions can be considered into the TBT formulation.

A theoretical comparison is conducted in terms of the computational demand as a function of the number of uncertain parameters. According to this comparison, it can be concluded that the TBT method have a potential to save considerable computational demand as compared to Monte Carlo simulations and PCEs. For instance, if the robust horizon is 3 with 7 uncertainty parameters and each of them have 10 different uncertainty conditions, the computational time of the Monte Carlo based method is expected to be 1×10^{18} times the computational time of the TBT method. The two main reasons that are contributing to this advantage are: the computational time in the TBT method is cubic, for a robust horizon of a limit time intervals, whereas for the computation in MC is exponential with respect to the number of uncertain parameters; the TBT method has the ability to prune branches with low probability thus saving unnecessary computations.

It is worth to note that the TBT also have its scope of application. When uncertainty parameters are less or RHS dimensions are too many, the TBT method might not such appropriate than other algorithms. For instance, the case study of the bioreactor indicates that the performance of this TBT based robust controller is more competitive than the sampling based measured (Monte Carlo simulations, PCEs, etc.) when the number of sampling points is more than 30.4 in each time interval.

In this thesis, the application of TBT-based robust EMPC controller was illustrated for a fed-batch reactor that is described by a dynamic metabolic flux model (DMFM). The robust controller showed a potential for obtaining higher final productivity (biomass level). This computational efficiency of the proposed approaches, i.e. the combination of the CCM with the TBT algorithms, strengthens the applicability of the algorithm in real time applications.

As a novel method, this TBT uncertainty propagating approach cannot avoid challenges and limitations. The first limitation is that it does not show a clear advantage in performance to other robust control methods, such as nominal controller or MC sampling method, when the number of uncertain parameters is small. Though the TBT method have been proved to be more efficient theoretically when dealing with a large number of uncertain parameters, the condition with such a large number has not been investigated in this work due to time limitations and thus a practical comparison of nominal or other robust controller methods has not been established yet. Another challenge of this method is that it is based on the assumption that the uncertain parameters are independent of each other, and the distribution of the uncertainty is assumed to be uniform. Therefore, when the full dimensional probability distribution needs to be considered with the highly correlated uncertain parameters. Accounting for such correlation might be necessary for reducing the conservatism considerably at the expense of increasing the computational costs.

5.2 Future work

The current work generated several research questions to be considered in future research for future research work, i.e.

1. **Update the formulation of CCM algorithm in order to improve efficiency:** Though the CCM algorithm proposed in this work is able to generate the entire region of RHS map and the generation process is off-line, the time required by this method as well as the proposed method for eliminating redundant constraints could be potentially reduced. Other topics that need further study are: finding bounds and make suitable decisions in case that for two tableaus the value of the cost function is equal in the overlap region, dealing with the case where the shape of the uncertainty region is not a convex polyhedron, etc.
2. **Multivariable probability distribution of the multi-dimensional output space:** In chapter 4, the probability distribution of each convex region can be calculated based on the assumption that the probability distribution of occurrence of parameters in each polyhedron of RHs space are uniform. Other more realistic probability distributions should be considered in the future.
3. **Correlation between parameters:** The uncertainty in the model parameters were assumed to be independent but in reality, they may be highly correlated. Accounting for such correlation may reduce further the conservatism of the method.
4. **Develop a method that can handle larger number of scenarios:** In the case study of current research, the amount of scenario is only 2 to simplify the computation. It is necessary to investigate the algorithm with a large number of scenarios.
5. **Application to large-scale process systems:** In this thesis, the application of the proposed algorithms is based on the fed-batch reactor. However, control has to be often applied to continuous processes thus necessitating modifications to the currently proposed algorithm. A moving horizon approach could be considered for such case.

References

Al-Gherwi, W., Budman, H., & Elkamel, A. (2011). A robust distributed model predictive control algorithm. *Journal of Process Control*, *21*(8), 1127–1137.

Allgöwer, F., Findeisen, R., & Nagy, Z. K. (2004). Nonlinear Model Predictive Control : From Theory to Application. *Chinese Institute Of*, *35*(3), 299–315.

Angeli, D., Amrit, R., J. R., U. (2009). Receding horizon cost optimization for overly constrained nonlinear plants. *In Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference* (pp. 7972-7977). IEEE.

Angeli, D., Amrit, R., & Rawlings, J. B. (2012). On average performance and stability of economic model predictive control. *IEEE transactions on automatic control*, 57(7), 1615-1626.

Banga, J. R., Alonso, A. A., & Singh, R. P. (1997). Stochastic Dynamic Optimization of Batch and Semicontinuous Bioprocesses. *Biotechnology Progress*, *13*(3), 326–335.

Berger, M., Pansu, P., Berry, J.-P., & Saint-Raymond, X. (2013). *Problems in geometry*. Springer Science & Business Media.

Birge, J. R., & Louveaux, F. (2011). *Introduction to stochastic programming*. Springer.

Boyd, S., Ghaoui, L. El, Feron, E., & Balakrishnan, V. (1994). *Linear matrix inequalities in system and control theory* (Vol. 15).

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Bradley, S. P., Hax, A. C., & Magnanti, T. L. (1977). Applied mathematical programming.

Chen, H., & Allgöwer, F. (1998). A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability. This paper was accepted for publication in revised form by Associate Editor W. Bequette under the direction of. *Automatica*, *34*(10), 1205–1217.

de la Pena, D. M., Alamo, T., Bemporad, A., & Camacho, E. F. (2006). Feedback min-max model predictive control based on a quadratic cost function. In 2006 *American Control Conference* (pp. 6-pp). IEEE.

de Oliveira, W. L., Sagastizábal, C., Penna, D. D. J., Maceira, M. E. P., & Damázio, J. M. (2010). Optimal scenario tree reduction for stochastic streamflows in power generation planning problems. *Optimization Methods and Software*, *25*(6), 917–936.

Defourny, B. (2010). *Machine Learning Solution Methods for Multistage Stochastic Programming*.

Díaz-Mendoza, R., & Budman, H. (2010). Structured Singular Valued based robust nonlinear model predictive controller using Volterra series models. *Journal of Process Control*, *20*(5), 653–663.

Diehl, M., Amrit, R., On, J. R.-I. T., & 2011, U. (2011). A Lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control*, *56(3)*, 703-707.

Diehl, M., Gerhard, J., W. M.-C. & C., & 2008. Numerical solution approaches for robust nonlinear optimal control problems. *Computers & Chemical Engineering*, *32(6)*, 1279-1292.

Doyle, F. J., Packard, A. K., & Morari, M. (1989). Robust controller design for a nonlinear CSTR. *Chemical Engineering Science*, *44*(9), 1929–1947.

Ellis, M., Liu, J., & Christofides, P. D. (2017). Advances in Industrial Control Economic Model Predictive Control Theory*, Formulations and Chemical Process Applications*.

Findeisen, R., Imsland, L., Allgower, F., & Foss, B. A. (2003). State and Output Feedback Nonlinear Model Predictive Control: An Overview. *European Journal of Control*, *9*(2–3), 190–206.

Gatley, S. L., Bates, D. G., Hayes, M. J., & Postlethwaite, I. (2002). Robustness analysis of an integrated flight and propulsion control system using μ and the ν-gap metric. *Control Engineering Practice*, *10*(3), 261–275.

Ghanem, R., & Spanos, P. D. (1990). Polynomial Chaos in Stochastic Finite Elements. *Journal of Applied Mechanics*, *57*(1), 197.

Gutierrez, G., Ricardez-Sandoval, L. A., Budman, H., & Prada, C. (2014). An MPC-based control structure selection approach for simultaneous process and control design. *Computers & Chemical Engineering*, *70*, 11–21.

He, Z., Sahraei, M. H., & Ricardez-Sandoval, L. A. (2016). Flexible operation and simultaneous scheduling and control of a CO2 capture plant using model predictive control. *International Journal of Greenhouse Gas Control*, *48*, 300–311.

Henson, M. A. (1998). Nonlinear model predictive control: current status and future directions. *Computers and Chemical Engineering*, *23*, 187–202.

Henson, M. A. (2010). Model-based control of biochemical reactors. *The Control Handbook. New York: Taylor and Francis*.

Hillier, F. (2001). *Introduction to operations research* (7th ed.). Tata McGraw-Hill Education.

Hjersted, J. L., & Henson, M. A. (2006). Optimization of Fed-Batch Saccharomyces cerevisiae Fermentation Using Dynamic Flux Balance Models. *Biotechnology Progress*, *22*(5), 1239–1248.

Hover, F. S., & Triantafyllou, M. S. (2006). Application of polynomial chaos in stability and control. *Automatica*, *42*(5), 789–795.

Høyland, K., Kaut, M., & Wallace, S. W. (2003). A Heuristic for Moment-Matching Scenario Generation. *Computational Optimization and Applications*, *24*(2/3), 169–185.

Kawohl, M., Heine, T., & King, R. (2007). A new approach for robust model predictive control of biological production processes. *Chemical Engineering Science*, *62*(18–20), 5212–5215.

Kothare, M. V., Balakrishnan, V., & Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, *32*(10), 1361–1379.

Kuhlmann, C., Bogle, I. D. L., & Chalabi, Z. S. (1998). Robust operation of fed batch fermenters. *Bioprocess Engineering*, *19*(1), 53.

Kumar, D., & Budman, H. (2017). Applications of Polynomial Chaos Expansions in optimization and control of bioreactors based on dynamic metabolic flux balance models. *Chemical Engineering Science*, *167*, 18–28.

Lindhorst, H., Lucia, S., Findeisen, R., & Waldherr, S. (2016). Modeling metabolic networks including gene expression and uncertainties. *arXiv preprint arXiv:1609.08961*.

Lübbert, A., & Jørgensen, S. (2001). Bioreactor performance: a more scientific approach for practice. *Journal of Biotechnology, 85(2)*, 187-212.

Lucia, S., Andersson, J. A. E., Brandt, H., Diehl, M., & Engell, S. (2014). Handling uncertainty in economic nonlinear model predictive control: A comparative case study. *Journal of Process Control*, *24*(8), 1247–1259.

Lucia, S., Finkler, T., & Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, *23*, 1306–1319.

Ma, D. L., & Braatz, R. D. (2001). Worst-case analysis of finite-time control policies. *IEEE Transactions on Control Systems Technology*, *9*(5), 766–774.

Ma, D. L., Chung, S. H., & Braatz, R. D. (1999). Worst-case performance analysis of optimal batch control trajectories. *AIChE Journal*, *45*(7), 1469–1476.

Ma, J., Qin, J., Salsbury, T., & Xu, P. (2011). Demand reduction in building energy systems based on economic model predictive control. *Chemical Engineering Science*, *67*, 92–100.

Magni, L., Raimondo, D. M., & Scattolini, R. (2006). Regional Input-to-State Stability for Nonlinear Model Predictive Control. *IEEE Transactions on Automatic Control*, *51*(9), 1548–1553.

Magni, L., & Scattolini, R. (2010). An Overview of Nonlinear Model Predictive Control. In L. del Re, F. Allgöwer, L. Glielmo, C. Guardiola, & I. Kolmanovsky (Eds.), *Automotive Model Predictive Control: Models, Methods and Applications* (pp. 107–117). London: Springer London.

Mahadevan, R., Edwards, J. S., & Doyle, F. J. (2002). Dynamic Flux Balance Analysis of Diauxic Growth in Escherichia coli. *Biophysical Journal*, *83*(3), 1331–1340.

Mandur, J., & Budman, H. (2012). A Polynomial-Chaos based Algorithm for Robust optimization in the presence of Bayesian Uncertainty. *IFAC Proceedings Volumes*, *45*(15), 549–554.

Matt, J. (2017). Analyze N-dimensional Polyhedra in terms of Vertices or (In)Equalities. Accessed 21 Dec 2016. *http://www.mathworks.com/matlabcentral/fileexchange/30892-representing-polyhedral-convex-hulls-by-vertices-or--in-equalities/content/vert2lcon.m*.

Mayne, D. Q., Kerrigan, E. C., van Wyk, E. J., & Falugi, P. (2011). Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, *21*(11), 1341–1353.

Mehta, S., & Ricardez-Sandoval, L. A. (2016). Integration of Design and Control of Dynamic Systems under Uncertainty: A New Back-Off Approach. *Industrial & Engineering Chemistry Research*, *55*(2), 485–498.

Michalska, H., & Mayne, D. Q. (1993). Robust Receding Horizon Control of Constrained Nonlinear Systems. *Ieee Transactions on Automatic Control*, *38*(I), 1623–1633.

Nagy, Z. K., & Braatz, R. D. (2003). Robust nonlinear model predictive control of batch processes. *AIChE Journal*, *49*(7), 1776–1786.

Niederreiter, H. (1978). Quasi-Monte Carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, *84*(6), 957–1042.

Patil, B. P., Maia, E., & Ricardez-Sandoval, L. A. (2015). Integration of scheduling, design, and control of multiproduct chemical processes under uncertainty. *AIChE Journal*, *61*(8), 2456–2470.

Poole, D. (2014). *Linear algebra: A modern introduction*. Cengage Learning.

Rasoulian, S., & Ricardez-Sandoval, L. A. (2015). A robust nonlinear model predictive controller for a multiscale thin film deposition process. *Chemical Engineering Science*, *136*, 38–49.

Rasoulian, S., & Ricardez-Sandoval, L. A. (2016). Stochastic nonlinear model predictive control applied to a thin film deposition process under uncertainty. *Chemical Engineering Science*, *140*, 90–103.

Rawlings, J. B., Angeli, D., & Bates, C. N. (2012). *Fundamentals of Economic Model Predictive Control.*

Santander, O., Elkamel, A., & Budman, H. (2016). Economic model predictive control of chemical processes with parameter uncertainty. *Computers & Chemical Engineering*, *95*, 10–20.

Shapiro, A. (2003). Monte Carlo Sampling Methods. *Handbooks in Operations Research and Management Science*, *10*, 353–425.

Smith, A., Monti, A., & Ponci, F. (2009). Uncertainty and Worst-Case Analysis in Electrical Measurements Using Polynomial Chaos Theory. *IEEE Transactions on Instrumentation and Measurement*, *58*(1), 58–67.

Srinivasan, B., Bonvin, D., Visser, E., & Palanki, S. (2003). Dynamic optimization of batch processes: II. Role of measurements in handling uncertainty. *Computers & Chemical Engineering*, *27*(1), 27–44.

Vanantwerp, J. G., & Braatz, R. D. (2000). Tutorial on linear and bilinear matrix inequalities. *Journal of Process Control*, *10*(4), 363–385.

Varma, A., & Palsson, B. O. (1994). Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. *Bio/Technology*, *12*(10), 994–998.

Yamuna Rani, K., & Ramachandra Rao, V. S. (1999). Control of fermenters – a review. *Bioprocess Engineering*, *21*(1), 77–88.

Appendix A
Supplementary information for Chapter 4

The RHS map generated for the problem (4.48) is listed in this section. For the LP form of the problem (4.48), the main parameters such as $\boldsymbol{A}, \boldsymbol{c}$ and $\boldsymbol{b}$ are listed below:

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

$$\boldsymbol{c} = [1 \quad 1 \quad 1 \quad 1]$$

$$b_1 = \frac{\boldsymbol{GUR_{max}}\, \boldsymbol{Z_{Glcxt}}}{\boldsymbol{K_m} + \boldsymbol{Z_{Glcxt}}}$$

$$b_2 = \boldsymbol{OUR_{max}}$$

$$b_3 = 100$$

$$b_4 = \boldsymbol{X}^{-1}\left(\frac{r_F\left(\boldsymbol{z_{Gl,in}} - \boldsymbol{z_{Gl}}\right)}{V} + \frac{\boldsymbol{z_{Gl}}}{dt}\right)$$

$$b_5 = \boldsymbol{X}^{-1}\left(-\frac{r_F \boldsymbol{z_{O_2}}}{V} + \boldsymbol{k_L a}\left(0.21 - \boldsymbol{z_{O_2}}\right) + \frac{\boldsymbol{z_{O_2}}}{dt}\right)$$

$$b_6 = \boldsymbol{X}^{-1}\left(-\frac{r_F \boldsymbol{z_{Ac}}}{V} + \frac{\boldsymbol{z_{Ac}}}{dt}\right)$$

In most cases, the constraint corresponding to b_3, b_4, b_5 and b_6 are too large to be active; thus, when all these four parameters are larger than 50, only b_1 and b_2 are considered to generate the RHS map in order to improve the efficiency of the sensitivity analysis procedure. In this case, the RHS map is 2 dimensional which is referred as 2D RHS map for short. When any one of the b_4, b_5 and b_6 are smaller than 50, then all of these 3 parameters are also considered in the RHS map generation procedure, the map is referred to as 5D case. The map for 2D case is listed in Table A.1 for $\boldsymbol{MR_1}$ and Table A.2 for $\boldsymbol{MR_2}$, which correspond to the basic fluxes $\{v_1, v_2\}$ or $\{v_2, v_4\}$ respectively. Similarly, the map for 5D case is also able to maintain these two tableaus which are listed Table A.3 and Table A.4, respectively.

Table A.1 Main tableau MR_1 $\{v_1, v_2\}$ for 2D RHS map

Sub-tableaus	Convex cones	Main properties	coefficients		
				α_1	α_2
[1,2]	1	Edge constrains	e_1	-1.0000	0.0000
			e_2	1.0000	-0.7322
		Basic solutions	v_1	-0.0390	0.0286
			v_2	0.1057	0.0000
			v_3	0.0000	0.0000
			v_4	0.0000	0.0000

Table A.2 Main tableau MR_2 $\{v_2, v_4\}$ for 2D RHS map

Sub-tableaus	Convex cones	Main properties	coefficients		
				α_1	α_2
[2,4]	1	Edge constrains	e_1	-1.0000	0.7322
			e_2	1.0000	-0.7730
		Basic solutions	v_1	0.0000	0.0000
			v_2	0.0000	0.0774
			v_3	0.0000	0.0000
			v_4	0.0520	-0.0381
	2	Edge constrains	e_1	0.0000	-1.0000
			e_2	-1.0000	0.7730
		Basic solutions	v_1	0.0000	0.0000
			v_2	0.0000	0.0774
			v_3	0.0000	0.0000
			v_4	0.0520	-0.0381

As shown in Table A.1. and Table A.2., there are only two main-tableaus $\boldsymbol{MR_{mc}}$ which correspond to two alternative cell strategies for its survival, i.e. either an aerobic or anaerobic respiration modes, while the sub-tableaus $\boldsymbol{TR_{nc}}$ of each one of these two main-tableaus are not significantly different in terms of fluxes, i.e. the $\boldsymbol{\alpha}$ matrix of basic solutions, from the results obtained with the main tableaus.

Table A.3 Main tableau MR_1 $\{\boldsymbol{\nu_1}, \boldsymbol{\nu_2}\}$ for 5D RHS map

Sub-tableau	[1 2 5 6 8]	
Convex cones	Edge constrains (Matrix $\boldsymbol{\alpha}$)	Basic solutions (Matrix $\boldsymbol{\alpha}$)

1	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 1.3658 -0.0000 0.8876
	0.0000 1.0000 -1.4056 0.0000 -0.8876
	0.0000 1.0000 0.0000 -1.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0254
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

2	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -0.0000 1.0000 -0.7322 0.6499
	0.0000 1.0000 -1.4056 0.0000 -0.8876
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
		0.0000 0.0000 0.0000 0.0000 0.0254
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

3	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -0.0000 1.0000 -0.7322 0.6499
	-0.0000 -1.0000 1.4056 -0.0000 0.8876
	0.0000 0.0000 -1.0000 0.7115 -0.6315
		0.0000 0.0000 0.0000 0.0000 0.0254
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

4	-0.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 1.4056 -0.0000 0.8876
	-0.0000 -0.0000 1.0000 -0.7115 0.6315
		0.0000 0.0000 0.0000 0.0000 0.0254
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

Sub-tableau	[1 2 5 6 9]	

1	-0.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 -0.0000 1.0000 -0.7322 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	0.0000 0.0000 0.0000 1.0000 -0.8876
		0.0000 0.0000 -0.0390 0.0286 0.0000
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

2	-0.0000 -0.0000 -0.0000 -1.0000 0.8876
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	-0.0000 -0.0000 1.0000 -0.7322 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	0.0000 0.0000 -1.0000 1.7875 -1.5866
		0.0000 0.0000 -0.0390 0.0286 0.0000
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

3	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	-0.0000 -0.0000 1.0000 -0.7322 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -0.0000 1.0000 -1.7875 1.5866
	0.0000 0.0000 -1.0000 0.7322 -0.6499
		0.0000 0.0000 -0.0390 0.0286 0.0000
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

Sub-tableau	[1 2 5 8 9]	

1	-0.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 1.3658 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	0.0000 1.0000 0.0000 0.0000 -0.8876
		0.0000 0.0286 -0.0390 0.0000 0.0000
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

2	-0.0000 -1.0000 -0.0000 -0.0000 0.8876
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-0.0000 -1.0000 1.3658 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	0.0000 1.0000 -0.5595 0.0000 -0.8876
		0.0000 0.0286 -0.0390 0.0000 0.0000
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

3	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-0.0000 -1.0000 1.3658 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 0.5595 -0.0000 0.8876
	0.0000 1.0000 -1.3658 0.0000 -0.8876
		0.0000 0.0286 -0.0390 0.0000 0.0000
		0.0000 0.0000 0.1057 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

Sub-tableau	[1 2 6 7 8]	

1	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	1.0000 -0.7322 -0.0000 -0.0000 0.6499
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-1.0000 0.7115 0.0000 0.0000 -0.6315
	0.0000 1.0000 0.0000 -1.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0254
		0.1057 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

2	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000 1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0254 0.1057 0.0000 0.0000 0.0000 0.0000

	1.0000 -0.0000 -0.0000 -0.7322 0.6499	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 0.7115 0.0000 0.0000 -0.6315	0.0000 0.0000 0.0000 0.0000 0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	
3	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000	0.0000 0.0000 0.0000 0.0000 0.0254
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -0.0000 -0.7322 0.6499	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.7115 -0.0000 -0.0000 0.6315	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 0.0000 0.0000 0.7115 -0.6315	
4	-1.0000 -0.0000 -0.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0254
	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.7115 -0.0000 -0.0000 0.6315	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -0.0000 -0.7115 0.6315	
Sub-tableau	[1 2 6 7 9]	
1	-1.0000 -0.0000 -0.0000 -0.0000 -0.0000	-0.0390 0.0000 0.0000 0.0286 0.0000
	1.0000 -0.0000 -0.0000 -0.7322 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	0.0000 0.0000 0.0000 1.0000 -0.8876	
2	-0.0000 -0.0000 -0.0000 -1.0000 0.8876	-0.0390 0.0000 0.0000 0.0286 0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -0.0000 -0.7322 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 0.0000 0.0000 1.7875 -1.5866	
3	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	-0.0390 0.0000 0.0000 0.0286 0.0000
	1.0000 -0.0000 -0.0000 -0.7322 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -0.0000 -1.7875 1.5866	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 0.0000 0.0000 0.7322 -0.6499	
Sub-tableau	[1 2 7 8 9]	
1	-1.0000 -0.0000 -0.0000 -0.0000 -0.0000	-0.0390 0.0286 0.0000 0.0000 0.0000
	1.0000 -0.7322 -0.0000 -0.0000 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	0.0000 1.0000 0.0000 0.0000 -0.8876	
2	-0.0000 -1.0000 -0.0000 -0.0000 0.8876	-0.0390 0.0286 0.0000 0.0000 0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.7322 -0.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 1.7875 0.0000 0.0000 -1.5866	
3	-0.0000 1.0000 -0.0000 -1.0000 -0.0000	-0.0390 0.0286 0.0000 0.0000 0.0000
	1.0000 -0.7322 -0.0000 -0.0000 -0.0000	0.1057 0.0000 0.0000 0.0000 0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	1.0000 -1.7875 -0.0000 -0.0000 1.5866	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 0.7322 0.0000 0.0000 -0.6499	

Table A.4 Main tableau MR_2 $\{\nu_2, \nu_4\}$ for 5D RHS map

Sub-tableau	[2 4 5 6 9]	
Convex cones	Edge constrains (Matrix $\boldsymbol{\alpha}$)	Basic solutions (Matrix $\boldsymbol{\alpha}$)
1	-0.0000 -0.0000 -1.0000 0.7322 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0774 0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	0.0000 0.0000 1.0000 -0.7730 0.0000	0.0000 0.0000 0.0520 -0.0381 0.0000
	0.0000 0.0000 0.0000 1.0000 -0.8876	
2	-0.0000 -0.0000 -1.0000 0.7322 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0774 0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	0.0000 0.0000 1.0000 -0.7730 0.0000	0.0000 0.0000 0.0520 -0.0381 0.0000
	-0.0000 -0.0000 -0.0000 -1.0000 0.8876	
	0.0000 0.0000 -1.0000 1.7875 -1.5866	
3	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-0.0000 -0.0000 -1.0000 0.7322 -0.0000	0.0000 0.0000 0.0000 0.0774 0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000	0.0000 0.0000 0.0520 -0.0381 0.0000
	0.0000 0.0000 1.0000 -0.7730 0.0000	
	-0.0000 -0.0000 1.0000 -1.7875 1.5866	
4	-0.0000 -0.0000 -0.0000 -1.0000 -0.0000	0.0000 0.0000 0.0000 0.0000 0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000	0.0000 0.0000 0.0000 0.0774 0.0000

	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	-0.0000 -0.0000 -1.0000 0.7730 -0.0000
	0.0000 0.0000 0.0000 1.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0520 -0.0381 0.0000

5	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	-0.0000 -0.0000 -1.0000 0.7730 -0.0000
	-0.0000 -0.0000 -0.0000 -1.0000 0.8876
	0.0000 0.0000 -1.0000 1.7875 -1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0520 -0.0381 0.0000

6	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	-0.0000 -0.0000 -1.0000 0.7730 -0.0000
	-0.0000 -0.0000 1.0000 -1.7875 1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0520 -0.0381 0.0000

Sub-tableau	[2 4 5 8 9]	

1	-0.0000 -1.0000 -0.0000 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	0.0000 1.0000 -1.2937 0.0000 0.0000
	0.0000 1.0000 0.0000 0.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 -0.0381 0.0520 0.0000 0.0000

2	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	0.0000 1.0000 -1.2937 0.0000 0.0000
	-0.0000 -1.0000 -0.0000 -0.0000 0.8876
	0.0000 1.0000 -0.5595 0.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 -0.0381 0.0520 0.0000 0.0000

3	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	0.0000 1.0000 -1.2937 0.0000 0.0000
	-0.0000 -1.0000 0.5595 -0.0000 0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 -0.0381 0.0520 0.0000 0.0000

4	-0.0000 1.0000 -1.3658 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-0.0000 -1.0000 1.2937 -0.0000 -0.0000
	0.0000 1.0000 0.0000 0.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 -0.0381 0.0520 0.0000 0.0000

5	-0.0000 1.0000 -1.3658 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-0.0000 -1.0000 1.2937 -0.0000 -0.0000
	-0.0000 -1.0000 -0.0000 -0.0000 0.8876
	0.0000 1.0000 -0.5595 0.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 -0.0381 0.0520 0.0000 0.0000

6	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-0.0000 1.0000 -1.3658 -0.0000 -0.0000
	-1.0000 -0.0000 1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-0.0000 -1.0000 1.2937 -0.0000 -0.0000
	-0.0000 -1.0000 0.5595 -0.0000 0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 -0.0381 0.0520 0.0000 0.0000

Sub-tableau	[2 4 6 7 9]	

1	-1.0000 -0.0000 -0.0000 0.7322 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	1.0000 0.0000 0.0000 -0.7730 0.0000
	0.0000 0.0000 0.0000 1.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 0.0000 0.0000 -0.0381 0.0000

2	-1.0000 -0.0000 -0.0000 0.7322 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	1.0000 0.0000 0.0000 -0.7730 0.0000
	-0.0000 -0.0000 -0.0000 -1.0000 0.8876
	-1.0000 0.0000 0.0000 1.7875 -1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 0.0000 0.0000 -0.0381 0.0000

3	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 -0.0000 -0.0000 0.7322 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	1.0000 0.0000 0.0000 -0.7730 0.0000
	1.0000 -0.0000 -0.0000 -1.7875 1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 0.0000 0.0000 -0.0381 0.0000

4	-0.0000 -0.0000 -0.0000 -1.0000 -0.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000

	-1.0000 -0.0000 -0.0000 0.7730 -0.0000 0.0000 0.0000 0.0000 1.0000 -0.8876	0.0520 0.0000 0.0000 -0.0381 0.0000

5	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-1.0000 -0.0000 -0.0000 0.7730 -0.0000
	-0.0000 -0.0000 -0.0000 -1.0000 0.8876
	-1.0000 0.0000 0.0000 1.7875 -1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 0.0000 0.0000 -0.0381 0.0000

6	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-0.0000 -1.0000 -0.0000 1.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-1.0000 -0.0000 -0.0000 0.7730 -0.0000
	1.0000 -0.0000 -0.0000 -1.7875 1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0774 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 0.0000 0.0000 -0.0381 0.0000

Sub-tableau	[2 4 7 8 9]	

1	-1.0000 0.7322 -0.0000 -0.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	1.0000 -0.7730 0.0000 0.0000 0.0000
	0.0000 1.0000 0.0000 0.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 -0.0381 0.0000 0.0000 0.0000

2	-1.0000 0.7322 -0.0000 -0.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	1.0000 -0.7730 0.0000 0.0000 0.0000
	-0.0000 -1.0000 -0.0000 -0.0000 0.8876
	-1.0000 1.7875 0.0000 0.0000 -1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 -0.0381 0.0000 0.0000 0.0000

3	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	-1.0000 0.7322 -0.0000 -0.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	1.0000 -0.7730 0.0000 0.0000 0.0000
	1.0000 -1.7875 -0.0000 -0.0000 1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 -0.0381 0.0000 0.0000 0.0000

4	-0.0000 -1.0000 -0.0000 -0.0000 -0.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-1.0000 0.7730 -0.0000 -0.0000 -0.0000
	0.0000 1.0000 0.0000 0.0000 -0.8876
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 -0.0381 0.0000 0.0000 0.0000

5	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-1.0000 0.7730 -0.0000 -0.0000 -0.0000
	-0.0000 -1.0000 -0.0000 -0.0000 0.8876
	-1.0000 1.7875 0.0000 0.0000 -1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 -0.0381 0.0000 0.0000 0.0000

6	-0.0000 -0.0000 -0.0000 -0.0000 -1.0000
	1.0000 -0.0000 -1.0000 -0.0000 -0.0000
	-0.0000 1.0000 -0.0000 -1.0000 -0.0000
	-1.0000 0.7730 -0.0000 -0.0000 -0.0000
	1.0000 -1.7875 -0.0000 -0.0000 1.5866
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0000 0.0774 0.0000 0.0000 0.0000
		0.0000 0.0000 0.0000 0.0000 0.0000
		0.0520 -0.0381 0.0000 0.0000 0.0000

Appendix B
MATLAB Codes

Main codes that designed for Chapter 3 CCM algorithm

```
classdef AtabDistribution
    properties
        Tab_type = 0;
        Basic_tab = [];
        List = [];
        List_edge = [];
        List_temp = [];
        Possible = 0;
        MaxMin = [];
    end

    methods
        function obj = AtabDistribution(tab_type, basic_tab, list, list_edge,
list_temp, possible, maxmin)
            % Tab_type, Basic_tab, List, List_edge, List_temp, Possible, MaxMin
            obj.Tab_type = tab_type;
            obj.Basic_tab = basic_tab;
            obj.List = list;
            obj.List_edge = list_edge;
            obj.List_temp = list_temp;
            obj.Possible = possible;
            obj.MaxMin = maxmin;
        end

    end
end


classdef branchList
    % tree list
    properties
        Time = 0;
        BranchSubtab = [];
        Pos_list = []; % possiblility of different line
        Scenario = [];
        Wholerange = [];
    end

    methods
        function list = branchList(time, branchsub, pos_list, scenario, wholerange)
            % time, branchsub, pos_list, scenario
            list.Time = time;
            list.BranchSubtab = branchsub;
            list.Scenario = scenario;
            list.Pos_list = pos_list;
            list.Wholerange = wholerange;
        end

    end % methods

end % classdef
```

```
classdef branchSubtab

    properties
        Prev = []; %[ subtab#]
        Mid = [];
        Next = [];
        Time = 0;
        Up_leaf = [];
        Mid_leaf = [];
        Low_leaf = [];
        List_conbine = [];
        Possible = 0;
        X_grad = []; % grad of biomass
        Prob_dis = []; % Probalitity distribution
        Unit = [];
        Nodes = [];
        MaxMin_nu_mu = [];
        MaxMin_y = [];
        Data
    end

    methods
        function node = branchSubtab(time, up_leaf, mid_leaf, low_leaf, list_conbine, 
possible, x_grad, prob_dis, unit, nodes, maxmin_nu_mu, maxmin_y, data)
            % time, up_leaf, mid_leaf, low_leaf, list_conbine, possible, x_grad, 
prob_dis, unit, nodes, maxmin_nu_mu, maxmin_y, data
            % DLNODE  Constructs a node object.
            node.Time = time;
            node.Up_leaf = up_leaf;
            node.Mid_leaf = mid_leaf;
            node.Low_leaf = low_leaf;
            node.List_conbine = list_conbine;
            node.Possible = possible;
            node.X_grad = x_grad;
            node.Prob_dis = prob_dis;
            node.Unit = unit;
            node.Nodes = nodes;
            node.MaxMin_nu_mu = maxmin_nu_mu;
            node.MaxMin_y = maxmin_y;
            node.Data = data;
        end



    end % methods
end % classdef




classdef Leaves
    % tree list
    properties
        Time = 0;
        Num = 0; % 1-up, 2-mid, 3-low
        Prev = []; %[ subtab#]
        Lenu = []; % unitlength
        Possible = 0;
        Br_node_list = [];
        MaxMin_nu_mu = [];
        MaxMin_y = [];
    end
    methods
        function list = Leaves(time, num, prev, lenu, possible, br_node_list, 
maxmin_nu_mu, maxmin_y)
            % time, num, prev, lenu, possible, br_node_list, maxmin_nu_mu, maxmin_y
            list.Time = time;
            list.Num = num;
            list.Prev = prev;
            list.Lenu = lenu;
            list.Possible = possible;
            list.Br_node_list = br_node_list;
            list.MaxMin_nu_mu = maxmin_nu_mu;
            list.MaxMin_y = maxmin_y;
        end

    end % methods

end % classdef


classdef Subspace
    properties
        Num = 0;
        Base_con = [];
        Space_con = [];
        K = [];
        Verties = [];
        Vol = [];
        Mid_point = [];
        Tab = [];
        Basic_sol = [];
        Tab_type = [];
        Radius = 0;
        Basic_sol_v = [];
        MaxMin = [];
        Possible = 0;
    end

    methods
        function obj = Subspace(n, base_con, sc, k, v, vol, mid, tab, radius, 
basic_sol, tab_type, basic_sol_v, max_min, possible)
            % Num, Base_con, Space_con, K, Verties, Vol,
            % Mid_point, Tab, Basic_sol, Tab_type, MaxMin, Possible
            obj.Num = n;
            obj.Base_con = base_con;
            obj.Space_con = sc;
            obj.K = k;
            obj.Verties = v;
            obj.Vol = vol;
            obj.Mid_point = mid;
            obj.Tab = tab;
            obj.Radius = radius;
            obj.Basic_sol = basic_sol;
            obj.Tab_type = tab_type;
            obj.Basic_sol_v = basic_sol_v;
            obj.MaxMin = max_min;
            obj.Possible = possible;
        end

        function Show(obj)
            disp(['Subspace number: ', num2str(obj.Num)]);
        end

    end
end


function [Tab_distribution] = gen_rhs_map_subspace_5d_newmet(A, c, rand_b, lowbound,
possible_tol)

% This function is intend to

% example for this research:
% A = [      0    9.4600    9.8400   19.2300
%     35.0000   12.9200   12.7300         0
%     0    9.4600    9.8400   19.2300
%     35.0000   12.9200   12.7300         0
%     39.4300         0   -1.2400  -12.1200];
% c = [1 1 1 1]';
% c = -c; % max
% rand_b = [10 15]; % an asumption of b range for calculation, no need to be changed
% lowbound = 0;
% possible_tol = 1e-9;
% [Tab_distribution] = gen_rhs_map_subspace_5d_newmet(A, c, rand_b, lowbound,
possible_tol)

%%%%%%% Main Part of Functoin %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initialize constrains

lenb = length(A(:,1));
lenA = length(A(1,:));
lenc = length(c(:, 1));
Anew = [A, eye(lenb)];
lenAnew = length(Anew(1,:));

syms_b = ' ';
syms_bound = ' ';

for i = 1:lenb
    syms_b = [syms_b 'b' num2str(i) ' '];
    % b1 b2 b3;
end

eval(['syms', syms_b, 'real'])
% syms b1 b2 b3 real
eval(['b = [', syms_b, '];'])
% b = [b1 b2 b3];
b = b';

for i = 1:lenb
    syms_bound = [syms_bound 'diff(xi,b' num2str(i) ') '];
    % diff(xi, b1) diff(xi, b2) diff(xi, b3)
end

N = nchoosek(1:lenAnew,lenb);
N = N(1: (end-1), :);%eliminate 0 0 0 ... condition
lenN = length(N(:,1));
factor = ff2n(lenA);
factor = factor(2:end, :);%eliminate 0 0 0 ... condition
lenF = length(factor);
```

```
Tab_distribution = cell(lenF, 1);
for i = 1:lenF
    Tab_distribution{i} = AtabDistribution(i, factor(i, :), [], [], [], 0, []);
    % Tab_type, Basic_tab, List, List_edge, List_temp, Possible, MaxMin
end

%% Initialize Subspaces
Subspace_list = cell(lenN, 1);
space_con = [];
vert_set = [];
Mid_point = zeros(lenb,1);
radius = inf;
base_con = [];
basic_sol = [];
tab_type = [];
Subspace_list{1} = Subspace(1, base_con, space_con, [], ...
    vert_set, [], Mid_point, [], radius, basic_sol, tab_type, [], [], 0);
% Num, Base_con, Space_con, K, Verties, Vol, Mid_point, Tab, Radius,
% Basic_sol, Tab_type, Basic_sol_v

%% Find Base Subspaces
n = 0;
bound_set = zeros(lenb, lenb + 1);
basic_sol_v = bound_set;
basic_zero_tab = zeros(1, lenA);
for i = 1:lenN
    Atemp = Anew(:,N(i,:)');
    if rank(Atemp) == lenb
        ls = linsolve(Atemp,b);
        for j = 1:lenb
            xi = ls(j,:);
            eval(['bound = [', syms_bound, '];'])
            % bound = [diff(xi, b1) diff(xi, b2) diff(xi, b3)];
            basic_sol_v(j, :) = [bound, 0];
            unitlize = find(abs(bound) > 0); % trying to avoid bug of noredund
            bound = bound/abs(bound(unitlize(1)));
            bound_set(j, :) = [bound, 0];% f(b_i) > bound_set(:,end)
        end
        % setting each subspace
        % core properties
        f = ones(lenb, 1);
        [~,~,EXITFLAG] = linprog(f,- bound_set(:, 1:end-1),- bound_set(:,end));
        % [An,bn,~] = noredund(A_space, b_space);
        if EXITFLAG ~= 2
            n= n + 1;
            basic_sol_temp = basic_zero_tab;
            Subspace_list{n} = Subspace_list{1};
            Subspace_list{n}.Num = n;
            Subspace_list{n}.Base_con = - bound_set;% f(b_i) < bound_set(:,end)
            Subspace_list{n}.Tab = N(i,:);
            Subspace_list{n}.Basic_sol_v = basic_sol_v;
            basic_sol_temp(Subspace_list{n}.Tab(N(i,:) <= lenA)) = 1;
            for k = 1:lenF
                if basic_sol_temp == Tab_distribution{k}.Basic_tab
                    Subspace_list{n}.Basic_sol = basic_sol_temp;
                    Subspace_list{n}.Tab_type = k;
                    % obtaining basic solutoin value which could be used in cost
function
                    basic_tab = Subspace_list{n}.Tab(Subspace_list{n}.Tab <= lenc);
                    basic_n = (sum(1 == basic_sol_temp));
                    basic_sol_v = Subspace_list{n}.Basic_sol_v;
                    basic_v = basic_sol_v(1:basic_n, :);
                    basic_sol_v = zeros(lenc, lenb + 1);
                    s = 0;
                    for r = 1:lenc
                        if 1 == basic_sol_temp(r)
                            s = s + 1;
                            basic_sol_v(r, :) =  basic_v(s, :);
                        end
                    end
                    Subspace_list{n}.Basic_sol_v = basic_sol_v;
                    Tab_distribution{k}.List = [Tab_distribution{k}.List;
Subspace_list{n}];
                    break
                end
            end
        end
    end
end
Subspace_list((n+1): end) = [];

%% upper section can run for only one time

A_wholeSpace = [-ones(1,lenb);ones(1,lenb); -eye(lenb)];
b_wholeSpace = [-rand_b(1); rand_b(2); ones(lenb,1)*lowbound];

[V_whole,~,~] = lcon2vert_ef(A_wholeSpace, b_wholeSpace,[],[],[]);

[~,wholeVol] = convhulln(V_whole);

[Subspace_edge_list] = subspace_edge_new(Subspace_list, lenb, rand_b, lowbound);
lenSe= length(Subspace_edge_list);

for i = 1:lenF
    Tab_distribution{i}.List_edge = [];
end

for i = 1:lenSe
    type = Subspace_edge_list{i}.Tab_type;
    Tab_distribution{type}.List_edge = [Tab_distribution{type}.List_edge;
Subspace_edge_list{i}];
end

% eliminate some tableaus which volume is too small to be happen
j_eli = 0;
for i = 1:lenF
    lenL = length(Tab_distribution{i}.List_edge);
    sub_s_list = Tab_distribution{i}.List_edge;
    edge_list_empty = [];
    for j = 1:lenL
        % [~,Voltab] = convhulln(sub_s_list(j).Verties, {'QJ'});
        try
            [~,~] = convhulln(sub_s_list(j).Verties);
            % Voltab/wholeVol <= possible_tol
        catch
            edge_list_empty = [edge_list_empty, j];
            j_eli = j_eli + 1;
        end
    end
    sub_s_list(edge_list_empty) = [];
    Tab_distribution{i}.List_edge = sub_s_list;
end
```

```
%% cut one side of the overlap range from same basic tableau %% analyse by set theory
(this part is needed to be reconsidered)

for i = 1:lenF
    lenL = length(Tab_distribution{i}.List_edge);
    Tab_distribution{i}.List_temp = Tab_distribution{i}.List_edge;
    if lenL >= 2
        list_edge = Tab_distribution{i}.List_edge;
        List_temp = [];
        for j = 1: lenL
            n_sub_cut = Tab_distribution{i}.List_edge;
            n_sub_cut(j) = [];
            n_sub_remain = Tab_distribution{i}.List_edge(j);
            for k = 1 : lenL - 1
                n_sub_remain_temp = [];
                for kk = 1:length(n_sub_remain)
                    [~, extr_Subspace_list_q, ~, overlap_q] = ...
                        cost_fun_cut_better(n_sub_cut(k), n_sub_remain(kk), c);
                    n_sub_remain_temp = [n_sub_remain_temp; extr_Subspace_list_q;
overlap_q];
                end
                n_sub_remain = n_sub_remain_temp;
            end
            List_temp = [List_temp; n_sub_remain];
        end
        Tab_distribution{i}.List_temp = List_temp;
        Tab_distribution{i}.List_edge = Tab_distribution{i}.List_temp;
    end
end


% getting vertices and midpoint
Subspace_edge_list = [];
j_eli2 = 0;
V_eli = 0;
P_eli = 0;
for i = 1:lenF
    lenL = length(Tab_distribution{i}.List_edge);
    if lenL >= 1
        sub_s_list = Tab_distribution{i}.List_edge;
        edge_list_empty = [];
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%% ( this part may not need
)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        for j = 1:lenL
            A_space = sub_s_list(j).Space_con(:, 1:(end - 1));
            b_space = sub_s_list(j).Space_con(:, end);
            [V,~,~]=lcon2vert(A_space, b_space, [], [], 1e-9, []);
            if isempty(V)
                edge_list_empty = [edge_list_empty, j];
            else
                sub_s_list(j).Verties = V;
                [~,Voltab] = convhulln(V, {'QJ'});
                if Voltab/wholeVol <= possible_tol
                    edge_list_empty = [edge_list_empty, j];
                    j_eli2 = j_eli2 + 1;
                    V_eli = V_eli + Voltab;
                    P_eli = P_eli + Voltab/wholeVol;
                else
                    for k = 1:lenb
                        sub_s_list(j).Mid_point(k) = (max(V(:,k))+min(V(:,k)))/2;
                    end
                    midpoint = sub_s_list(j).Mid_point;
                    lenV = length(V(:,1));
                    distance_p = zeros(lenV, 1);
                    for k = 1:lenV
                        distance_p(k) = norm(midpoint - V(1, :)');
                    end
                    sub_s_list(j).Radius = max(distance_p);
                end
            end
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        sub_s_list(edge_list_empty) = [];
        Tab_distribution{i}.List_edge = sub_s_list;
        Subspace_edge_list = [Subspace_edge_list; sub_s_list];
    end
end

% eliminate empty Tab_distribution
tab_edge_dis = [];
for i = 1:lenF
    if ~isempty(Tab_distribution{i}.List_edge)
        tab_edge_dis = [tab_edge_dis; i];
    end
end


% save('5d_de_bug_308.mat');

%% calculating cost function cutting

% load 5d_de_bug_308.mat

lent = length(tab_edge_dis);

% A_space = [-ones(1,lenb);ones(1,lenb); -eye(lenb)];
% b_space = [-rand_b(1); rand_b(2); ones(lenb,1)];
% [V,~,~]=lcon2vert(A_space, b_space, [], [], 1e-9, [])
% [~,sum_vol_real] = convhulln(V);

Tab_distribution_mant = Tab_distribution;

error_tab_conb = [];
% for i = 1:lent
for i = 1:lent
    j_list = tab_edge_dis;
    j_list(i) = [];
    for j = 1:length(j_list)
        disp([i, j, tab_edge_dis(i), j_list(j), lent, lent - 1]);
        List_L = Tab_distribution{tab_edge_dis(i)}.List_edge;
        List_R = Tab_distribution_mant{j_list(j)}.List_edge;
        if ~isempty(List_L) && ~isempty(List_R)
            L_remain_temp = List_L;
            for k = 1:length(List_R)
                L_remain = [];
                for kk = 1:length(L_remain_temp)
                    try
                        [extr_Subspace_list_p, ~, overlap_p, ~] = ...
                            cost_fun_cut_better(L_remain_temp(kk), List_R(k), c);
                        L_remain = [L_remain; extr_Subspace_list_p; overlap_p];
                        %  if j > 1
                        %  [L_remain, ~] = check_L_remain(L_remain, 1e-12, 1, 
sum_vol_real);
                        %  end
                    catch
                        error_tab_conb = [error_tab_conb; tab_edge_dis(i), j_list(j)];
                        L_remain = L_remain_temp;
                    end
                end
                L_remain_temp = L_remain;
            end
            %  [L_remain, ~] = check_L_remain(L_remain, 1e-12, 1, sum_vol_real);
            Tab_distribution{tab_edge_dis(i)}.List_edge = L_remain;
        end
    end
    % [Tab_distribution{tab_edge_dis(i)}.List_edge, ~] = 
    % check_L_remain(Tab_distribution{tab_edge_dis(i)}.List_edge, 1e-9, 0, 0);
    % elimate subtab smaller than 1e-9 of whole tab
end


for i = 1:lenF
    lenL = length(Tab_distribution{i}.List_edge);
    if lenL >= 1
        for j = 1:lenL
            Tab_distribution{i}.List_edge(j).Space_con(1:2, :) = [];
        end
    end
end

save('rhs_map_5d_new_all');
save('rhs_map_5d_new', 'Tab_distribution', 'A', 'c');
end


function [extr_Subspace_list_p, extr_Subspace_list_q, overlap_p, overlap_q] = ...
    cost_fun_cut_better(Subspace_list_p, Subspace_list_q, c)
% since this part might generate nonconvex hull, using convex part cutting another one
% always want to obtain min cost function

% [p_Space_con, q_Space_con] = 
cot_fun_cut(Subspace_edge_list{p},Subspace_edge_list{q}, c);
% Subspace_edge_list{p}.Space_con = p_Space_con;
% Subspace_edge_list{q}.Space_con = q_Space_con;

% ef = one of [1 2 3 4 5]
% ef = 0;

cost_fun_con =  c'*Subspace_list_p.Basic_sol_v - c'*Subspace_list_q.Basic_sol_v;

unitlize = find(abs(cost_fun_con) > 0); % trying to avoid bug of noredund
cost_fun_con = cost_fun_con/abs(cost_fun_con(unitlize(1)));

% cost_p < cost_q ==>  cost_p - cost_q < 0  ==> cost_fun_con < 0

% 1. find the overlap range
% 2.  check if c across or not the range
lenSp = length(Subspace_list_p.Space_con);
overlap_range_re = [Subspace_list_p.Space_con; Subspace_list_q.Space_con];
```

```
%% overlap cutting
[~,nro,~,ef] = lcon2vert_ef(overlap_range_re(:, 1: end-1), overlap_range_re(:, end),
[], [], 1e-10, []);
% [An,bn,nro] = noredund(overlap_range_re(:, 1: end-1), overlap_range_re(:, end));

if  0 == ef && ~isempty(nro) %overlap

    overlap_range = overlap_range_re(nro,:);

    extr_p = find(nro <= lenSp);
    if ~isempty(extr_p)
        overlap_con_p = overlap_range_re(nro(extr_p), :);
        extr_Subspace_list_q = overlap_cut(Subspace_list_q, overlap_con_p);
    else
        extr_Subspace_list_q = [];
    end

    extr_q = find(nro > lenSp);
    if ~isempty(extr_q)
        overlap_con_q = overlap_range_re(nro(extr_q), :);
        extr_Subspace_list_p = overlap_cut(Subspace_list_p, overlap_con_q);
    else
        extr_Subspace_list_p = [];
    end


    %% cosfunction cutting

    overlap_p = Subspace_list_p;
    overlap_q = Subspace_list_q;

    A_con = [overlap_range(:, 1: end-1); cost_fun_con(1: end-1)];
    b_con = [overlap_range(:, end); cost_fun_con(end)];
    [~,nr,~,ef]=lcon2vert_ef(A_con, b_con, [], [], 1e-10, []);

    if 0 ~= ef || isempty(nr) % overlap part belongs to q
        overlap_q.Space_con = overlap_range;
        overlap_p = [];
    else
        overlap_p.Space_con = [A_con(nr,:), b_con(nr,:)];

        A_con = [overlap_range(:, 1: end-1); - cost_fun_con(1: end-1)];
        b_con = [overlap_range(:, end); - cost_fun_con(end)];
        [~,nr,~,ef]=lcon2vert_ef(A_con, b_con, [], [], 1e-10, []);
        if 0 ~= ef || isempty(nr) % overlap part belongs to p
            overlap_q = [];
        else
            overlap_q.Space_con = [A_con(nr,:), b_con(nr,:)];
        end

    end

else %non-overlap
    extr_Subspace_list_p = Subspace_list_p;
    extr_Subspace_list_q = Subspace_list_q;
    overlap_p = [];
    overlap_q = [];
end

end
```

Main codes that designed for Chapter 4 Nominal EMPC

```
% Nominal Control
%-------------------------------------------------------------------
% Main function for model, control parameters and initial conditions
%-------------------------------------------------------------------
clear variables
close all
clc
%% Initial Conditions
% z0_model = [0.4 0.21 0.2 0.001]'; %(Glc,O2,Aci,X)
z0_model = [0.4 0.21 0.2 0.001]'; %(Glc,O2,Aci,X)
V0 = 0.3; % L, Initial Volume of the reactor, guess
Vmax = 0.4; % L, Final maximum batch volume, guess
Vmin = 0.2; % L, Final minimum batch volume, guess
% Fig = 0.1; % L/h
Fmax = 0.3; % L/h
Zgl_feed = 5;
%% Model parameters
A = [0 9.46 9.84 19.23; 35 12.92 12.73 0; -39.43 0 1.24 12.12];
c = ones(4,1);
kla = 4; % hr^-1, Mahadevan paper
Km = 0.015; % mM, Mahadevan paper
GUR_max = 6.5; % mM/g-dw/hr, Mahadevan paper
OUR_max = 12; % mM/g-dw/hr, Mahadevan paper
Ki = 1.0;
% Uncertainty Information
GUR_sig = 0.2; % +/- 20%, guess
OUR_sig = 0.2; % +/- 20%, guess
kla_sig = 0.2;
Km_sig = 0.01;
Ki_sig = 0.2;
% # of time steps and step size
% nit = 110;
% tend = 11; % h, total time of cell culture growth, guess
nit = 100; %110
tend = 10; % 1h, total time of cell culture growth, guess
dt = tend/nit; % h, time, guess
%% frequency for disturbance
%% frequency for disturbance
t = [1:1:nit]'; n = 11; % frequency for disturbance
% disturbance for 109, se01
% disturbance = sin(2*pi/n*t);
% disturbance2 = cos(pi/n*t);
% disturbance3 = - sin(pi/n*t);
% disturbance for se02——1.5
disturbance = -1*ones(length(t), 1);
disturbance2 = 1*ones(length(t), 1);
disturbance3 = 1*ones(length(t), 1);
dist = [disturbance, disturbance2, disturbance3];
% n2 = 8; % frequency for changing plant definition
% load('perf_disturbance.mat');
% load('perf_disturbance2.mat');
beta = 0; % cell death paramete
%% controller parameters
number_of_inputs = 10;  % # of manipulation
p = number_of_inputs;
u = zeros(2*p,1); % combination of F and P predictions
dl = tend/number_of_inputs; % h, time interval of each measurement and manipulation
% Initial guess for Feed rate
% load('feed_rate_ig_nit_110_nominal.mat');
uig = 0.001*ones(p*2,1);
```

```
% uig = [(Vmax-V0)/dt/nit(1)*ones(nit(1),1); 1e-3*ones(nit,1)];
% uig = 0.02*ones(2*number_of_inputs,1);
umax = Fmax*ones(2*number_of_inputs,1);
%% Optimisation parameters
op = optimset('fmincon');
op.Display = 'On';
op.TolFun = 1E-7;
op.TolX = 1E-6;
op.MaxIter = 10000;
op.MaxFunEvals = 100000;
op.Algorithm = 'interior-point';
op.UseParallel = 1;
GUR_plant_list = [6.5 6.5 6.5 8 4];
kla_plant_list = [4 2.4 5.6 4 4];
% GUR_plant = GUR_max;
%% the control and plant loop
% initialisation
z0_plant = z0_model;%(Glc,O2,Aci,X)
% z0_plant = [0.4 0.21 0.2 0.001]';%(Glc,O2,Aci,X)
fb_k = 0; % initialisation of fb error
if isempty(gcp('nocreate')) == 1  %matlab pool not yet started
    parpool('local');
    % for both========================================
else  %matlab pool already started
    disp('matlab pool already started');
end
    GUR_plant = 6.5;
    kla_plant = 4;
    filename = strcat('nominal_control_GUR_OUR_kla1010_se02_-1_1_1_0.1.mat');
    main_controller (nit, dt, tend,A, c, kla, kla_sig, Km, GUR_max, GUR_sig,...
        OUR_max, Ki, Ki_sig, uig, Fmax, fb_k, p, u, op, z0_model, z0_plant,...
        V0, Vmax, Vmin, Zgl_feed, OUR_sig, beta, filename, ...
        GUR_plant, kla_plant, dl, number_of_inputs, dist)
% end
% delete(gcp('nocreate'))%commented when adjusting or debugging parpool
% matlabpool close %commented when adjusting or debugging parpool


%-----------------------------------------------------------
% Main Controller (Nominal Control)
%-----------------------------------------------------------
function main_controller (nit, dt, tend,A, c, kla, kla_sig, Km, GUR_max, GUR_sig,...
    OUR_max, Ki, Ki_sig, uig, Fmax, fb_k, p, u, op, z0_model, z0_plant,...
    V0, Vmax, Vmin, Zgl_feed, OUR_sig, beta, filename, ...
    GUR_plant, kla_plant, dl, number_of_inputs, dist)
%% variables to store
Basic_model = zeros(nit,4);
Basic_plant = zeros(nit,4);
y_plant = zeros(nit,5);
z_plant = zeros(nit,3);
x_plant = zeros(nit,1);
V_plant = zeros(nit,1);
u_plant = zeros(2*nit,p);
nu_plant = zeros(nit,4);
F_plant = zeros(nit,1);
P_plant = zeros(nit,1);
x_model = zeros(nit,1);
u_model = zeros(2*p,p);
fval_model = zeros(p,1);
% y_model = zeros(nit,5);
z_model = zeros(nit,3); % stores model predictions
nu_model = zeros(nit,4);
```

```
fb_plant = zeros(nit,1); % feedback error in nominal model prediction
time_store = zeros(p,1);
ef_plant = zeros(nit,1);
fval_plant = zeros(nit,1);
cost_plant = zeros(nit,1); % cost or yield is equal to x*V
% obj_plant = zeros(nit,5);
y_measured = zeros(p,5); %(Glc,O2,Aci,X,V)
nu_measured = zeros(p,4);
%% initialise the control inputs
%n = length(c);
y = [z0_model;V0];%initial plant (Glc,O2,Aci,X,V)
nu = zeros(4,1);%initial rate (Glc,O2,Aci,X)
mu = sum(nu);
mu_m = mu;
tl = 0;
maninter = dl/dt;
disp('Controller without cell death,bounded reference trajectory');
%% Plant parameters
for k = 1:nit
    %% controller prediction
    if mod((k-1), maninter) == 0 % decide which time to measure & manipulate
        kp = k;
        tl = 1+tl;
        % fb_k0 = fb_k/maninter;
        fb_k0 = 0;
        p = number_of_inputs+1-tl;
        % measurements records
        y_measured(tl,:) = y';
        nu_measured(tl,:) = nu';
        z0_model = y(1:4); % adjusting model after be measured
        tstart = clock; % to save the time that fmincon cost
        % Linear Optimisation Constraints
        umax = Fmax*ones(2*p,1);
        % du_max = Fmax;
        % u = zeros(2*p,1);
        % objective function weightsw = [10 0.05 0.5]; % e_x-xref, var(x)
        A_cons = [tril(ones(p,p)) tril(-ones(p,p)); ... % Linear Constraint:
V(k+i)<=Vmax
            -tril(ones(p,p)) tril(ones(p,p))]; % Linear Constraint: V(k+i)>=Vmin
        b_cons = [(Vmax - V0)/dl*ones(p,1); ... % Linear Constraint: V(k+i)<=Vmax
            (V0 - Vmin)/dl*ones(p,1)]; % Linear Constraint: V(k+i)>=Vmin
        [u, fval, exitflag] = fmincon(@(u) objfun (u, z0_model, V0, dt, A, c, kla, Km,
GUR_max,...
            OUR_max, Ki, Zgl_feed, nit, p, kp, maninter,
fb_k),uig,A_cons,b_cons,[],[],zeros(2*p,1),umax,[],op); %ÕâÀï²»´«µÝmu
        % save prediction results and time consumption
        u_model(tl:number_of_inputs,tl) = u(1:p)';
        u_model(number_of_inputs+tl:end,tl) = u(p+1:end)';
        fval_model(tl,1) = -fval;
        time_store(tl,1) = etime(clock,tstart)/60; % min
        % initial guess for next time interval comes from computed solution
        uig = [u(2:p); u(p+2:end)];
    end
    F0 = u(1);
    P0 = u(p+1);
    %% plant dynamics
    % plant parameters
    GUR_p = GUR_max*(1 + GUR_sig*dist(k, 1));

    OUR_p = OUR_max*(1 + OUR_sig*dist(k, 2));

    kla_p = kla*(1 + kla_sig*dist(k, 3));
    Ki_p = 1;

    % GUR_p = GUR_plant(k); kla_p = kla_plant(k);
    % OUR_p = OUR_plant(k); Ki_p = Ki_plant(k);
    [y, nu, Basic, mu] = plant_dynamics(z0_plant, V0, F0, P0, dt, A, c, ...
        kla_p, Km, GUR_p, OUR_p, Ki_p, Zgl_feed, mu);
    %% run nominal model dynamics for one time step
    fb_ke = fb_k0;
    [y_m, nu_m, Basic_m, mu_m] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c, ...
        kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu_m);
    % reinitialise
    z0_plant = y(1:4);
    z0_model = y_m(1:4);
    z0_model(4, :) = z0_model(4, :) + fb_ke;
    V0 = y(5);
    fb_k = y(4) - z0_model(4);
    %% save the results
    Basic_model(k,:) = Basic_m;
    Basic_plant(k,:) = Basic;
    y_plant(k,:) = y';
    z_plant(k,:) = y(1:3)';
    u_plant(k,1:p) = u(1:p)';
    u_plant(nit+k,1:p) = u(p+1:end)';
    F_plant(k,1) = u(1);
    P_plant(k,1) = u(p+1);
    % F_plant(k,1:p) = u(1:p)';
    % P_plant(k,1:p) = u(p+1:end)';
    V_plant(k,1) = y(5);
    x_plant(k,1) = y(4);
    nu_plant(k,:) = nu';
    ef_plant(k,1) = exitflag;
    fval_plant(k,1) = -fval; % modeled cost
    cost_plant(k,1) = y(4)*y(5);
    x_model(k,1) = z0_model(4);
    % y_model(k,:) = [z0_model; V0]';
    z_model(k,:) = z0_model(1:3)';
    nu_model(k,:) = nu_m';
    fb_plant(k,1) = fb_k;
    save(filename);
end
end


function ydot = model_dynamics(~,z, kla, ...
Anu_gl, Anu_o2, Anu_Ac, mu, Zgl_feed, F0, P0)
% function ydot = model_dynamics(t,z, kla, ...
% Anu_gl, Anu_o2, Anu_Ac, mu, Zgl_feed, F0, P0)
V = z(5); X0 = z(4);
ydot(1,1) = F0/V*(Zgl_feed - z(1)) - Anu_gl*X0 ;
ydot(2,1) = kla*(0.21 - z(2)) - Anu_o2*X0 - F0/V*z(2);
ydot(3,1) = -F0/V*z(3) + Anu_Ac*X0 ;
ydot(4,1) = mu*X0 - (F0-P0)/V*X0;
ydot(5,1) = F0-P0;
end


function fmin = objfun (u, z0_model, V0, dt, A, c, kla, Km, GUR_max, ...
    OUR_max, Ki, Zgl_feed, nit, p, kp, maninter, fb_k)
% structure of z0_model = [zgl, zo2, zac, x]
% y_plant_p = zeros(nit,5);
% nu_plant_p = zeros(nit,4);
```

```
tl = 0;
mu = 0;% initial condition of mu always be zero.
for k = kp:nit
    if mod((k-1), maninter)==0
        tl = tl+1;
        F0 = u(tl);P0 = u(p+tl);
    end
    [y, ~, ~, mu] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c, ...
        kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu);
    % Reinitialise z0_model and V0 for next time step
    % y(4) = y(4) + fb_k;
    z0_model = y(1:4);
    V0 = y(5);
    % store the plant dynamics (not essential)
    % nu_plant_p(k,:) = nu';
    % y_plant_p(k,:) = y';
end
fmin = -(y(4) + fb_k)*y(5); % max cost = final_x*final_V


function [y, nu, Basic, mu] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c, ...
    kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu)
% structure of z0_plant = [zgl, zo2, zac, x]
% LP Model solution
% n = length(c);
if (z0_model(1)<=1e-3 && z0_model(3)<=1e-3)
    X_k = z0_model(4);
else
    X_k = z0_model(4) + (-mu*z0_model(4) - (F0-P0)/V0*z0_model(4))*dt;
end
Anew = [A; A(1:2,:)*X_k; -A(3,:)*X_k];
b = [GUR_max*(z0_model(1)/(Km + z0_model(1)));...
    OUR_max; ...
    100; ...
    F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt; ...
    kla*(0.21 - z0_model(2)) - F0/V0*z0_model(2) + z0_model(2)/dt;...
    F0/V0*z0_model(3) + z0_model(3)/dt; ];
% - F0/V0*z0_model(3) + z0_model(3)/dt; ];
[nu, mu, Basic] = simplex_tab(-c,Anew,b,[],[],[]);
Anu_gl = A(1,:)*nu; Anu_o2 = A(2,:)*nu; Anu_Ac = A(3,:)*nu;
tspan = [0 dt];
z0 = [z0_model; V0];
if (z0(1)<=0 && z0(2)<=0 && z0(3)<=0)
    z = [0 0 0 z0(4) V0];
else
    [~,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2, Anu_Ac, ...
        -mu, Zgl_feed, F0, P0), tspan, z0);
%   [t,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2, Anu_Ac, ...
%       -mu, Zgl_feed, F0, P0), tspan, z0);
end
y = z(end,:)';
if (z(end,1) <= 1e-6) % check for O2 concentration
    y(1,1) = 1e-6;
end
if (z(end,2) <= 1e-6) % check for O2 concentration
    y(2,1) = 1e-6;
end
if (z(end,3) <= 1e-6) % check for O2 concentration
    y(3,1) = 1e-6;
end
end
```

Main codes that designed for Chapter 3 Robust EMPC

```
% Nominal Control
%-------------------------------------------------------------------
% Main function for model, control parameters and initial conditions
%-------------------------------------------------------------------
clear variables
close all
clc
%% Initial Conditions
z0_model = [0.4 0.21 0.2 0.001]'; %(Glc,O2,Aci,X)
V0 = 0.3; % L, Initial Volume of the reactor, guess
Vmax = 0.4; % L, Final maximum batch volume, guess
Vmin = 0.2; % L, Final minimum batch volume, guess
% Fig = 0.1; % L/h
Fmax = 0.3; % L/h
Zgl_feed = 5;
%% Model parameters
A = [0 9.46 9.84 19.23; 35 12.92 12.73 0; -39.43 0 1.24 12.12];
c = ones(4,1);
kla = 4; % hr^-1, Mahadevan paper
Km = 0.015; % mM, Mahadevan paper
GUR_max = 6.5; % mM/g-dw/hr, Mahadevan paper
OUR_max = 12; % mM/g-dw/hr, Mahadevan paper
Ki = 1.0;
% Uncertainty Information
GUR_sig = 0.2; % +/- 20%, guess
OUR_sig = 0.2; % +/- 20%, guess
kla_sig = 0.2;
Km_sig = 0.01;
Ki_sig = 0.2;
% # of time steps and step size
% nit = 110; %110
% tend = 11; % 1h, total time of cell culture growth, guess
nit = 100; %110
tend = 10; % 1h, total time of cell culture growth, guess
dt = tend/nit; % h, time, guess
%% frequency for disturbance
t = [1:1:nit]'; n = 12; % frequency for disturbance
disturbance = -1*ones(length(t), 1);
disturbance2 = 1*ones(length(t), 1);
disturbance3 = 1*ones(length(t), 1);
dist = [disturbance, disturbance2, disturbance3];
% n2 = 8; % frequency for changing plant definition
% load('perf_disturbance.mat');
% load('perf_disturbance2.mat');
beta = 0; % cell death paramete
%% controller parameters
number_of_inputs = 10;  %11% # of manipulation
eti = 0.5; %0.5h per estimation (propagate uncertainty)
p = number_of_inputs;
u = zeros(2*p,1); % combination of F and P predictions
dl = tend/number_of_inputs; % h, time interval of each measurement and manipulation
% Initial guess for Feed rate
% load('feed_rate_ig_nit_110_nominal.mat');
% load feed_rate_ig_nit_110_wb.mat;
uig = 0.001*ones(p*2,1);
% u = uig_discrete;
% uig = [(Vmax-V0)/dt/nit(1)*ones(nit(1),1); 1e-3*ones(nit,1)];
% uig = 0.02*ones(2*number_of_inputs,1);
load rhs_map_5d_new.mat
load rhs_map_2d_new.mat
```

```
t2d = Tab_distribution_2d;
t5d = Tab_distribution_5d;
umax = Fmax*ones(2*number_of_inputs,1);
tab_tol = 0.20;
%% Optimisation parameters
op = optimset('fmincon');
op.Display = 'On';
op.TolFun = 1E-7;
op.TolX = 1E-6;
% op.TolFun = 1E-6;
% op.TolX = 1E-5;
% op.MaxIter = 10000;1000
op.MaxIter = 10000;
% op.MaxFunEvals = 100000;3000
op.MaxFunEvals = 100000;
op.Algorithm = 'interior-point';
op.UseParallel = 1;
% op.LargeScale = 'on';
% objective function weightsw = [10 0.05 0.5]; % e_x-xref, var(x)
opt = optimoptions('linprog','Algorithm','interior-point');
GUR_plant = GUR_max;
kla_plant = kla;
%% the control and plant loop
% initialisation
z0_plant = z0_model;%(Glc,O2,Aci,X)
% z0_plant = [0.4 0.21 0.2 0.001]';%(Glc,O2,Aci,X)
fb_k = 0; % initialisation of fb error
if isempty(gcp('nocreate')) == 1  %matlab pool not yet started

    parpool('local');
    % for both========================================
else  %matlab pool already started
    disp('matlab pool already started');
end
    filename = strcat('robust_control_wb_GUR-20_OUR-20_kla-201010_se02_-
1_1_1_dt0.50.1.mat');
    main_controller(nit, dt, tend,A, c, kla, kla_sig, Km, GUR_max, GUR_sig,...
        OUR_max, Ki, Ki_sig, uig, Fmax, fb_k, p, u, op, z0_model, z0_plant,...
        V0, Vmax, Vmin, Zgl_feed, OUR_sig, beta, filename, ...
        GUR_plant, kla_plant, dl, number_of_inputs, dist, tab_tol, eti, opt, t2d, t5d)
% delete(gcp('nocreate'))%commented when adjusting or debugging parpool
% matlabpool close %commented when adjusting or debugging parpool


%-----------------------------------------------------------
% Main Controller (Nominal Control)
%-----------------------------------------------------------
function main_controller (nit, dt, tend,A, c, kla, kla_sig, Km, GUR_max, GUR_sig,...
    OUR_max, Ki, Ki_sig, uig, Fmax, fb_k, p, u, op, z0_model, z0_plant,...
    V0, Vmax, Vmin, Zgl_feed, OUR_sig, beta, filename, ...
    GUR_plant, kla_plant, dl, number_of_inputs, dist, tab_tol, eti, opt, t2d, t5d)
%% variables to store
Basic_model = zeros(nit,4);
Basic_plant = zeros(nit,4);
y_plant = zeros(nit,5);
z_plant = zeros(nit,3);
x_plant = zeros(nit,1);
V_plant = zeros(nit,1);
u_plant = zeros(2*nit,p);
nu_plant = zeros(nit,4);
F_plant = zeros(nit,1);
P_plant = zeros(nit,1);
```

```
x_model = zeros(nit,1);
u_model = zeros(2*p,p);
fval_model = zeros(p,1);
% y_model = zeros(nit,5);
z_model = zeros(nit,3); % stores model predictions
nu_model = zeros(nit,4);
fb_plant = zeros(nit,1); % feedback error in nominal model prediction
time_store = zeros(p,1);
ef_plant = zeros(nit,1);
fval_plant = zeros(nit,1);
cost_plant = zeros(nit,1); % cost or yield is equal to x*V
% obj_plant = zeros(nit,5);
y_measured = zeros(p,5); %(Glc,O2,Aci,X,V)
nu_measured = zeros(p,4);
%% initialise the control inputs
%n = length(c);
y = [z0_model;V0];%initial plant (Glc,O2,Aci,X,V)
nu = zeros(4,1);%initial rate (Glc,O2,Aci,X)
mu = sum(nu);
mu_m = mu;
tl = 0;
maninter = dl/dt;
disp('Controller without cell death,bounded reference trajectory');

for k = 1:nit
    %% controller prediction
    if mod((k-1), maninter) == 0 % decide which time to measure & manipulate
        kp = k;
        tl = 1+tl;
        % fb_k0 = fb_k;
        fb_k0 = 0;
        p = number_of_inputs+1-tl;
        % measurements records
        y_measured(tl,:) = y';
        nu_measured(tl,:) = nu';
        z0_model = y(1:4); % adjusting model after be measured
        z0_model_n = z0_model;
        tstart = clock; % to save the time that fmincon cost
        % Linear Optimisation Constraints
        umax = Fmax*ones(2*p,1);
        % du_max = Fmax;
        % u = zeros(2*p,1);
        % objective function weightsw = [10 0.05 0.5]; % e_x-xref, var(x)
        A_cons = [tril(ones(p,p)) tril(-ones(p,p)); ... % Linear Constraint:
V(k+i)<=Vmax
            -tril(ones(p,p)) tril(ones(p,p))]; % Linear Constraint: V(k+i)>=Vmin
        b_cons = [(Vmax - V0)/dl*ones(p,1); ... % Linear Constraint: V(k+i)<=Vmax
            (V0 - Vmin)/dl*ones(p,1)]; % Linear Constraint: V(k+i)>=Vmin
        [u, fval, exitflag] = fmincon(@(u) objfun (u, z0_model, V0, dt, A, c, kla, Km,
GUR_max, ...
    OUR_max, Ki, Zgl_feed, nit, p, kp, maninter, GUR_sig, OUR_sig, kla_sig, fb_k,
tab_tol, eti, opt, t2d, t5d),uig,A_cons,b_cons,[],[],zeros(2*p,1),umax,[],op);
        % save prediction results and time consumption
        u_model(tl:number_of_inputs,tl) = u(1:p)';
        u_model(number_of_inputs+tl:end,tl) = u(p+1:end)';
        fval_model(tl,1) = -fval;
        time_store(tl,1) = etime(clock,tstart)/60; % min
        % initial guess for next time interval comes from computed solution
        uig = [u(2:p); u(p+2:end)];
    end
    F0 = u(1);
    P0 = u(p+1);
    %% plant dynamics
    % plant parameters
    GUR_p = GUR_max*(1 + GUR_sig*dist(k, 1));

    OUR_p = OUR_max*(1 + OUR_sig*dist(k, 2));

    kla_p = kla*(1 + kla_sig*dist(k, 3));

    Ki_p = 1;
    % GUR_p = GUR_plant(k); kla_p = kla_plant(k);
    % OUR_p = OUR_plant(k); Ki_p = Ki_plant(k);
    [y, nu, Basic, mu] = plant_dynamics(z0_plant, V0, F0, P0, dt, A, c, ...
        kla_p, Km, GUR_p, OUR_p, Ki_p, Zgl_feed, mu);
    %% run worst branch dynamics for one time step
    %     [y_m, nu_m, Basic_m, mu_m] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c,
...
    %         kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu_m);

    bran_tol = 0.05;
    % tab_tol = 0.05;
    fb_ke = fb_k0;
    ti = eti/dt;
    if mod((k-1), maninter) == 0
        % lentime = nit - kp + 1;
        time_conv = 2; % every 2 time interval converge branchs

        GUR_range = GUR_max*[1-GUR_sig 1+GUR_sig];
        OUR_range = OUR_max*[1-OUR_sig 1+OUR_sig];
        kla_range = kla*[1-kla_sig 1+kla_sig];

        % [y, nu, Basic, mu] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c, ...
        %     kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu)

        y_m = [z0_model; V0];
        time = 0;
        brsubtab = branchSubtab(time, [], [], [], [], 1, [], [], [], [], [], y_m, []);
        % time, up_leaf, mid_leaf, low_leaf, list_conbine, possible, x_grad, prob_dis,
unit, nodes, maxmin_nu_mu, maxmin_y, data
        brsubtab.Prev = 0;
        brsubtab.Mid = [];
        brsubtab.Next = 1;
        branchlist = branchList(time, brsubtab, 1, 1, []);
        % time, branchsub, pos_list, scenario

        i = 2;
        branchlist(i) = branchlist(i - 1);

        time = branchlist(i - 1).Time + 0.1;
        branchlist(i).Time = time;

        branchlist(i).BranchSubtab = [];
        branchlist(1).Wholerange = z0_model;
        z0_model = branchlist(i - 1).BranchSubtab.MaxMin_y(1:4);
        X_k = z0_model(4);

        pre = 1;

        [brsubtab_list] = gene_branch_node_of1(time, z0_model, V0, F0, P0, dt, A, c,
...
            kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki, Zgl_feed,
mu, X_k, pre, bran_tol, tab_tol, opt, t2d, t5d);
```

```
        branchlist(i).BranchSubtab = brsubtab_list;

        %generate scenario

        branchlist(i).Wholerange = branchlist(i).BranchSubtab.MaxMin_y;
        list_sub = branchlist(i).BranchSubtab;
        lensub = length(list_sub);
        branchlist(i).Pos_list = ones(lensub, 1);
        branchlist(i).Scenario = 1;
        % for j = 1:lensub
        %     branchlist(i).Pos_list(j) = list_sub(j).Possible;
        %     branchlist(i).Scenario(j, end) = j;
        %     branchlist(i).Scenario(j, 1: end - 1) = branchlist(i -
1).Scenario(list_sub(j).Prev, :);
        % end

        y_m = branchlist(i).Wholerange(:, 2);
%         z0_model = y_m;
%         V0 = y_m(5);

        branchlist(1) = branchlist(2);
        branchlist(2) = [];
        cur_subtabmap = [];
        fti = 1;
    else
        i = k - kp + 1;

        lenbran = length(branchlist(i - 1).BranchSubtab);
        branchlist(i) = branchlist(i - 1);
        branchlist(i).BranchSubtab = [];
        time = branchlist(i - 1).Time + 0.1;
        branchlist(i).Time = time;
        sump = 0;

        %% only for worst case branch
        if branchlist(i - 1).Pos_list <= tab_tol
            z0_model = branchlist(i - 1).Wholerange(1:4, :);
            V0 = branchlist(i - 1).Wholerange(5, :);
            n = [2 1];
            for j = 1:2
                nn = n(j);
%                 [y, ~, ~, mu] = plant_dynamics(z0_model(:, j), V0(j), F0, P0, dt, A,
c, ...
%                     kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu);
                [y_m, ~, ~, mu] = plant_dynamics(z0_model(:, j), V0(j), F0, P0, dt, A,
c, ...
                kla_range(nn), Km, GUR_range(nn), OUR_range(nn), Ki, Zgl_feed, mu);
                branchlist(i).Wholerange(:, j) = y_m;
            end
        else
            %%

            for j = 1:lenbran
                % lenbfr = length(branchlist(i).BranchSubtab);
                pre = j;
                %              [subtablist, sump] = gene_sub_tab(time, z0_model, V0,
F0, P0, dt, A, c, ...
                %                  kla, Km, GUR_max, OUR_max,GUR_range,OUR_range,
kla_range, Ki, Zgl_feed,...
                %                  mu, branchlist(i - 1).BranchSubtab(j), pre,
bran_tol, tab_tol, sump);
```

```
                [subtablist, sump, cur_subtabmap] = gene_sub_tab_wb(time, z0_model, 
V0, F0, P0, dt, A, c, ...
                    kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki, 
Zgl_feed,...
                    mu, branchlist(i - 1).BranchSubtab(j), pre, bran_tol, tab_tol, 
sump, fti, cur_subtabmap, opt, t2d, t5d);

                if mod((i + 1), ti)==0
                    fti = 1;
                else
                    fti = 0;
                end

                branchlist(i).BranchSubtab = [branchlist(i).BranchSubtab; subtablist];
            end

            % generate scenario
            lensub = length(branchlist(i).BranchSubtab);
            branchlist(i).Pos_list = zeros(lensub, 1);
            branchlist(i).Scenario = zeros(lensub, i);
            yylistmi = zeros(5, lensub);
            yylistma = zeros(5, lensub);

            for j = 1:lensub
                branchlist(i).BranchSubtab(j).Possible = 
branchlist(i).BranchSubtab(j).Possible/sump;
                branchlist(i).Pos_list(j) = branchlist(i).BranchSubtab(j).Possible;
                branchlist(i).Scenario(j, end) = j;
                % branchlist(i).Scenario(j, 1: end - 1) = branchlist(i - 
1).Scenario(branchlist(i).BranchSubtab(j).Prev, :);
                branchlist(i).Scenario(j, 1: end - 1) = 1;
                yylistmi(:, j) = branchlist(i).BranchSubtab(j).MaxMin_y(1:5, 1);
                yylistma(:, j) = branchlist(i).BranchSubtab(j).MaxMin_y(1:5, 2);
            end

            branchlist(i).Wholerange = [min([yylistmi, yylistma], [], 2) 
max([yylistmi, yylistma], [], 2)];
            branchlist(i).Wholerange(4, :) = fliplr(branchlist(i).Wholerange(4, :));
            y_m = branchlist(i).Wholerange(:, 2);
            %% only for whole branch method

            % if rem(i, time_conv) == 0 && lensub > 1
            %     [branchlist(i)] = branch_conv(branchlist(i), 
branchlist(i).Wholerange);
            % end
            % y_m = branchlist(i).Wholerange(:, 2);
            %% only for worst case branch method
            branchlist(i).Pos_list = 
branchlist(i).BranchSubtab.List_conbine.Possible*branchlist(i).Pos_list;
            %%
%             z0_model = y_m;
%             V0 = y_m(5);

        end
    end

    V0 = V0(1);

     [z0_model_y, ~, ~, mu] = plant_dynamics(z0_model_n, V0, F0, P0, dt, A, c, ...
        kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu);
    z0_model_n = z0_model_y(1:4);

    % reinitialise
    z0_plant = y(1:4);
    z0_model = tab_tol*y_m(1:4) + (1 - tab_tol)*z0_model_n;
    z0_model(4, :) = z0_model(4, :) + fb_ke;
    z0_model_n = z0_model;
    V0 = y(5);
    fb_k = y(4) - z0_model(4);
    %% save the results
%     Basic_model(k,:) = Basic_m;
    Basic_plant(k,:) = Basic;
    y_plant(k,:) = y';
    z_plant(k,:) = y(1:3)';
    u_plant(k,1:p) = u(1:p)';
    u_plant(nit+k,1:p) = u(p+1:end)';
    F_plant(k,1) = u(1);
    P_plant(k,1) = u(p+1);
    % F_plant(k,1:p) = u(1:p)';
    % P_plant(k,1:p) = u(p+1:end)';
    V_plant(k,1) = y(5);
    x_plant(k,1) = y(4);
    nu_plant(k,:) = nu';
%     ef_plant(k,1) = exitflag;
%     fval_plant(k,1) = -fval; % modeled cost
    cost_plant(k,1) = y(4)*y(5);
    x_model(k,1) = z0_model(4);
    % y_model(k,:) = [z0_model; V0]';
    z_model(k,:) = z0_model(1:3)';
%     nu_model(k,:) = nu_m';
    fb_plant(k,1) = fb_k;
    try
    save(filename);
    catch
        keyboard
    end
end
end



function [fmin, branchlist] = objfun (u, z0_model, V0, dt, A, c, kla, Km, GUR_max, ...
    OUR_max, Ki, Zgl_feed, nit, p, kp, maninter, GUR_sig, OUR_sig, kla_sig, fb_k,
tab_tol, eti, opt, t2d, t5d)
% structure of z0_model = [zgl, zo2, zac, x]
V0n = V0;
z0_model_n = z0_model;
ti = eti/dt;

% y_plant_p = zeros(nit,5);
% nu_plant_p = zeros(nit,4);

% fb_ke = fb_k;
fb_ke = 0;
tl = 1;
mu = 0;% initial condition of mu always be zero.
bran_tol = 0.05;
% tab_tol = 0.05;
F0 = u(1);P0 = u(p+1);

% nit >= 3
```

```
lentime = nit - kp + 1;
time_conv = 2; % every 2 time interval converge branchs

GUR_range = GUR_max*[1-GUR_sig 1+GUR_sig];
OUR_range = OUR_max*[1-OUR_sig 1+OUR_sig];
kla_range = kla*[1-kla_sig 1+kla_sig];

% [y, nu, Basic, mu] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c, ...
%     kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu)

y = [z0_model; V0];
time = 0;
brsubtab = branchSubtab(time, [], [], [], [], 1, [], [], [], [], [], y, []);
% time, up_leaf, mid_leaf, low_leaf, list_conbine, possible, x_grad, prob_dis, unit,
nodes, maxmin_nu_mu, maxmin_y, data
brsubtab.Prev = 0;
brsubtab.Mid = [];
brsubtab.Next = 1;
branchlist = branchList(time, brsubtab, 1, 1, []);
% time, branchsub, pos_list, scenario

i = 2;
branchlist(i) = branchlist(i - 1);

time = branchlist(i - 1).Time + 0.1;
branchlist(i).Time = time;

branchlist(i).BranchSubtab = [];
branchlist(1).Wholerange = z0_model;
z0_model = branchlist(i - 1).BranchSubtab.MaxMin_y(1:4);
X_k = z0_model(4);
pre = 1;

[brsubtab_list] = gene_branch_node_of1(time, z0_model, V0, F0, P0, dt, A, c, ...
    kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki, Zgl_feed, mu, X_k,
pre, bran_tol, tab_tol, opt, t2d, t5d);
branchlist(i).BranchSubtab = brsubtab_list;

%generate scenario

branchlist(i).Wholerange = branchlist(i).BranchSubtab.MaxMin_y;
list_sub = branchlist(i).BranchSubtab;
lensub = length(list_sub);
branchlist(i).Pos_list = ones(lensub, 1);
branchlist(i).Scenario = 1;
% for j = 1:lensub
%     branchlist(i).Pos_list(j) = list_sub(j).Possible;
%     branchlist(i).Scenario(j, end) = j;
%     branchlist(i).Scenario(j, 1: end - 1) = branchlist(i -
1).Scenario(list_sub(j).Prev, :);
% end

y = branchlist(i).Wholerange(:, 2);
z0_model = y;
z0_model(4, :) = z0_model(4, :) + fb_ke;
V0 = y(5);

branchlist(1) = branchlist(2);
branchlist(2) = [];
cur_subtabmap = [];
```

```
fti = 1;

for i = 2:lentime

    if mod((i - 1), maninter)==0
        tl = tl+1;
        F0 = u(tl);P0 = u(p+tl);
    end
    lenbran = length(branchlist(i - 1).BranchSubtab);
    branchlist(i) = branchlist(i - 1);
    branchlist(i).BranchSubtab = [];
    time = branchlist(i - 1).Time + 0.1;
    branchlist(i).Time = time;
    sump = 0;

    %% only for worst case branch
    if branchlist(i - 1).Pos_list <= tab_tol
        z0_model = branchlist(i - 1).Wholerange(1:4, :);
        V0 = branchlist(i - 1).Wholerange(5, :);
        n = [2 1];
        for j = 1:2
            nn = n(j);
%             [y, ~, ~, mu] = plant_dynamics(z0_model(:, j), V0(j), F0, P0, dt, A, c,
...
%                 kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu);
            [y, ~, ~, mu] = plant_dynamics(z0_model(:, j), V0(j), F0, P0, dt, A, c,
...
            kla_range(nn), Km, GUR_range(nn), OUR_range(nn), Ki, Zgl_feed, mu);
            branchlist(i).Wholerange(:, j) = y;
        end
    else
        %%

        for j = 1:lenbran
            % lenbfr = length(branchlist(i).BranchSubtab);
            pre = j;
%             [subtablist, sump] = gene_sub_tab(time, z0_model, V0, F0, P0, dt, A, c,
...
%                 kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki,
Zgl_feed,...
%                 mu, branchlist(i - 1).BranchSubtab(j), pre, bran_tol, tab_tol,
sump);
            [subtablist, sump, cur_subtabmap] = gene_sub_tab_wb(time, z0_model, V0,
F0, P0, dt, A, c, ...
                kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki,
Zgl_feed,...
                mu, branchlist(i - 1).BranchSubtab(j), pre, bran_tol, tab_tol, sump,
fti, cur_subtabmap, opt, t2d, t5d);

            if mod((i + 1), ti)==0
                fti = 1;
            else
                fti = 0;
            end

            branchlist(i).BranchSubtab = [branchlist(i).BranchSubtab; subtablist];
        end

        % generate scenario
        lensub = length(branchlist(i).BranchSubtab);
        branchlist(i).Pos_list = zeros(lensub, 1);
        branchlist(i).Scenario = zeros(lensub, i);
        yylistmi = zeros(5, lensub);
        yylistma = zeros(5, lensub);

        for j = 1:lensub
            branchlist(i).BranchSubtab(j).Possible =
branchlist(i).BranchSubtab(j).Possible/sump;
            branchlist(i).Pos_list(j) = branchlist(i).BranchSubtab(j).Possible;
            branchlist(i).Scenario(j, end) = j;
            % branchlist(i).Scenario(j, 1: end - 1) = branchlist(i -
1).Scenario(branchlist(i).BranchSubtab(j).Prev, :);
            branchlist(i).Scenario(j, 1: end - 1) = 1;
            yylistmi(:, j) = branchlist(i).BranchSubtab(j).MaxMin_y(1:5, 1);
            yylistma(:, j) = branchlist(i).BranchSubtab(j).MaxMin_y(1:5, 2);
        end

        branchlist(i).Wholerange = [min([yylistmi, yylistma], [], 2) max([yylistmi,
yylistma], [], 2)];
        branchlist(i).Wholerange(4, :) = fliplr(branchlist(i).Wholerange(4, :));
        y = branchlist(i).Wholerange(:, 2);
       %% only for whole branch method

        % if rem(i, time_conv) == 0 && lensub > 1
        %     [branchlist(i)] = branch_conv(branchlist(i), branchlist(i).Wholerange);
        % end
        % y = branchlist(i).Wholerange(:, 2);
       %% only for worst case branch method
        branchlist(i).Pos_list =
branchlist(i).BranchSubtab.List_conbine.Possible*branchlist(i).Pos_list;
        %%
        z0_model = y;
        z0_model(4, :) = z0_model(4, :) + fb_ke;
        V0 = y(5);
    end %%%only for worst case branch method
end
fmin_wb = - (y(4) + fb_k)*y(5);% max cost = final_x*final_V


%% nominal part
y = [z0_model_n;V0n];%initial plant (Glc,O2,Aci,X,V)
% structure of z0_model = [zgl, zo2, zac, x]
% y_plant_p = zeros(nit,5);
% nu_plant_p = zeros(nit,4);
tl = 0;
mu = 0;% initial condition of mu always be zero.
for k = kp:nit
    if mod((k-1), maninter)==0
        tl = tl+1;
        F0 = u(tl);P0 = u(p+tl);
    end
    [y, ~, ~, mu] = plant_dynamics(z0_model_n, V0n, F0, P0, dt, A, c, ...
        kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu);
    % Reinitialise z0_model and V0 for next time step
    z0_model_n = y(1:4);
    V0n = y(5);
    % store the plant dynamics (not essential)
    % nu_plant_p(k,:) = nu';
    % y_plant_p(k,:) = y';
end
fmin_n = - (y(4) + fb_k)*y(5); % max cost = final_x*final_V

%%
```

```
% worst case
fmin = fmin_n*(1 - tab_tol) + fmin_wb*tab_tol;
% fmin = fmin_wb;
end


function [brsubtab_list] = gene_branch_node_of1(time, z0_model, V0, F0, P0, dt, A, c,
...
    kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki, Zgl_feed, mu, X_k,
pre, bran_tol, tab_tol, opt, t2d, t5d)


% Anew = [A; A(1:2,:); -A(3,:)];
Anew = [A(1:2,:); -A(3,:)];

bmax = [GUR_range(2)*(z0_model(1)/(Km + z0_model(1)));...
    OUR_range(2); ...
    100; ...
    (F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt)/X_k; ...
    (kla_range(2)*(0.21 - z0_model(2)) - F0/V0*z0_model(2) + z0_model(2)/dt)/X_k;...
    (-F0/V0*z0_model(3) + z0_model(3)/dt)/X_k];

% GUR_range(1)*(z0_model(1)/(Km + z0_model(1) + (z0_model(1)^2)/Ki)
bmin = [GUR_range(1)*(z0_model(1)/(Km + z0_model(1)));...
    OUR_range(1); ...
    100; ...
    (F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt)/X_k; ...
    (kla_range(1)*(0.21 - z0_model(2)) - F0/V0*z0_model(2) + z0_model(2)/dt)/X_k;...
    (-F0/V0*z0_model(3) + z0_model(3)/dt)/X_k];

rand_b = [bmin bmax];

[cur_subtabmap, ~, ~] = propagate_rhs_5d([rand_b(1:2, :);rand_b(4:6, :)], opt, t2d,
t5d);

yrange = zeros(5,2);
brsubtab_list = [];

for i = 1:length(cur_subtabmap)
    list_conbine = cur_subtabmap(i);
    possible = cur_subtabmap(i).Possible;
    sump = 0;

    if possible >= tab_tol
        sump = sump + possible;
        brsubtab = branchSubtab(time, [], [], [], list_conbine, possible, [], [], [],
[], [], [], []);
        % time, up_leaf, mid_leaf, low_leaf, list_conbine, possible, x_grad, prob_dis,
unit, nodes, maxmin_nu_mu, maxmin_y, data
        brsubtab_list = [brsubtab_list; brsubtab];

        for j = 1:2 %leaves
            kla = kla_range(j);
            nu = cur_subtabmap(i).MaxMin(1:4, j);
            %%%avoiding unnecessary eliminatating upper bound of nu. mu
            bnew = [(F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt)/X_k; ...
                (kla*(0.21 - z0_model(2)) - F0/V0*z0_model(2) +
z0_model(2)/dt)/X_k;...
                (-F0/V0*z0_model(3) + z0_model(3)/dt)/X_k];
            % [numin, ~, ~] = simplex_tab(-c,Anew,bnew,[],[],[]);
            [numin, ~] = linprog( - c, Anew, bnew,[],[],zeros(1, 4), nu' + 1e-9, opt);
            %%%
            % mu = cur_subtabmap(i).MaxMin(5, j);
            % mu = - max(sum(nu), mu);
            nu = min([nu, numin], [], 2);
            mu = - sum(nu);

            Anu_gl = A(1,:)*nu; Anu_o2 = A(2,:)*nu; Anu_Ac = A(3,:)*nu;
            tspan = [0 dt];
            z0 = [z0_model; V0];
            if (z0(1)<=0 && z0(2)<=0 && z0(3)<=0)
                z = [0 0 0 z0(4) V0];
            else
                [~,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2, Anu_Ac,
...
                    -mu, Zgl_feed, F0, P0), tspan, z0);
                %   [t,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2,
Anu_Ac, ...
                %       -mu, Zgl_feed, F0, P0), tspan, z0);
            end
            y = z(end,:)';
            if (z(end,1) <= 1e-6) % check for O2 concentration
                y(1,1) = 1e-6;
            end
            if (z(end,2) <= 1e-6) % check for O2 concentration
                y(2,1) = 1e-6;
            end
            if (z(end,3) <= 1e-6) % check for O2 concentration
                y(3,1) = 1e-6;
            end
            yrange(:, j) = y;

        end
        maxmin_y = [yrange(:,2), yrange(:, 1)];
        possible = brsubtab_list(end).Possible;

        brsubtab_list(end).Mid_leaf = Leaves(time, [], [], [], possible, [], [],
maxmin_y);
        % time, num, prev, lenu, possible, br_node_list, maxmin_nu_mu, maxmin_y

        brsubtab_list(end).MaxMin_y = maxmin_y;
        %         brsubtab = branchSubtab(time, [], [], [], list_conbine, possible,
[], [], [], [], []);
        % time, up_leaf, mid_leaf, low_leaf, list_conbine, possible, x_grad, nodes,
maxmin_nu_mu, maxmin_y, data
        brsubtab_list(end).Prev = pre;
        brsubtab_list(end).X_grad = maxmin_y(1:5, 2) - maxmin_y(1:5, 1);

        [prob_dis, unit, nodes] = pro_dist(brsubtab_list(end).X_grad, maxmin_y(1:5,
1));

        brsubtab_list(end).Prob_dis = prob_dis;
        brsubtab_list(end).Unit = unit;
        brsubtab_list(end).Nodes = nodes;

    end % if 1 == length(current_tabmap)

end

for i = 1:length(brsubtab_list)
```

```
    brsubtab_list(i).Possible = brsubtab_list(i).Possible / sump;
end

%% only for find worst case branch

lensub = length(brsubtab_list);
yylistmi = zeros(5, lensub);
yylistma = zeros(5, lensub);

for j = 1:lensub
    % brsubtab_list.BranchSubtab(j).Possible =
brsubtab_list.BranchSubtab(j).Possible/sump;
    % brsubtab_list.Pos_list(j) = brsubtab_list.BranchSubtab(j).Possible;
    yylistmi(:, j) = brsubtab_list(j).MaxMin_y(1:5, 1);
    yylistma(:, j) = brsubtab_list(j).MaxMin_y(1:5, 2);
end

Wholerange = [min([yylistmi, yylistma], [], 2) max([yylistmi, yylistma], [], 2)];
Wholerange(4, :) = fliplr(Wholerange(4, :));

for j = 1:lensub
    if brsubtab_list(j).MaxMin_y(4, 2) == Wholerange(4, 2)
        maint = j;
        break
    end
end

brsubtab_list = brsubtab_list(maint);

%     brsubtab_list(j).MaxMin_y = Wholerange;
brsubtab_list.Possible = brsubtab_list.Possible*sump;


end


function [prob_dis, unit, nodes] = pro_dist(grad, miny)
dgrad = grad./7;
nodes = zeros(5,5);
for i = 1:5
    miny = miny + dgrad;
    nodes(:, i) = miny;
end
prob_dis = {ones(1, 5)*(1/5), ones(1, 5)*(1/5), ones(1, 5)*(1/5), ones(1, 5)*(1/5)};
unit = abs(grad./5);
end


function [brsubtab_list, sump, cur_subtabmap] = gene_sub_tab_wb(time, ~, V0, F0, P0,
dt, A, c, ...
    kla, Km, GUR_max, OUR_max,GUR_range,OUR_range, kla_range, Ki, Zgl_feed, mu,
BranchSubtab, pre, bran_tol, tab_tol, sump, fti, cur_subtabmap, opt, t2d, t5d)
% worst case branch
% sub_pos_list = [];
pos = BranchSubtab.Possible;
%% posible of subtab
% Anew = [A; A(1:2,:); -A(3,:)];
Anew = [A(1:2,:); -A(3,:)];
```

```
% [z0_model, X_k] = cubcut(BranchSubtab, 1);
    z0_model = BranchSubtab.MaxMin_y(1:3, 2);
    X_k = BranchSubtab.MaxMin_y(4, 2);

bmax = [GUR_range(2)*(z0_model(1)/(Km + z0_model(1)));...
    OUR_range(2); ...
    100; ...
    (F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt)/X_k; ...
    (kla_range(2)*(0.21 - z0_model(2)) - F0/V0*z0_model(2) + z0_model(2)/dt)/X_k;...
    (-F0/V0*z0_model(3) + z0_model(3)/dt)/X_k];


% [z0_model, X_k] = cubcut(BranchSubtab, 2);
    z0_model = BranchSubtab.MaxMin_y(1:3, 1);
    X_k = BranchSubtab.MaxMin_y(4, 1);

bmin = [GUR_range(1)*(z0_model(1)/(Km + z0_model(1)));...
    OUR_range(1); ...
    100; ...
    (F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt)/X_k; ...
    (kla_range(1)*(0.21 - z0_model(2)) - F0/V0*z0_model(2) + z0_model(2)/dt)/X_k;...
    (-F0/V0*z0_model(3) + z0_model(3)/dt)/X_k];


rand_b = [bmin bmax];
rand_b = [rand_b(1:2, :);rand_b(4:6, :)];


if fti == 1
    [cur_subtabmap, ~, ~] = propagate_rhs_5d([min(rand_b, [], 2) max(rand_b, [], 2)],
opt, t2d, t5d);
%
% else
%     [tab_m] = simp_prop_rhs_5d([min(rand_b, [], 2) max(rand_b, [], 2)], opt, t2d,
t5d);
%     for i = 1:2
%         cur_subtabmap(i).MaxMin = tab_m{i};
%     end
end

brsubtab_list = [];

lencur = length(cur_subtabmap);
poslist = zeros(lencur, 1);
for i = 1:lencur
    poslist(i) = cur_subtabmap(i).Possible;
end
maxpos = max(poslist);
if maxpos < tab_tol
poslist(poslist == maxpos) = tab_tol;
end
%
for i = 1:lencur
    if poslist(i) >= tab_tol
        poslist(i) = poslist(i)*pos;
        sump = sump + poslist(i);
        yrange = zeros(5,2);

        list_conbine = cur_subtabmap(i);
        brsubtab = branchSubtab(time, [], [], [], list_conbine, poslist(i), [], [],
[], [], [], [], []);
        % time, up_leaf, mid_leaf, low_leaf, list_conbine, possible, x_grad, prob_dis,
unit, nodes, maxmin_nu_mu, maxmin_y, data
        brsubtab_list = [brsubtab_list; brsubtab];

        splist = zeros(7, 5, 2);
        spnodes = [BranchSubtab.MaxMin_y(:, 1), BranchSubtab.Nodes,
BranchSubtab.MaxMin_y(:, 2)];

        % for j = 1:7 %Up and Low leaves
        for j = 1:6:7 %Up and Low leaves
            z0_model = spnodes(1:4, j);
            X_k = z0_model(4);

            for k = 1:2
                kla = kla_range(k);
                nu = cur_subtabmap(i).MaxMin(1:4, k);
                %%%avoiding unnecessary eliminatating upper bound of nu. mu
                bnew = [(F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt)/X_k; ...
                    (kla*(0.21 - z0_model(2)) - F0/V0*z0_model(2) +
z0_model(2)/dt)/X_k;...
                    (-F0/V0*z0_model(3) + z0_model(3)/dt)/X_k];
                % [numin, ~, ~] = simplex_tab(-c,Anew,bnew,[],[],[]);
                [numin, ~] = linprog( - c, Anew, bnew,[],[],zeros(1, 4), nu' + 1e-9,
opt);
                %%%
                % mu = cur_subtabmap(i).MaxMin(5, k);
                % mu = - max(sum(nu), mu);
                nu = min([nu, numin], [], 2);
                mu = - sum(nu);

                Anu_gl = A(1,:)*nu; Anu_o2 = A(2,:)*nu; Anu_Ac = A(3,:)*nu;
                tspan = [0 dt];
                z0 = [z0_model; V0];
                if (z0(1)<=0 && z0(2)<=0 && z0(3)<=0)
                    z = [0 0 0 z0(4) V0];
                else
                    [~,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2,
Anu_Ac, ...
                        -mu, Zgl_feed, F0, P0), tspan, z0);
                    %   [t,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2,
Anu_Ac, ...
                    %       -mu, Zgl_feed, F0, P0), tspan, z0);
                end
                y = z(end,:)';
                if (z(end,1) <= 1e-6) % check for O2 concentration
                    y(1,1) = 1e-6;
                end
                if (z(end,2) <= 1e-6) % check for O2 concentration
                    y(2,1) = 1e-6;
                end
                if (z(end,3) <= 1e-6) % check for O2 concentration
                    y(3,1) = 1e-6;
                end
                yrange(:, k) = y;

                maxmin_y = [yrange(:,2), yrange(:, 1)];

            end
            splist(j, :, :) = maxmin_y;
            if 7 == j
```

```
                brsubtab_list(end).Up_leaf = Leaves(time, [], [], [], [], [], [],
maxmin_y);
                % time, num, prev, lenu, possible, br_node_list, maxmin_nu_mu,
maxmin_y
            end
            if 1 == j
                brsubtab_list(end).Low_leaf = Leaves(time, [], [], [], [], [], [],
maxmin_y);
                % time, num, prev, lenu, possible, br_node_list, maxmin_nu_mu,
maxmin_y
            end
        end

        maxmin_y_list = [brsubtab_list(end).Low_leaf.MaxMin_y(:, 2),
brsubtab_list(end).Up_leaf.MaxMin_y(:, 1)];

        % maxmin_y = [min(maxmin_y_list, [],2), max(maxmin_y_list, [],2)];

        brsubtab_list(end).MaxMin_y = [brsubtab_list(end).Low_leaf.MaxMin_y(:, 1),
brsubtab_list(end).Up_leaf.MaxMin_y(:, 2)];

        brsubtab_list(end).Mid_leaf = Leaves(time, [], [], [], [], [], [],
maxmin_y_list);
        % time, num, prev, lenu, possible, br_node_list, maxmin_nu_mu, maxmin_y

        %% Probability distribution for each element

        xgrad = brsubtab_list(end).MaxMin_y(1:5, 2) - brsubtab_list(end).MaxMin_y(1:5,
1);
        brsubtab_list(end).X_grad = xgrad;
        brsubtab_list(end).Prev = pre;

        dgrad = brsubtab_list(end).X_grad ./ 7;
        nodes = zeros(5,5);
        miny = brsubtab_list(end).MaxMin_y(1:5, 1);
        for j = 1:5
            miny = miny + dgrad;
            nodes(:, j) = miny;
        end
        brsubtab_list(end).Nodes = nodes;

    end

end % if 1 == length(current_tabmap)

%% find worst case branch

    lensub = length(brsubtab_list);
    yylistmi = zeros(5, lensub);
    yylistma = zeros(5, lensub);

    for j = 1:lensub
        % brsubtab_list.BranchSubtab(j).Possible =
brsubtab_list.BranchSubtab(j).Possible/sump;
        % brsubtab_list.Pos_list(j) = brsubtab_list.BranchSubtab(j).Possible;
        yylistmi(:, j) = brsubtab_list(j).MaxMin_y(1:5, 1);
        yylistma(:, j) = brsubtab_list(j).MaxMin_y(1:5, 2);
    end

    Wholerange = [min([yylistmi, yylistma], [], 2) max([yylistmi, yylistma], [], 2)];
    Wholerange(4, :) = fliplr(Wholerange(4, :));

    for j = 1:lensub
        if brsubtab_list(j).MaxMin_y(4, 2) == Wholerange(4, 2)
            maint = j;
            break
        end
    end
    try
    brsubtab_list = brsubtab_list(maint);
    catch
       keyboard
    end

    %     brsubtab_list(j).MaxMin_y = Wholerange;
    brsubtab_list.Possible = brsubtab_list.Possible*sump;

end



function [current_tabmap,cur_subtabmap, efl] = propagate_rhs_5d(rand_b, opt, t2d, t5d)
% A = [      0    9.4600    9.8400   19.2300
%     35.0000   12.9200   12.7300         0
%     0    9.4600    9.8400   19.2300
%     35.0000   12.9200   12.7300         0
%     39.4300         0   -1.2400  -12.1200];
% b_norm = [1
%     1
%     1
%     1
%     1];    % 10
% % using 1e-10 instead of 0 in this part
% rand_b1 = [0 2]; %min and max of b. max = rand_b(1)*b_range
% % use 1e-6 instead of 0 for avoiding an error of lcon2vert

% rand_b = [0 1; 0 1; 0 1; 0 1; 0 1]; %delta[b1 b2 b4 b5 b6]'
efl = 0;
if min(rand_b(3:5,1)) <= 50

    A = [0    9.4600    9.8400   19.2300
        35.0000   12.9200   12.7300         0
        0    9.4600    9.8400   19.2300
        35.0000   12.9200   12.7300         0
        39.4300         0   -1.2400  -12.1200];
    c = [-1;-1;-1;-1];
    Tab_distribution = t5d;

else

    A = [0    9.4600    9.8400   19.2300
        35.0000   12.9200   12.7300         0];
    c = [-1;-1;-1;-1];
    Tab_distribution = t2d;
    rand_b = rand_b(1:2, :);
end

lenb = length(rand_b(:,1));

b_spacecon = [eye(lenb) rand_b(:, 2); -eye(lenb)  -rand_b(:, 1)];
```

```
lenF = length(Tab_distribution);

whole_vol = prod(rand_b(:, 2) - rand_b(:, 1));

if whole_vol <= 1e-9 %a small epsilon for lower dim range
    rand_b(:, 2) = rand_b(:, 2) + 1e-4;
    b_spacecon = [eye(lenb) rand_b(:, 2); -eye(lenb)  -rand_b(:, 1)];
    whole_vol = prod(rand_b(:, 2) - rand_b(:, 1));
end

% finding possible subspace
tab_map = Tab_distribution;

error = [];

lenf = length(A(1, :)) + 1;

for i = 1:lenF
    lenL = length(Tab_distribution{i}.List_edge);
    if lenL >= 1
        tab_vol = 0;
        empty_tab = [];
        max_s = [];
        min_s = [];
        for j = 1:lenL
            Space_con = Tab_distribution{i}.List_edge(j).Space_con;
            subpace_con = [Space_con; b_spacecon];
            A_space = subpace_con(:, 1:end - 1);
            b_space = subpace_con(:, end);
            try
                % [V,nr,ef] = con2vert(A_space,b_space);
                [V,nr,~,ef] = lcon2vert_ef(A_space,b_space,[],[],[],[]);
            catch
                ef = 5;
                ef1 = 1;
                error = [error; i, j];
            end
            if 0 == ef && ~isempty(V)
                At = A_space(nr,:);
                bt = b_space(nr,:);
                tab_map{i}.List_edge(j).Space_con = [At, bt];
                tab_map{i}.List_edge(j).Verties = V;
                [K,vol] = convhulln(V);
                tab_map{i}.List_edge(j).Vol = vol;
                tab_map{i}.List_edge(j).K = K;
                tab_vol = tab_vol + vol;
                tab_map{i}.List_edge(j).Possible = vol/whole_vol;
                % calculate for the worst case
                sol_fun_con = tab_map{i}.List_edge(j).Basic_sol_v;
                cost_fun_con = c'*sol_fun_con;
                % [maxmin_mat] = maxmin(At, bt, sol_fun_con, cost_fun_con, opt);
                [maxmin_mat] = maxmin_v(V, sol_fun_con, cost_fun_con);
                max_s = [max_s, maxmin_mat(:, 2)];
                min_s = [min_s, maxmin_mat(:, 1)];
                tab_map{i}.List_edge(j).MaxMin = maxmin_mat;
                % mid/morm point
                for k = 1:lenb
                    tab_map{i}.List_edge(j).Mid_point(k) =
(max(V(:,k))+min(V(:,k)))/2;
                end
            else
                empty_tab = [empty_tab; j];
            end
        end
        tab_map{i}.List_edge(empty_tab) = [];
        tab_map{i}.Possible = tab_vol/whole_vol;
        % calculate for the worst case for base tab
        if tab_map{i}.Possible > 0
            maxmin_wt = zeros(lenf, 2);
            for k = 1:lenf
                maxmin_wt(k, 1) = min(min_s(k, :));
                maxmin_wt(k, 2) = max(max_s(k, :));
            end
            tab_map{i}.MaxMin = maxmin_wt;
        end
    end
end

empty_tab_map = [];
for i = 1:lenF
    if 0 == tab_map{i}.Possible
        empty_tab_map = [empty_tab_map; i];
    end
end
tab_map(empty_tab_map) = [];
current_tabmap = [];

for i = 1:length(tab_map)
current_tabmap = [current_tabmap; tab_map{i}];
end

%% generate subspace tab map
cur_subtabmap = [];

end


function [maxmin_mat] = maxmin(A, b, sol_fun_con, cost_fun_con, opt)
fm = [sol_fun_con(:, 1:end - 1); cost_fun_con(:, 1:end - 1)];
[lenf, lens] = size(fm);
maxmin_mat = zeros(lenf, 2);

for i = 1:lenf
    f = fm(i, :)';
    % [bmin, tmin, ~] = simplex_tab(f, A, b, [], [], []);
    [~,tmin] = linprog(f,A,b,[],[],zeros(1, lens),[],opt);
    % [bmax, tmax, ~] = simplex_tab(- f, A, b, [], [], []);
    [~,tmax] = linprog( - f,A,b,[],[],zeros(1, lens),[],opt);
    if i == lenf
        maxmin_mat(i, :) = [tmax, - tmin];
        %actuall [min max] since c = -c
    else
        maxmin_mat(i, :) = [tmin, - tmax];
    end
    %convert from -c condition
end
end
function [maxmin_mat] = maxmin_v(V, sol_fun_con, cost_fun_con)
fm = [sol_fun_con(:, 1:end - 1); cost_fun_con(:, 1:end - 1)];
[lenf, ~] = size(fm);
maxmin_mat = zeros(lenf, 2);
```

```
for i = 1:lenf
    f = fm(i, :)';
    sol = V*f;
    tmax = max(sol);
    tmin = min(sol);
    if i == lenf
        maxmin_mat(i, :) = abs([tmax, tmin]);
        %actuall [min max] since c = -c
    else
        maxmin_mat(i, :) = abs([tmin, tmax]);
    end
    %convert from -c condition
end
end


function [y, nu, Basic, mu] = plant_dynamics(z0_model, V0, F0, P0, dt, A, c, ...
    kla, Km, GUR_max, OUR_max, Ki, Zgl_feed, mu)
% structure of z0_plant = [zgl, zo2, zac, x]
% LP Model solution
% n = length(c);
if (z0_model(1)<=1e-3 && z0_model(3)<=1e-3)
    X_k = z0_model(4);
else
    X_k = z0_model(4) + (-mu*z0_model(4) - (F0-P0)/V0*z0_model(4))*dt;
end

Anew = [A; A(1:2,:)*X_k; -A(3,:)*X_k];
b = [GUR_max*(z0_model(1)/(Km + z0_model(1)));...
    OUR_max; ...
    100; ...
    F0/V0*(Zgl_feed - z0_model(1)) + z0_model(1)/dt; ...
    kla*(0.21 - z0_model(2)) - F0/V0*z0_model(2) + z0_model(2)/dt;...
    F0/V0*z0_model(3) + z0_model(3)/dt; ];
% - F0/V0*z0_model(3) + z0_model(3)/dt; ];
[nu, mu, Basic] = simplex_tab(-c,Anew,b,[],[],[]);
Anu_gl = A(1,:)*nu; Anu_o2 = A(2,:)*nu; Anu_Ac = A(3,:)*nu;
tspan = [0 dt];
z0 = [z0_model; V0];
if (z0(1)<=0 && z0(2)<=0 && z0(3)<=0)
    z = [0 0 0 z0(4) V0];
else
    [~,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2, Anu_Ac, ...
        -mu, Zgl_feed, F0, P0), tspan, z0);
%   [t,z] = ode45(@(t,z) model_dynamics(t,z, kla, Anu_gl, Anu_o2, Anu_Ac, ...
%       -mu, Zgl_feed, F0, P0), tspan, z0);
end
y = z(end,:)';
if (z(end,1) <= 1e-6) % check for O2 concentration
    y(1,1) = 1e-6;
end
if (z(end,2) <= 1e-6) % check for O2 concentration
    y(2,1) = 1e-6;
end
if (z(end,3) <= 1e-6) % check for O2 concentration
    y(3,1) = 1e-6;
end
end
```