University of Alberta

Computationally effective optimization methods for complex process control and scheduling problems

by

Yu Yang

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Process Control

Department of Chemical and Materials Engineering

Dedicated to

My Parents and Teachers

Abstract

Motivated by the soaring production cost, intensive competitions and public attentions on environmental issues, how to reduce the operational cost, raise the profit and enhance the operational safety attracts tremendous interests in the chemical and petroleum industry. Since the regulatory control strategy may not achieve such rigorous requirements, higher level process control activities, such as production planning, real time optimization (RTO) and multi-variable control are more frequently taken into account. Moreover, to attain the better performance, process control engineers often consider plant-wide operations rather than unit-based actions. As a result, both dynamic and discrete optimization techniques for the large scale problem nowadays play a more important role in the industry than before. Even the classical optimization based techniques, such as model predictive control (MPC), have seen considerable successes in many practical applications. However, they are still suffering from computational issues in the circumstances of a large-scale plant, complex dynamic system or the short sampling time period. Furthermore, these traditional optimization techniques usually employ the deterministic formulations, but often become unsuitable for uncertain dynamics. Hence, this thesis is mainly concerned with developing computationally effective algorithms to solve practical problems arising from those high level process control activities and highly affected by the disturbances.

Approximate dynamic programming (ADP) is one of the most efficient computational frameworks to handle large-scale, stochastic dynamic optimization problems. While a large number of successful cases based on ADP have been reported, several critical issues, including risk management, continuous state space representation and the stability of the control policy, prohibit its application in process control. To overcome these shortcomings,

- We developed a systematic approach to extract the probabilistic model from

the operational data of a plant-wide system and proposed a risk-sensitive RTO approach based on ADP.

- An innovative procedure for designing control Lyapunov function (CLF) and robust control Lyapunov function (RCLF) is presented for a nonlinear control affine system under the input and state constraints.
- Based on the well-designed RCLF, a mixed control strategy, combining the advantages of MPC and ADP, is proposed to handle the stability issue of the ADP control scheme.

In addition to dynamic optimization, another focus of this research is the discrete optimization. Considering mixed integer linear programming (MILP) becomes increasingly common in the planning and scheduling of the chemical production, it is worthwhile to explore a more efficient algorithm for solving this NP hard problem. A modified Benders decomposition approach, featured by its tighter cutting plane, is presented to accelerate the solution procedure.

All the proposed approaches are demonstrated and evaluated by several benchmark examples. The comparisons with previous works also show the superiority of the suggested methods.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors Dr. Jong Min Lee and Dr. Fraser Forbes, not only for their financial support, but also the encouragement, guidance and patience. Dr. Lee gives me sufficient freedom to work out on my idea and spends much time on improving my writing skills, presentation ability and insight to the practical process control, which are considerably beneficial to my future career. Dr. Forbes is so nice to take care of my study and graduation when I was in trouble. The suggestions provided by him are always constructive and helpful to my research.

Besides my supervisors, another two professors in the Business school, Dr. Armann Ingolfsson and Dr. Ray Patterson are also the people that I would like to thank. They bring me into the operation research field and help me to figure out some great ideas, now becoming part of my thesis.

My colleagues in the department make my life pretty enjoyable. I should thank XinGuang Shao, Fei Qi and Fan Yang for their patience to answer my questions and the discussion with them is always insightful. LuNing Wang, ShaoFeng Yang, Wei Zhu, Nancy, Cindy Yin, JiHua Gong and Sara are my best friends to help me handle all kinds of worries in my life. Qi Li, Hua Deng, Yan Liu, Hector, Kartik and Vanket are the best organizers to hold the party and make the travel plan. I never feel alone and boring in the Edmonton when I stay with them.

In addition, I should thank my roommates, Chuan Wang, YunLong Xia and YiFeng Zhang. Staying and talking with them is always interesting.

Last but not the least, I should show my deepest appreciation to my parents, JinPing Yang and Ling Li, who bring me to this world and raise me up. More importantly, they build my character to overcome any difficulties in my life and the goodness to get on with all my friends and teachers.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of this thesis is to develop some computationally efficient frameworks for scheduling, real time optimization (RTO), and advanced process control problems. To this end, modified approximate dynamic programming (ADP) strategies and an improved cutting plane approach are brought forward in this research.

1.1 Motivation

Process control activities can be classified in the form of a hierarchy [108] as shown in Fig. 1.1. Although each level executes various functions, more and more leading companies in the oil and chemical industries are prone to integrate those high level activities, including the plant-wide advanced control, real-time optimization and production scheduling together to form a unified solution [47]. In this integrated framework, one needs to solve a large-scale optimization problem with time constraints. For instance, the plant-wide RTO considers the economic objectives of the entire process which may be governed by more than 200 differential and algebraic equations (DAE) [54]; the advanced controller, even the unit-based, should offer reliable and effective actions to regulate highly nonlinear plant to track its setpoint in the presence of state or input constraints and reject those outer disturbances and the model mismatch, within a few seconds or minutes; To solve the scheduling problem is very time consuming or even infeasible using the routine methods and current computational resources, no matter working out the production plan over the future 10 to 20 years or the management of a fleet with thousands of drivers in the dynamic environment [96].

The cutting-edge techniques in these areas are revisited here and their computational inefficiency are illustrated as follows:

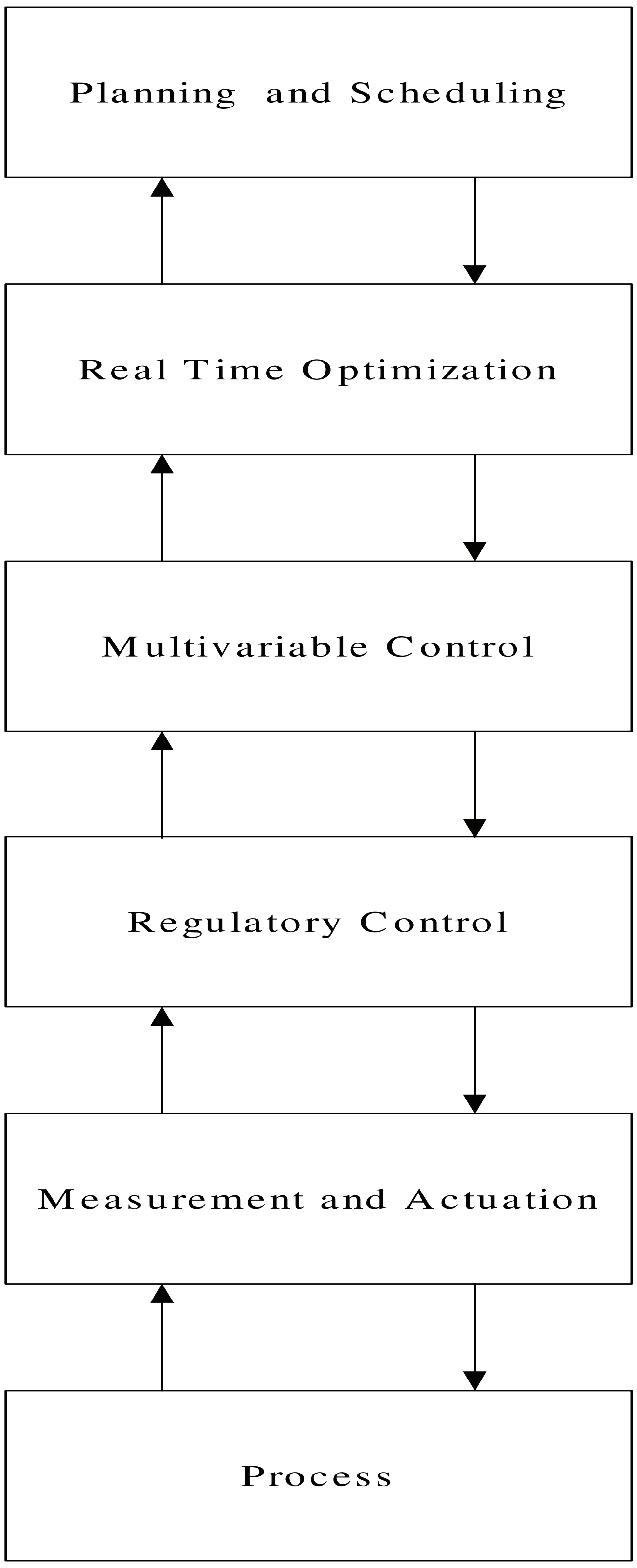

Figure 1.1: Hierarchy of process control activities

1. Model predictive control (MPC): MPC is one of the most celebrated advanced control techniques in the industry because it is fully capable of regulating the highly interactive multi-variable process subject to constraints. In general, it relies on the linear dynamic model, but in order to achieve good performance, the more accurate first principle model should be adopted and long prediction and control horizons are needed. Moreover, there has been a trend to extend the unit-based MPC to the plant-wide application [23]. The resulting optimization problem is a large-scale nonlinear program, whose on-line computation is non-trivial. Indeed, several large-scale nonlinear programming (NLP) solvers, such as IPOPT [124] and SNOPT [42], have been developed to advocate such plant-wide optimization. However, owing to the presence of model uncertainty, disturbances and measurement noise in practice, the sophisticated MPC schemes, such as Min-Max MPC [1, 45, 75, 70] or stochastic MPC [27], may need to be implemented online, which is NP-hard in general. Hence, the pursuit of the computational ability will be endless.

2. Steady state and Dynamic RTO: Currently, the steady state RTO dominates the applications in the industry due to its simplicity. However, the steady state RTO overlooks the dynamic transitions, which may last quite a few hours in some cases, thus leading to a sub-optimal solution. Moreover, it can be of no use in periodic operational systems, such as simulated moving beds (SMBs) [122], where the steady steady is never or seldom reached. Consequently, a number of researchers have realized that the steady state RTO is insufficient for complex chemical processes [14, 110, 121]. The dynamic RTO considers the transition process by employing the dynamic system model and adjusts setpoints more frequently, thereby improving the economic performance. Nevertheless, similar with MPC, dealing with large-scale nonlinear system and predicting its evolvement in a long horizon is computationally demanding and may be not achievable by the current hardware and software resources [65].

3. Mixed integer linear programming (MILP): The MILP has received much attention in the field of scheduling and planning research where both the discrete and continuous variables coexist. The formulation of MILP is similar to that of linear programming (LP), except the integer decision variables.

Unfortunately, solving MILP is considerably more difficult than solving LP. It is generally a NP-hard problem, which means that possibly no polynomial-time algorithm exists to get the solution. The Branch-and-Bound, one of the most prevalent approaches, divides the solution space into many branches according to the value of the integer variables, and the bound of each branch is calculated to determine whether to reject this branch or do the deeper exploration. Although this method does not need to enumerate all possible solutions (nodes), checking a small fraction is still time consuming for the large-scale problem.

Recently, the approximate dynamic programming (ADP) was introduced into the process control field by [69, 71] to alleviate the computational complexity in solving complex dynamic optimization problems. Whereas dynamic RTO as well as the advanced control strategies are formulated as a multi-stage decision problem, dynamic programming (DP) achieves the same goal by solving a single-stage optimization problem, thus saving computational time significantly. However, DP also suffers from the "curse of dimensionality" [7], which means that the optimization becomes much more complicated as the number of the variables increases. The ADP circumvents these computational issues by replacing the true value function, i.e. how much reward or cost we can expect under a particular policy starting from state x, with an approximate one. Therefore, it may be one of the most suitable frameworks to obtain a nearly optimal solution for complex dynamic optimization problems.

In solving a large-scale MILP problem, the decomposition technique plays a significant role in improving the performance of conventional Branch-and-Bound. Because of the separable and sparse nature of those large-scale formulations, the whole problem usually can be divided into several sub-systems and solved separately. The sub-systems in essence generate simplified polyhedral approximations to the feasible region of the original problem, thus providing strong bound to facilitate the Branch-and-bound approach [39]. Although it is implemented in an iterative fashion, the total computing time can be far less than the conventional method.

Although ADP and decomposition method have a potential for solving large-scale and complex optimization problems that emerge in the high level process control activities, both of them still encounter challenges for the practical applications, which are discussed in the following.

1.2 Issues of the Existing Methods

In this section, outstanding issues of the existing ADP and decomposition approaches are discussed.

1.2.1 Issues in Current ADP Methodologies

As an efficient computational tool, ADP has been actively explored in the Machine Learning (ML) community during the last few decades, with various names, such as Reinforcement Learning (RL) and Neuro-dynamic programming. The rationale behind it is training each agent to learn the optimal action by reward and punishment through the experience, historical data or simulations, without specifying the relation of action and outcome directly[115].

In another direction, the scholars in operations research also recognize the importance of the ADP to solve the complicated planning problem, such as resource allocation, aircraft scheduling and network management. Usually, such kind of problem has a considerable number of decision variables and interacts with the highly dynamic environment. Solving the Bellman optimality equation explicitly in that case is intractable, whereas the simulation based method can approximate and infer the value function for the entire state space, thereby quickly yielding an sub-optimal solution. A complete tutorial on related work and contributions can be found in [96].

While the ADP is proposed by those computer or operation scientists to address the computational problems in conventional DP, i.e. the "curse of dimensionality", there are still some critical issues rendering this methodology impractical in the process control applications.

1. **State representation:** Different from most of tasks in the computer science and operations research, the number of possible state and input in the process control domain is infinite because the regulated plant is continuous. Moreover, some complicated processes usually are described by high dimensional equations, further raising the difficulties of the DP approach. Although the Hamilton-Jacobian-Bellman (HJB) equation is the counterpart of DP in continuous-time equation, this partial differential equation is difficult to solve. Discretization is the possible choice only for the low dimensional scenario, but infeasible for the real system. So far, the function approximation is one of the

most promising ways to derive the value function over the continuous space. However, how to choose the basis function and control the approximation error are still open questions.

2. **Closed-loop stability:** Although some ADP control strategies have been reported for their successful applications, most of them cannot explicitly demonstrate their stability in control. The conventional DP based regulation approach attains the optimal control law, whose value function is also a control Lyapunov function, thus guaranteeing the stability. On the other hand, the general ADP scheme does not have this property. The traditional model-based stability analysis is not applicable to the model-free version of the ADP. Even though the ADP can improve the control performance in some cases, any methodology in the absence of stability is not acceptable in the industry. Thus, it is worthwhile to establish the stability analysis for the ADP control scheme.

3. **Exploration vs. Exploitation:** One of challenges in ADP is the dilemma between exploration and exploitation. In order to get more profits, the agent will select the best action based on the current experience; however, this greedy choice may lose the opportunity to learn the new knowledge and find the better actions in the future. Some sophisticated methods to balance these two aspects can be found in [4, 5, 20, 36].

   However, the divergence between the chemical industry and computer science just lies in the attitude to the exploration. To pursue the better performance, the computer scientists are more likely to explore unknown states or inputs region even undertaking some risks. On the contrary, chemical engineers, who emphasize the safety, cannot accept such risks. Hence, it is necessary to develop a risk-sensitive exploration scheme for the ADP.

4. **Data quality:** The data collected from the historical operations is usually corrupted by the measurement noise and concentrated within some particular regions of the state space. The direct use of these data without any pre-processing may lead to large errors in the approximation. Moreover, most of function approximation based ADP strategies cannot reliably generalize the approximate value function given the sparse data area. Therefore, the

application of the learned value function in these regions should be very cautious.

1.2.2 Issues in Current Decomposition Methods for Scheduling Problem

Although many commercial softwares have been released to handle large-scale MILP problems, the state-of-the-art decomposition method, which highly relies on the structure of the problem itself, is still worth further studying to speed up the algorithm. The typical decomposition methods, including Danzig-Wolfe [29] and Benders partition [9], share a similar idea. It first divides the original formulation into master problem (MP) and sub-problems (SPs), then start solving the MP and send its primal or dual solution to each sub-block. The cutting planes, generated from the SPs, are added to the MP to refine its feasible area sequentially. By repeating this procedure, one can finally obtain the exact solution. This framework is neat but more thorough research should be done to improve the quality of constraints and reduce the quantity of total cutting planes. Clearly, the deep cuts contribute to accelerating the algorithm; however, how to determine and find such qualified cut is non-trivial. Some works can be referred [25, 34], but more efficient mechanism is still needed.

In summary, this research devotes to overcoming the disadvantages mentioned above, which make the ADP and decomposition approaches more suitable for the process control activities.

1.3 Outline of the Thesis

The remainder of this thesis is organized as follows:

In Chapter 2, the fundamental concepts about Markov chain model and approximate dynamic programming are introduced. Several classical methodologies for DP and ADP are illustrated to compare their pros and cons, respectively.

In Chapter 3, in order to enhance the stability of advanced control techniques, we present a fractional programming formulation and its solution strategy for the design of general form of control Lyapunov function (CLF) and quadratic robust control Lyapunov function (RCLF) to (1) guarantee the closed-loop stability of a control affine system in a specified region of state space; and (2) enlarge the the region of attraction (ROA). Without restrictive assumptions found in previous approaches, the fractional programming problem is reformulated in a recursive manner. Instead of considering the true point-wise objective function, its bound is constructed and updated continuously to assist the optimization. Several examples, including CSTR and two tank systems are used to demonstrate the effectiveness of this design method.

Throughout chapters 4,5 and 6, we present the tailored ADP and decomposition approaches for scheduling, RTO and advanced control, respectively.

In Chapter 4, in the presence of the well designed RCLF, a mixed control strategy, combining the merits of ADP and MPC, is proposed to improve the control performance. Given the simulations or historical operational data, the state space is divided into several parts and their local value function approximators are constructed, respectively, to reduce the estimation error. The nearly optimal action is selected greedily based on the parameterized value function. In order to avoid aggressive control actions in the unexperienced regions of the state space or make the exploration safe, the regulator can be switched to the MPC at the right moment. The RCLF, as an attractor, exists both in ADP and MPC formulations to guarantee the stability. The CSTR example with outer disturbances is used to test this framework.

In Chapter 5, a novel algorithm for constructing a probabilistic model based on historical operation data and performing dynamic optimization for plant-wide control applications is developed. The proposed approach consists of applying a self organizing map (SOM) for identifying representative plant operation modes based on a discounted infinite horizon cost and ADP techniques for learning an optimal policy. A quantitative measure for risk is defined in terms of transition probability, and a systematic guideline for striking balance between risk and profit in decision making is provided with a mathematical proof. The efficacy of the proposed approach is illustrated on an integrated plant consisting of a reactor, a storage tank, and a separator with a recycle loop and the Tennessee Eastman challenge problem. The algorithm is useful for learning an improved policy and reducing risk in plant operation when the plant-wide first principle model is difficult to obtain and uncertainties affect operation performance significantly.

In Chapter 6, since the mixed integer linear programming (MILP) model plays an important role in the scheduling and planning problem and Benders decomposition can handle such kind of formulation with special structure more efficiently, we present a novel strategy for speeding up the classical Benders decomposition for large-scale MILP problems. The proposed method is particularly useful when the optimality cut is difficult to obtain. A ratio of distances from a feasible point to an infeasible point is used as a metric to determine the tightest constraint, thus improving the convergence rate. Application of the proposed approach to a multi-product batch plant scheduling problem shows substantial improvement both in the computational time and the number of iterations.

Finally, Chapter 7 summarizes the contribution of this thesis and discusses possible future studies.

Chapter 2

Markov Decision Process and Approximation Dynamic Programming

2.1 Markov Decision Process

Markov decision process (MDP) [98] is a general modeling framework for multi-stage optimal control problems under uncertainty. The elements of a Markov chain model consist of: state, action, reward and transition probability. Given these fundamental concepts, we can describe the general Markov decision process, i.e. at each of discrete time steps, the state x_t is observed and used to select an action u_t, which then causes the system to change the state to x_{t+1} with some probabilities and emits a bounded reward or cost, $r(x_t, u_t)$.

Note that one of the most significant assumption of the MDP is that system endows the Markov property. Namely, its dynamics is probabilistic under uncertainty and

$$p(x_{t+1}|x_t, u_t) = p(x_{t+1}|x_t, u_t, x_{t-1}, u_{t-1}, \ldots, x_1, u_1) \tag{2.1}$$

which indicates that the probability of being in state x_{t+1} at time $t+1$ is only determined by state x_t and the action u_t implemented at time t. Hence, the transition probability from state i to state j, under the control policy π, can be directly expressed as p_{ij}^{π}. This premise can considerably simplify our optimization framework because the solver only needs to consider the current state, not the history. Fortunately, this assumption holds in many practical cases because an augmented state can be defined to incorporate all useful historical information to determine the future state. For instance, the general discrete time nonlinear system

(2.2) meets this assumption:

$$x_{t+1} = f(x_t, u_t, w_t) \tag{2.2}$$

where x_t, u_t and w_t are state, input and disturbance at time t, respectively.

The objective of the optimization is to find a policy that maximizes the total reward received or minimizes the total cost all over the whole time horizon. Here the policy is a mapping from the current state to action. Then, in order to evaluate how good the policy is, the value function should be calculated for each state.

Specifically, the MDP can be categorized into two cases: finite-horizon and infinite-horizon problems [98].

In the infinite-horizon case, the discount factor $\gamma \in (0,\ 1)$ is usually introduced to ensure the expected total reward to be finite, which gives a trade-off between immediate and delayed reward.

$$J^{\pi}(x_0) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \middle| x = x_0 \right\} \tag{2.3}$$

Or in another way, without the discount factor, the absorbing state can be proposed:

Definition 1. *(Absorbing state x_b): A state is called absorbing if it is impossible to leave this state under the policy π, namely, $p^{\pi}_{ii} = 1$ and $p^{\pi}_{ij} = 0$ for $i \neq j$.*

The MDP will never terminate until it reaches the absorbing state. From the initial state x_0, assume that process terminates in the n^{th} time period under the policy π, then the value function of x_0 is

$$J^{\pi}(x_0) = E\left\{r(x_0, u_0) + r(x_1, u_1) + r(x_2, u_2) + \ldots + r(x_{n-1}, u_{n-1}) + J_{x_b}\right\} \tag{2.4}$$

where J_{x_b} is the terminal reward/cost. It is worth noting that even if we employ the absorbing state as the stopping criterion, the MDP can still be viewed as an infinite horizon scheme. In that case, the system always transfers to the same state with zero cost.

For the finite-horizon problem, in order to guarantee the expected N steps rewards to be optimal, the value function for each state should be time varying. This topic is beyond the scope of this research. Hence, it will not be mentioned in the thesis. In the following sections of this chapter, we restrict our attentions to the infinite-horizon problem with discount factor.

No matter finite or infinite horizon cases, the value function always determines a partial ordering over policies, whereby $\pi_1 \geq \pi_2$ if and only if $J^{\pi_1}(x) \geq J^{\pi_2}(x),\ \forall x$. An optimal policy, π^*, achieves $J^{\pi^*}(x) \geq J^{\pi}(x)\ \forall \pi, x$. All optimal policies share the same optimal value function, $J^*(x) = \max_\pi J^\pi(x)$.

2.2 Dynamic Programming

In a very few cases dynamic optimization of MDP can be solved analytically, for example, the Ricatti equations for optimal control of linear systems with quadratic costs [10]. Application of MDP to the general case is feasible through dynamic programming (DP), which involves iteratively solving the following Bellman optimality equation:

$$J^*(x_t) = \max_{u_t} E\left\{r(x_t, u_t) + \gamma J^*(x_{t+1}(x_t, u_t))\right\} \tag{2.5}$$

where J^* denotes the optimal value function. The Bellman optimality operator $T : \Re^N \to \Re^N$ is defined for vector $f \in \Re^N$, such that

$$Tf(s) = \max_u \{r(x, u) + \gamma \sum_{x'} p(x'|x, u) f(x)\} \tag{2.6}$$

where x' is the nest stage state from state x using action u. Then, (2.5) indicates that the optimal value function is the fixed point of the Bellman optimality operator. Namely,

Proposition 2. *[10]* $TJ^*(x) = J^*(x)$

Solving (2.5) directly is nontrivial, however, some recursive formulations can be employed to obtain the solution efficiently.

2.2.1 Policy Evaluation

Before solving the Bellman equation and calculate the optimal value function, we first investigate how to determine the value function for a specified policy. This is named as *policy evaluation* in the dynamic programming literature.

Assume that there is a sequence of functions $J_0, J_1, \ldots, J_n$, which maps the state into a value under a control policy π. The initial function J_0 can be specified arbitrarily. Then, the next is obtained by using the operator T^π to update previous

J for every state:

$$J_{k+1}(x_t) = E\left\{r(x_t, u_t) + \gamma J_k(x_{t+1})\right\} \tag{2.7}$$
$$= r(x_t, u_t) + \sum_{x_{t+1}} P^{\pi}_{x_t x_{t+1}} \gamma J_k(x_{t+1}) \tag{2.8}$$

Let $k \to \infty$, one can prove the following proposition that the sequence will finally converge to a fixed point, which is the value function under policy π.

Proposition 3. *[10]* $J^{\pi}(x) = \lim_{k\to\infty}(T^{\pi}_k J)(x)$

2.2.2 Policy Improvement

Once the value function for a policy is obtained, one can apply the *policy improvement* procedure to get the new control input for each state.

For instance, to the state x_t, the action of policy π is u_t which yields the action value function $Q^{\pi}(x_t, u_t)$. Now, let us consider a better policy κ in terms of getting rewards, which is exactly the same with π except changing the input to a_t for state x_t. It means that

$$Q^{\kappa}(x_t, a_t) = E\{r(x_t, a_t) + \gamma J^{\pi}(x_{t+1})\} \geqslant Q^{\pi}(x_t, u_t) \tag{2.9}$$

Then one can prove that $J^{\pi}(x) \leqslant J^{\kappa}(x)$ holds for every state [115].

The above single state conclusion actually can be extended to the full state and action space. In fact, based on a known policy π, we can make use of Bellman optimality operator to achieve the best input greedily:

$$\kappa(x_t) = \arg\max_{u_t} E\{r(x_t, u_t) + \gamma J^{\pi}(x_{t+1})\} \tag{2.10}$$

where E means expectation. It is not difficult to check that Eq.(2.10) results in the input u satisfying Eq.(2.9). Thus, by employing this operator to all states, called a sweep, we can get a new policy that is better than the previous one.

2.2.3 Policy Iteration

Once the original policy is improved, the evaluation step is adopted to get the new value function again. By repeating this procedure, the value function will finally converge to the optimal one. The Fig 2.1 [115] illustrates the framework of policy iteration.

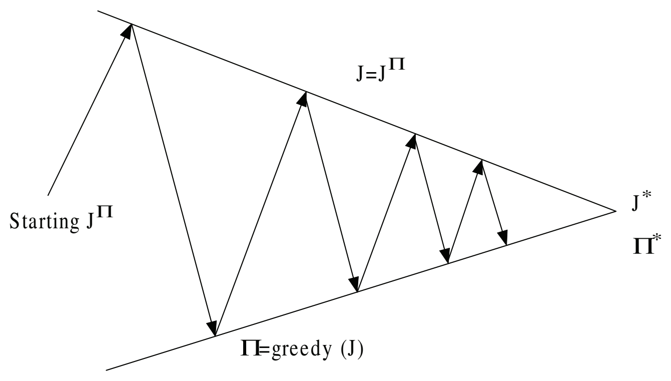

Figure 2.1: Graphical metaphor of policy iteration algorithm.

2.2.4 Value Iteration

If we skip policy evaluation and keep on using Bellman operator in the policy improvement step, then the improving process will continue until every state achieves its best value. It is called *value iteration.* The equation is:

$$J(x_t) = \max_{u_t} E\{r(x_t, u_t) + \gamma J(x_{t+1})\} \tag{2.11}$$

One can prove the following proposition for the convergence of value iteration.

Proposition 4. *$J^* = \lim_{N\to\infty} T_N J_0$, given any initial guess J_0.*

Compared with the policy iteration approach, this scheme omits the policy evaluation step, which saves much time on getting the exact value function for each policy. However, given the finite policy space, policy iteration can generate a finite sequence of control law with monotonicity to achieve an optimal one, whereas the value iteration converges in an infinite number of iterations [10]. Hence, how to select the scheme should depend on the structure of the problem and the requirement of the accuracy.

Though DP has been widely recognized as a feasible mean of solving general stochastic optimal control problems, many practical applications cannot be addressed with DP because of the following two major issues: 1) the computational requirement grows exponentially in the number of state variables, which is referred to as “curse-of-dimensionality,” and 2) the system dynamics or the model, is difficult to obtain. Therefore, we need to introduce the framework of approximate dynamic programming.

2.3 Approximate Dynamic Programming

Approximate dynamic programming (ADP) is a body of theories developed to address the modeling and computational issues of DP [69, 12, 115, 96]. In ADP, the optimal (or *nearly optimal*) value function/action value function is obtained by iteratively improving the initial estimate of the value function/action value function based on a set of sampled state points visited by simulation or real-operation data. One of the simplest methods in ADP is the temporal difference [115] learning of order 0 (TD(0)), which performs an update as follows:

$$\tilde{J}(x_t) \leftarrow \tilde{J}(x_t) + \alpha[r(x_t, u_t) + \gamma \tilde{J}(x_{t+1})] \tag{2.12}$$

where $\leftarrow$ means that the righthand side is used as a target for an update of the left-hand side. The $\alpha \in (0, 1)$ is the learning factor. The TD can be used for both model based and model free control. However, researches have developed a number of particular methods for these two cases. Thus, in the following parts, we will discuss them, respectively. Note that the policy gradient method, which searches the parameterized control strategy in the policy space directly [116], is also a useful approach for ADP with remarkable success in some applications. However, this thesis does not cover the related field.

2.3.1 Model-based ADP

Generally, the model is in favor of the application of the DP. Algorithms mentioned above can be used to get the optimal solution straightforwardly. However, the remaining problems for these model based methods include how to construct the accurate model and solve Bellman optimality equation efficiently, especially for the large- scale system. People in Artificial Intelligence (AI) community developed a class of methodologies named model-based reinforcement learning (RL). For example, the model is frequently updated from online experience in Dyna algorithm to improve the prediction [114]. Considering that the conventional MDP requires to know the exact state transition distribution, which is an idealized assumption, the Bayesian RL [95] considers the probability of possible unknown parameters instead of all transition dynamics to circumvent this premise. The prior distribution is used to reduce the trial and the posterior distribution is updated iteratively via the interactions between the agent and environment. The artificial neural networks

(ANNs) is also brought up to model the complex nonlinear system to facilitate RL [50].

Due to the difficulties of solving the Bellman equation, it is necessary to figure out some approaches to estimate the value function quickly at the cost of some reasonable accuracy loss. The real time dynamic programming (RTDP) [6] asynchronously updates the states that are encountered during trial-based MDP simulations. Thus, the RTDP may generate a good policy by only exploring a small fraction of the whole states. The nonparametric instance-based interpolation such as k-nearest neighbor [71] is used to estimated the value function of each state point. However, it needs to maintain all historical data, rendering a huge challenge to the memory system and slowing down the computation. The parametric methods generalize the sampled value function to the whole state space. Most of attentions focus on the linear approximators and it is believed that the choice of the representative regressor functions is the key to the successful application [66]. Thus, several kinds of basis function, such as radial-basis functions (RBFs), tile coding, decision trees, have been studied. The evolutionary selection of the basis is also developed to enable the efficient individual learning [128]. The gradient descent/ascent method or least square [19], combined with TD, is applied to estimate the parameters. However, the complete convergence theory for these function approximation methods is not well-established. Since an arbitrary small change in the value of an action may lead it to be selected or not, this discontinuity is believed to be the main obstacle to prove the convergence of function approximation methods [12].

2.3.2 Model-free ADP

One of the advantages of ADP in solving the Bellman equation is that it can be extended to the case where transition probability or process model is totally unknown. This is possible by encoding the optimal value function with state and action pair as follows:

$$Q^*(x_t, u_t) = E\left\{ r(x_t, u_t) + \gamma \max_{u_{t+1}} Q^*(x_{t+1}, u_{t+1}) \right\} \tag{2.13}$$

where the optimal value function and the optimal action-value function have the following relationship:

$$J^*(x) = \max_{u} Q^*(x, u) \tag{2.14}$$

Then we shall introduce two important model free ADP or RL approaches: Q-learning and SARSA.

SARSA

The TD prediction method can be extended to the state action value function. After every transition from state x_t to x_{t+1}, the following update is applied:

$$Q(x_t, u_t) \longleftarrow Q(x_t, u_t) + \alpha[r(x_t, u_t) + \gamma Q(x_{t+1}, u_{t+1}) - Q(x_t, u_t)] \tag{2.15}$$

where α is the learning rate; u_{t+1} is chosen according to the current control policy and state x_{t+1}. Since SARSA always attempts to evaluate the policy that is being used, it is an on-policy method. Its control algorithm can be constructed as follows:

Step 1: Initialize the $Q(x, u)$ for each state action pair.

Step 2: Initialize the state x in the beginning of each episode.

Step 3: Select the input u according to the state action value function Q.

Step 4: Take the u and receive the r and observe the new state x'.

Step 5: Choose the u' according to the Q.

Step 6: Update the Q by (2.15).

Step 7: $x \leftarrow x'$, $u \leftarrow u'$ and return to Step 4, until this episode is terminated.

Step 8: Return to Step 2, until the whole learning process is terminated.

Clearly, in order to achieve the optimal Q, a large number of episodes should be run to visit all state-action pairs many times. It is also worthwhile noting that we do not always choose the input greedily in Step 3. The philosophy behind it is to strike the balance between exploration and exploitation. Although the greedy input can achieve the best performance according to the current knowledge (exploitation), sometimes we still need to try some other inputs (exploration) to broaden the understanding of the system. The typical method to balance exploration and exploitation is the ϵ-greedy. For more detail, readers can refer to [115].

Q-learning [127]

The primary difference between the Q-learning and SARSA is their updating rule:

$$Q(x_t, u_t) \longleftarrow Q(x_t, u_t) + \alpha[r(x_t, u_t) + \gamma \max_{u_{t+1}} Q(x_{t+1}, u_{t+1}) - Q(s_t, u_t)] \qquad (2.16)$$

The u_{t+1} in (2.16) may be different from the true u_{t+1} applied in the control policy because sometimes the non-greedy action may be chosen by the policy (e.g ϵ-greedy). Thus, Q-learning is the off-policy method, which is easy to analyze and prove the convergence. The algorithm is very similar with SARSA, except the updating formulation. Both SARSA and Q-learning will converge to the optimal policy if all state-action pairs are visited infinitely often.

Chapter 3

Control Lyapunov Function Design

Stability is the first priority in industrial control applications. Hence, even the ADP has shown considerable potentials to improve control performance, it cannot be employed to regulate plants reliably until its rigorous stability analysis is well established. With the model, considering that the Lyapunov function is the most powerful tool to guarantee the control stability, this chapter dedicates to develop the design procedure for that function to assist further applications of the ADP.

3.1 Background

Although nonlinear models are commonly used to describe chemical process dynamics, design of controllers that stabilize such systems is not an easy task and has attracted much attention in the last several decades. Due to its generality, Lyapunov theory has served as the main tool for designing stable controllers [111], and two distinct design philosophies exist in Lyapunov based strategies. The first class of methodologies designs control Lyapunov function (CLF) and controller separately, where the CLF is first constructed and subsequently used to choose a control law [113, 53]. The second methods design the CLF and controller simultaneously, and among them is the classical Back-stepping design[62].

Simultaneous design methods may require restrictive assumptions for the model form [64]. In the separate design approaches, a large number of previous studies are concerned with improving control algorithms itself [33, 84, 134]. However, the main bottleneck to the success of these methods lies in the construction of

[*] A full version of this chapter has been submitted to the *Journal of Process Control*

[†] Abbreviated sections of this chapter were presented at DYCOPS [129], July 5-7, 2010

CLF. Moreover, the region of attraction (ROA), an invariant set characterizing the stabilizable area around the equilibrium, needs further investigation to guarantee the local stabilization, because global stabilization is too strict for practical applications.

Many previous studies have focused on enlarging ROA or improving decay rate in CLF design. One simple approach is to obtain a quadratic CLF by solving the Riccati equation associated with the linearized model. This often leads to a small ROA due to approximation errors. A polynomial CLF can be constructed by the sum of squares (SOS) programming [118]. This method involves nonconvex bilinear matrix inequality constraints and often yields local optimal solutions. A simulation-based approach is proposed to select a qualified CLF and an initial condition for the bilinear search strategy [120]. The derivation from density function to CLF is analyzed under certain assumption [99] and searching that function is a convex optimization problem solved by SOS [97]. However, all these SOS based methods can handle polynomial systems only, which are not suitable for highly nonlinear systems including a chemical reactor with temperature-dependent kinetics. SOS programming is extended to non-polynomial systems by variable transformation with algebraic constraints [91]. This method, however, neglects non-polynomial constraints, thereby probably resulting in a CLF incompatible with the original system.

Linear parameter varying (LPV) or quasi-LPV embedding approaches represent a nonlinear system by linear dynamics depending on scheduling variables and yield a CLF by solving a linear matrix inequality (LMI) problem for a large number of points in the scheduling variable space [52]. A nonlinear system can also be represented by a polytopic linear differential inclusion [18], and a quadratic Lyapunov function can be obtained using LMI. However, these approximate models may incorporate dynamics which do not belong to the original system and yield conservative solutions.

In chemical process operations, the range of initial states, usually determined by upstream units, can be estimated in advance. Once the ROA includes the region, which may be far from the equilibrium, stabilizing control is achieved. However, there are few studies solving this problem explicitly in the literature because even a CLF with the largest ROA may not contain the required state area. To this end, this work is mainly concerned with developing a systematic method to construct a CLF that stabilizes the states in a specified region without restrictive assumptions on the polynomial system as previous approaches. Construction of a CLF that has a ROA including the subset of state space is formulated as a constrained fractional programming problem. Since the objective and constraints are non-convex and non-smooth, a derivative-free, coordinate search method is proposed to find a sub-optimal solution. We also show that optimization along the coordinate can be implemented efficiently by making use of piece-wise linear property of the problem. The grid point checking for the moderate-size system is implemented as in [55] to guarantee the state points of our interest be included in the ROA. The proposed scheme can significantly reduce the number of checking points in the state space compared with the method in [55]. In addition, the proposed scheme is applicable to a wide class of CLFs, not restricted to the quadratic or polynomial form.

Furthermore, note that the disturbances, measurement noise and model mismatch widely exist in the real plant, it is necessary to extend the control Lyapunov function method to the stochastic cases. Namely, the Lyapunov function should provide a guideline to lead the process to approach the equilibrium even if the model cannot represent the reality and the disturbances occur unfavorably. To this end, [38] proposes the robust control Lyapunov function (RCLF) method. Moreover, given a prior RCLF, the concept "inverse optimal robust stabilization" [38] is presented to show the optimality of the "pointwise min-norm control law" for some meaningful Hamilton-Jacobi-Isaacs (HJI) differential game with guaranteed stability.

Nevertheless, constructing such RCLF is also an intractable task and deserves to be studied further, but very few literatures concern with the direct design method. The Lyapunov redesign procedure is discussed in [59]. The key idea of this approach is to employ the CLF as its RCLF by neglecting the uncertain structures. The success of this method depends on so-called matching condition, in which "the uncertainties enters the system through the same channel as the control variables"[38]. Clearly, this condition restricts the applicable scope of the Lyapunov redesign, which motivates people to consider this problem in the different viewpoint. Consequently, a more attractive methodology, "robust backstepping" for the system in strict feedback form or lower triangular form becomes the breakthrough in the nonlinear control history. Starting from the RCLF for a rather simple subsystem, a recursive design procedure is repeated to extend this function to the entire system. The quadratic RCLF over a set of transformed coordinates is proposed first, but [37] points out that such kind of control law suffers from the "hardening", i.e. the feedback control gain will increase to infinity once the system states follow some particular trajectories, thereby resulting in the high magnitude chattering. Also in the same work, a new RCLF was presented for the softer robust control policy to alleviate the control effort but keep the performance of the stabilization.

Although the robust backstepping can derive the control policy and RCLF simultaneously, it is restricted to the system with strict feedback form. In order to overcome this limitation, we employ the similar idea with the CLF design to propose a systematic method to establish the RCLF. Since the uncertainties or disturbances render the formulation more complex and difficult to obtain the optimal solution for the common form RCLF compared with the deterministic case, we only consider the quadratic RCLF in this research. In addition, considering that the non-vanishing nature of the disturbances lead the system hardly to stay in the equilibrium, we need to define a residual set, in which the state can fluctuate but never escape. Thereafter, how to characterize and bound this region should be also brought into the design procedure.

The rest of the paper is organized as follows: Section 3.2 provides preliminaries on CLF. Section 3.3 and 3.4 present the problem formulation for the CLF and RCLF, respectively. Section 3.5 and 3.6 present the derivative-free optimization approach to the fractional optimization. Sections 3.7 and 3.8 discuss some implementation issues and properties of the algorithm, respectively. Section 3.9 presents the continuous feedback control law for control affine systems with infinity norm bounded input. Section 3.10 presents two simulation examples, and concluding remarks are provided in Section 3.11.

3.2 Preliminaries

3.2.1 Control Lyapunov Function

Consider the nonlinear time-invariant affine in the input dynamical system given by

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{3.1}$$

with state $x(t) \in \Re^n$, $f(x(t)) \in \Re^n$, $g(x(t)) \in \Re^{n\times m}$, and bounded control input $u(t) \in \Re^m$ with $\underline{u}_i \leq u_i \leq \overline{u}_i$ where $i = 1, \ldots, m$. $\underline{u}_i$ and $\overline{u}_i$ are the lower and upper bound of the i^{th} component of the control signal, respectively.

Without particular claim, we assume that $\{0\}$ is the equilibrium point in this chapter, then a function $V : \Re^n \to \Re$ is a control Lyapunov function if V is positive except $V(0) = 0$ and the following inequality holds [3]:

$$\inf_{\underline{u}_i \leq u_i \leq \overline{u}_i} \frac{\partial V}{\partial x} \cdot (f(x) + g(x)u) < 0 \quad \forall x \neq 0, i = 1, \ldots, m \tag{3.2}$$

Condition (3.2) usually is hard to be satisfied in the entire state space. Hence, the Σ is defined to characterize regions where Lyapunov function can decay:

$$\Sigma = \left\{ x \middle| \inf_{\underline{u}_i \leq u_i \leq \overline{u}_i} \frac{\partial V}{\partial x} \cdot (f(x) + g(x)u) < 0, i = 1, \ldots, m \right\} \tag{3.3}$$

3.2.2 Region of Attraction

Since the global asymptotic stability cannot be achieved for most nonlinear systems with input constraints, region of attraction (ROA) is analyzed for the local asymptotic stability. For any initial state $x_i \in \Sigma$, the asymptotic stability can be guaranteed if the closed-loop state trajectory does not escape from Σ [84]. Hence, an invariant set of Σ is employed to estimate its region of attraction (ROA) such that there always exists a certain input trajectory that leads the system to a desired equilibrium point, assumed as $\{0\}$, for any initial point in the ROA. Specifically, we want to find the maximum value of $r > 0$ satisfying

$$V(\tilde{x}) \leqslant r \rightarrow \lim_{t \rightarrow \infty} |x(t, \tilde{x})| = 0 \tag{3.4}$$

where $x(t, \tilde{x})$ is the state at time t with initial state of $\tilde{x}$. Then this invariant set, $\Omega = \{x | \, V(x) \leqslant r_{\max}\} \subset \Sigma$, serves as an estimate of the ROA.

3.2.3 Robust Control Lyapunov Function (RCLF)

Due to the disturbance and model mismatch, stabilizing the process in a certain steady state without any offset is non-trivial. However, in such scenario, some control strategies can be applied to guarantee the final states within the neighbor region of the equilibrium rather than the fixed state. Hence, the robust control scheme of the nonlinear system in the presence of the uncertainties is the more practical choice for the real application. We first introduce several fundamental definitions and then investigate the concept of robust stabilization.

Definition 5. *[73]($\mathcal{K}$ function) A continuous function $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is said to be a $\mathcal{K}$ function if it is continuous, strictly increasing and $f(0) = 0$. It is called of class $\mathcal{K}_\infty$ if, in addition, it is unbounded.*

Definition 6. *[73] ($\mathcal{L}$ function) A continuous function $f : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is said to be a $\mathcal{L}$ function if it is continuous, strictly decreasing and* $\lim_{t\rightarrow\infty} f(t) = 0$.

Definition 7. *[73] ($\mathcal{KL}$ function) A continuous function $f : \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is said to be a $\mathcal{KL}$ function if $f(\cdot, t) \in \mathcal{K}$ for each fixed $t \in \Re^+$ and $f(r, \cdot) \in \mathcal{L}$ for each fixed $r \in \Re^+$,*

Let χ represents the state space and given a disturbance constraint W, we say the disturbance $w(x, t) : \chi \times \Re \rightarrow \Re$ is admissible if $\forall x \in \chi$ and $\forall t$, $w(x, t) \in \mathcal{W}$. Then, the following definition can be presented.

Definition 8. *[38] To control the system f, let Θ be a compact set containing the equilibrium. The control to f is* ***robustly globally uniformly asymptotically stable with respect to Θ (RGUAS-Θ)*** *when there exists $\beta \in \mathcal{KL}$ such that for all admissible disturbances, and initial conditions (x_0, t_0), all solutions $(x_0, t_0) \in \chi \times \Re$, all solutions $x(t)$ exist for all $t \geqslant t_0$ and satisfy*

$$|x(t)|_\Theta \leqslant \beta(|x_0|_\Theta, t - t_0) \tag{3.5}$$

for all $t \geqslant t_0$, where the notation $|.|_\Theta$ represents the Euclidean point to set distance function.

Here Θ indicates that the system will converge to the residual set around the equilibrium rather than a point because of the non-vanishing nature of the disturbances. We called the system **robustly stabilizable** when there exist the admissible control and compact set Θ such that this control solution to system is RGUAS-Θ.

Now, it is ready to introduce the RCLF. Let $\nu(\chi)$ denote a set of functions $V : \chi \times \Re \longrightarrow \Re^+$ such that there exist class $\mathcal{K}_\infty$ functions α_1 and α_2 satisfying:

$$\alpha_1(||x||) \leqslant V(x, t) \leqslant \alpha_2(||x||) \tag{3.6}$$

for all $(x, t) \in \chi \times \Re$.

Definition 9. *[38] A function $V \in \nu(\chi)$ is called a robust control Lyapunov function (RCLF) for the control affine system:*

$$\dot{x} = f(x) + \Delta f + (g(x) + \Delta g)u(t) \quad x \in \Re^n, u \in U \subset \Re^m, \Delta f, \Delta g \in \mathcal{W} \tag{3.7}$$

provided that there exists $C_\Theta \in \Re_+$ and a time invariant positive function α_V, such that

$$\inf_{u\in U} \sup_{\Delta f,\Delta g} \nabla V(x)[f(x) + \Delta f + (g(x) + \Delta g)u] + \alpha_V(x) < 0 \tag{3.8}$$

whenever $V(x) > C_\Theta$. The Δf and Δg are bounded uncertainties in set $\mathcal{W}$. To each component, there are $|\Delta f_j| \leqslant \Delta_j$, $|\Delta g_{ij}| \leqslant \Upsilon_{ij}$. U characterizes the admissible input set. $\alpha_V(x)$ is often viewed as the margin of negativity. C_Θ is the set level of residual set.

The global robust stability is difficult to satisfy for most of systems subject to input constraints. Hence, it is necessary to analyze the local robust stability. Define the region Γ_R, such that

$$\Gamma_R := \{x | \inf_{u\in U} \sup_{\Delta f,\Delta g} \nabla V(x)[f(x) + \Delta f + (g(x) + \Delta g)u] < 0\} \tag{3.9}$$

Then we need to find a new set Ω with the largest level C_Ω, such that if $V(x) < C_\Omega$ then $x \in \Gamma_R$. The estimation of ROA for rclf:V is $\Omega\backslash\Theta$.

Comparing the (3.3) with (3.8), it can be observed that the RCLF considers the worst effect of the uncertainties to the stability. This is conservative but logical because the derived Lyapunov function and control law should guarantee absolute stability around the equilibrium in the presence of any disturbance signal within the bound. Thus, the decay area of the RCLF should be much smaller than that of the CLF and more difficult to obtain. In the following of this research, we restrict our attention on an easier case, the quadratic robust control Lyapunov function.

In the **Definition 9**, we also suppose that the bounds of uncertainties, i.e. Δ_j and Υ_{ij} are known. However, this assumption may not be easy to satisfy in practice. Further research about quantifying a tight bound of uncertainties is deserved to be explored in the future.

3.2.4 Small Control Property

Small control property is useful in proving the existence of a continuous control law given a CLF [113, 76]. A CLF for Eq. (3.1) has the small control property if for any $\varepsilon > 0$, there is a δ satisfying:

$$0 \neq ||x|| < \delta \Rightarrow \exists ||u|| < \varepsilon, \ \frac{\partial V}{\partial x} \cdot (f(x) + g(x)u) < 0 \tag{3.10}$$

, where $||\cdot||$ can be any norm.

It is guaranteed that there exists a stable feedback policy, which is smooth at $\Re^n - \{0\}$, given a CLF [3]. In addition, if the CLF has the small control property, one can always find a control law that is continuous in $\forall x \in \Re^n$.

3.3 Control Lyapunov Function (CLF) Construction

Constructing a CLF with the largest possible ROA for all the states may yield a small ROA, thus limiting its application to practical problems. Oftentimes, it is only necessary to guarantee the stability for certain operating regions. Motivated by this, this section proposes an optimization based approach to obtain a CLF given a subset of states, which we refer to a *target region*.

For simplicity, and without loss of generality, the target region can be defined as a polytope $\Psi = \{x | Lx < H\}$ with constant matrix L and vector H. The objective is to design a CLF, the ROA of which includes the target region, and the following form of CLF is employed:

$$V(x) = \sum_{k=1}^{l} \alpha_k \phi_k(x) = \alpha^T \phi(x) \tag{3.11}$$

where α is the parameter vector and $\phi(x)$ is the basis function vector. In order to make this scheme general, there are no more restrictions on the basis function, such as semi-definiteness [55] or polynomial form, other than $\phi_k(0) = 0$ and $\frac{\partial \phi_k(x)}{\partial x}|_0 = 0$. According to the positive definiteness requirement of CLF, we also specify a region Ξ satisfying:

$$\Psi \subset \Xi \qquad \text{and} \qquad 0 \in \Xi \tag{3.12}$$

Then the family of possible CLFs is: $\{V | V(x) > 0, x \in \Xi,\ x \neq 0\}$.

Given V, consider the following optimization problem:

Problem 1

$$\begin{aligned}
&\max_{x,\hat{x}} \frac{V(\hat{x})}{V(x)} \\
\text{s.t.} \quad &\frac{\partial V}{\partial x} f(x) + \frac{\partial V}{\partial x} g(x) u \geq 0 \quad x \neq 0,\ \forall \underline{u}_i \leq u_i \leq \bar{u}_i, \quad i = 1, \ldots, m \\
&V(x) > 0 \quad x \neq 0 \\
&\hat{x} \in \Psi, \quad x \in \Xi
\end{aligned}$$

Problem 1 yields $\hat{x}$, the farthest point from the origin in the target region, and x, the closest point to the origin in the "no-decay" region based on the Lyapunov function measure, and thus characterizing the ROA of given CLF. If the objective value is less than unity (i.e., $V(\hat{x}) < V(x)$), the target region is included in the ROA. Hence, the design of a CLF that has a ROA including $\forall \hat{x} \in \Psi$ can be formulated as Problem 2, if such α exists. This is illustrated in Fig. 3.1.

Problem 2

Solve for α subject to

$$\Phi(\alpha) = \max_{x,\hat{x}} \frac{V(\hat{x})}{V(x)} < 1 \tag{3.13}$$

$$\frac{\partial V}{\partial x} f(x) + \frac{\partial V}{\partial x} g(x) u \geq 0 \quad x \neq 0, \ \forall \underline{u}_i \leq u_i \leq \bar{u}_i \tag{3.14}$$

$$V(x) > 0 \quad x \neq 0 \tag{3.15}$$

$$\hat{x} \in \Psi, \quad x \in \Xi \tag{3.16}$$

The constraint (3.14) characterizes state points where Lyapunov function cannot decay. It should be also noted that Eq. (3.14) is affine in u and only the minimum value of the left-hand side needs checking. Hence, only the lower and upper bounds of u are considered.

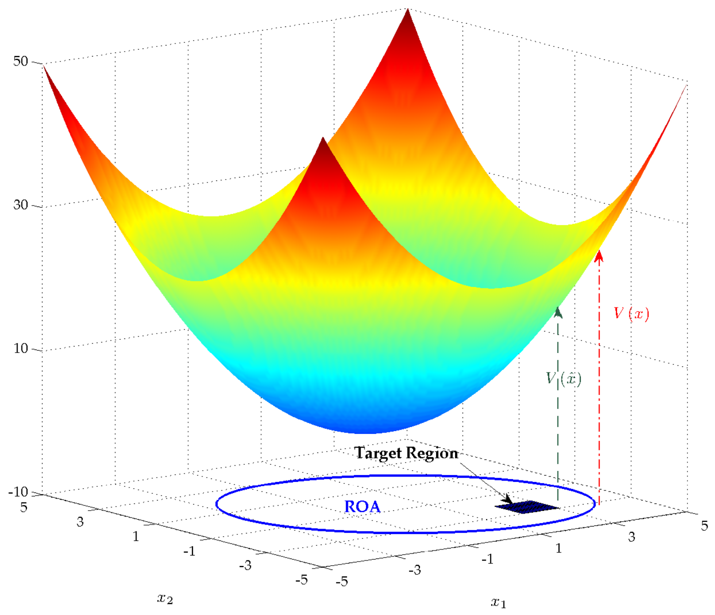

Figure 3.1: Illustration of Problem 1

One may consider using minimax techniques to obtain α, but it is difficult due to the non-differentiable nature of the constraint function. For example, convex approximation of minimax objective function is proposed in [60], but this method cannot deal with constraints on outer variables α. An interior point method is presented to solve continuous minimax optimization with constraints on outer parameters [103]. This, however, cannot handle coupled constraints like Eq. (3.14), where α and x exist together. Sampling state points and formulating it as LMI may yield a large-scale optimization problem. Furthermore, sampling is non-trivial because the shape of ROA cannot be determined in advance.

3.4 Robust Control Lyapunov Function (RCLF) Design

Due to the conservativeness of the RCLF formulation, a straightforward goal of the design algorithm is to construct the RCLF with largest ROA. Furthermore, note that the Θ in **Definition 8** determines an special set within which the state can fluctuate around the equlibrium, a small enough Θ can definitely improve the accuracy of the control. Thus, the presented design procedure should respect to both of aims, i.e. enlarging Ω and reducing Θ.

Assumed that the equilibrium is $\{0\}$, given a quadratic function with its set level C, i.e. $x^T Px = C$, its volume is proportional to $\frac{C}{\sqrt{|P|}}$ [18]. Both the ROA and final region Θ correspond their own C, which are defined as C_Ω and C_Θ, respectively. Then, the ROA and Θ can be depicted as:

$$\Omega : \{x | x^T Px \leqslant C_\Omega\} \qquad \Theta : \{x | x^T Px \leqslant C_\Theta\} \tag{3.17}$$

In order to integrate the two-fold objectives together, the following objective function for RCLF design is considered:

$$\Phi(P) = \frac{C_\Omega / \sqrt{|P|}}{C_\Theta / \sqrt{|P|}} = \frac{C_\Omega}{C_\Theta} \tag{3.18}$$

where C_Ω and C_Θ are determined by P. The (3.18) gets around the calculation of $|P|$, which is favorable to the optimization. Clearly, the best RCLF, owning the largest ROA and the smallest residual set Θ, should attain the maximum $\Phi(P)$.

Let us first consider the value of C_Ω. The estimation of the ROA for a given RCLF: $V = x^T Px$ can be determined by the following optimization formulation:

$$C_\Omega = \min_x x^T Px \tag{3.19}$$

$$\text{s.t.} \quad \inf_{u \in U} \sup_{\Delta f, \Delta g} \nabla V(x)[f(x) + \Delta f + (g(x) + \Delta g)u] \geqslant 0 \tag{3.20}$$

The constraint (3.20) derived from (3.78) characterizes the undecay point according to the Lyapunov function. Note that the function α_v in (3.78) has to be removed in this inequality, because the margin, α_v should be zero to find strict non-decayable point. By capturing the closest point in terms of Lyapunov measure satisfying (3.20), an ellipsoidal is derived within which any state is stabilizable.

As for the C_Θ, considering that the tolerable control error usually varies from different applications, thus we allow the Θ to tightly cover a user-specified region, in particular, a polyhedron: $\{0\} \in \tilde{\Theta}$ by solving the following optimization problem:

$$C_\Theta = \max_{\tilde{x}} \tilde{x}^T P \tilde{x} \tag{3.21}$$

$$\text{s.t.} \quad \tilde{x} \in \tilde{\Theta} \tag{3.22}$$

The resulting $\tilde{x}$ determines the C_Θ and it is not hard to see that $\tilde{\Theta} \subseteq \Theta$ because of the set level $C_\Theta \geqslant C_{\tilde{\Theta}}$. Due to the positive definiteness of P and polyhedron $\tilde{\Theta}$, (3.21) is a concave quadratic programming problem and the global optimal is attained in the vertex of the polyhedron.

Note that (3.19) and (3.21) are defined over the x and $\tilde{x}$, respectively, thus, we can combine those equations together according to the (3.18) to build the measure of the RCLF: $\Phi(P)$, which is a volume ratio of the ROA and residual set Θ.

Problem 3

$$\Phi(P) = \frac{\min_x x^T P x}{\max_{\tilde{x}} \tilde{x}^T P \tilde{x}} \tag{3.23}$$

$$\text{s.t.} \quad \inf_{u \in U} \sup_{\Delta f, \Delta g} \nabla V(x)[f(x) + \Delta f + (g(x) + \Delta g)u] \geqslant 0 \tag{3.24}$$

$$\tilde{x} \in \tilde{\Theta} \tag{3.25}$$

$$x^T P x \geqslant \max_{\tilde{x}} \tilde{x}^T P \tilde{x} \tag{3.26}$$

The new constraint (3.26) guarantees that any feasible x should be outside the Θ. Given the matrix P, the optimization over $\tilde{x}$ should be implemented first to derive the constraint (3.26), then the x can be calculated, successively.

To seek the optimal P which attains the largest $\Phi(P)$, one also needs to solve the following formulation:

Problem 4

$$\max_P \Phi(P) \tag{3.27}$$

$$\text{s.t.} \quad P \succ 0 \tag{3.28}$$

Here the constraint (3.28) indicates that P should be positive definite. The objective function of this max-min formulation is point-wise, thus hard to find the optimal solution. Before we develop the solving procedure, one of issues in the Problem 1 should be addressed first. The operators inf and sup in the constraint (3.24) are unfavorable to the optimization. However, given the boundary of the uncertainties and inputs, one can derive the following theorem:

Theorem 10. *With the admissible input set $U_j := \{\underline{u}_j \leqslant u_j \leqslant \overline{u}_j, j = 1, 2, \dots m\}$, the constraint (3.24) can be expressed explicitly and it is piecewise linear over the single element of P.*

Proof. Substituting the RCLF: $V = x^T P x$ into (3.24) and replacing $\{\Delta f, \Delta g\}$ by their bounds provided by **Definition 9**, there are

$$0.5\{\inf_{u_j \in U_j} \sup_{\Delta f, \Delta g} \frac{\partial V}{\partial x}(f(x) + \Delta f) + \frac{\partial V}{\partial x}(g(x) + \Delta g)u\} \tag{3.29}$$

$$= 0.5\{\min_{u_j \in U_j} \max_{\Delta f, \Delta g} \frac{\partial V}{\partial x}(f(x) + g(x)u) + \frac{\partial V}{\partial x}(\Delta f + \Delta g u)\} \tag{3.30}$$

$$= \min_{u_j \in U_j} \max_{\Delta f, \Delta g} x^T P(f(x) + g(x)u) + x^T P(\Delta f + \Delta g u) \tag{3.31}$$

$$= \min_{u_j \in U_j} x^T P(f(x) + g(x)u) + \sum_{j=1}^{n} \Delta_j |\sum_{i=1}^{n} x_i P_{ij}| + \sum_{i=1}^{n}\sum_{j=1}^{m} \Upsilon_{ij} |(x^T P)_i u_j| \tag{3.32}$$

$$= \sum_{j=1}^{n} \Delta_j |\sum_{i=1}^{n} x_i P_{ij}| + x^T P f(x) + \min_{u_j \in U_j} (x^T P g(x) u + \sum_{i=1}^{n}\sum_{j=1}^{m} \Upsilon_{ij} |(x^T P)_i||u_j|) \tag{3.33}$$

$$= \min_{u_j \in U_j} \sum_{j=1}^{m} \{\sum_{i=1}^{n} (x^T P)_i g(x)_{ij} u_j + \sum_{i=1}^{n} (\Upsilon_{ij} |(x^T P)_i|)|u_j|\}$$
$$+ \sum_{j=1}^{n} \Delta_j |\sum_{i=1}^{n} x P_{ij}| + x^T P f(x) \tag{3.34}$$

where the subscript i and ij represent the ith term of a vector and the item in the ith row, jth column of a matrix, respectively. The Υ_{ij} and Δ_j are the bound of disturbances.

It is worthwhile to point out that the value of different input can be determined separately. We further assume that $\overline{u}_j > 0$ because the positive input is usually allowable even under the physical limitation. Then, three terms, $\sum_{i=1}^{n} (x^T P)_i g(x)_{ij}$, $\sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i|$ and $\underline{u}_j$ determine the form of u_j in (3.34), which is problem-specific. To further investigate, different scenarios will be discussed, respectively.

For $\underline{u}_j \geqslant 0$, this is the simple case because the absolute operator can be left out. Then,

$$\begin{aligned} &\min_{u_j} \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} u_j + \sum_{i=1}^{n} (\Upsilon_{ij} |(x^T P)_i|) |u_j| \qquad (3.35) \\ = \; &\min_{u_j} (\sum_{i=1}^{n} (x^T P)_i g(x)_{ij} + \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i|) u_j \end{aligned}$$

It is not hard to derive the input:

$$u_j = \begin{cases} \underline{u}_j & \text{if } \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} + \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| \geqslant 0 \\ \overline{u}_j & \text{if } \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} + \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| < 0 \end{cases}$$

(3.35) also can be shown as a compact form:

$$\begin{aligned} &\frac{(\underline{u} - \overline{u})}{2} \left| \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| + |(x^T P)_i g(x)_{ij}| \right| \\ &+ \frac{(\underline{u} + \overline{u})}{2} \left(\sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| + |(x^T P)_i g(x)_{ij}| \right) \qquad (3.36) \end{aligned}$$

For $\underline{u}_j < 0$, the case is more complicated since other terms need to be considered. Note that $\sum_{i=1}^{n} (\Upsilon_{ij} |(x^T P)_i|) |u_j|$ is always positive, the sign of $\sum_{i=1}^{n} (x^T P)_i g(x)_{ij}$ directly determines the sign of u_j.

$$u_j \geqslant 0 \quad \text{if } \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} \leqslant 0 \qquad (3.37)$$

$$u_j \leqslant 0 \quad \text{if } \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} \geqslant 0 \qquad (3.38)$$

Based on the optimal conditions (3.37) and (3.38), the (3.34) can be written as:

$$\begin{aligned} &\min_{u_j} \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} u_j + \sum_{i=1}^{n} (\Upsilon_{ij} |(x^T P)_i|) |u_j| \qquad (3.39) \\ = \; &\min_{u_j} \{ - \left| \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} \right| + \sum_{i=1}^{n} (\Upsilon_{ij} |(x^T P)_i|) \} \, |u_j| \qquad (3.40) \end{aligned}$$

Then, the input is

$$u_j = \begin{cases} \underline{u}_j & \text{if} \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| - |\sum_{i=1}^{n} (x^T P)_i g(x)_{ij}| < 0, \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} \geqslant 0 \\ \overline{u}_j & \text{if} \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| - |\sum_{i=1}^{n} (x^T P)_i g(x)_{ij}| < 0, \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} \leqslant 0 \\ 0 & \text{if } \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| - |\sum_{i=1}^{n} (x^T P)_i g(x)_{ij}| \geqslant 0 \end{cases}$$

The result of (3.34) is

$$\begin{aligned} &\min_{u_j} \sum_{i=1}^{n} (x^T P)_i g(x)_{ij} u_j + \sum_{i=1}^{n} (\Upsilon_{ij} |(x^T P)_i|) |u_j| \qquad (3.41) \\ &= \frac{W_j}{2} \left(\sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| - |(x^T P)_i g(x)_{ij}| \right) - \frac{W_j}{2} \left| \sum_{i=1}^{n} \Upsilon_{ij} |(x^T P)_i| - |(x^T P)_i g(x)_{ij}| \right| \end{aligned}$$

where

$$W_j = \begin{cases} -\underline{u}_j & \text{if } \sum_{i=1}^{n}(x^T P)_i g(x)_{ij} \geqslant 0 \\ \overline{u}_j & \text{if } \sum_{i=1}^{n}(x^T P)_i g(x)_{ij} < 0 \end{cases}$$

In (3.35), (3.36), and (3.41), one can see that the function in each absolute value is linear to each single element of P. Hence, the whole function is still piecewise linear over the single element of P.

$\square$

This theorem shows the piecewise linear property of constraint (3.24) for single parameter, which plays an important role in our optimization procedure. The max operator in the constraint is left out and corresponding inputs are determined explicitly to favor the following optimization.

Moreover, we note that the significance of constraint (3.24) is twofold. Given the RCLF, i.e. P, inequality (3.24) characterizes the region of state in which the RCLF can dissipate. From another viewpoint, if the state x is fixed, (3.24) can determine a group of RCLF's that can stabilize the state x.

3.5 Derivative-free Optimization Approach for CLF

For searching the suitable α satisfying all constraints in Problem 2, we need to handle the point-wise, fractional function $\Phi(\alpha)$ and non-convex, non-differentiable constraints of Eq. (3.14). Hence, a derivative-free optimization method is proposed. Starting with the initial guess of α and its corresponding objective value $h^0 = \max_{x,\hat{x}} \frac{\sum_{k=1}^{l} \alpha_k \phi_k(\hat{x})}{\sum_{k=1}^{l} \alpha_k \phi_k(x)}$, the solver repeatedly samples the parameter space and calculates its objective function until Eq. (3.13) is satisfied.

Clearly, this sample and accept/reject procedure may waste lots of efforts to check the α^i but without any improvement. Thus, it falls into the brute force approach, which is not applicable even for a two-dimensional system. The smarter algorithm proposed in this chapter employs a recursive coordinate search manner. That is to say, only one parameter of α is considered rather than the whole parameter set in each iteration. In addition, the new α is accepted as long as the its objective value is lower than the previous round. Then, the solver will switch the sampling direction along another coordinate. Through this way, the sampling and search is implemented only in the one dimension, which is much easier than the whole space. Even more important, we will show that by taking advantage of the piecewise linear property, this one dimensional search can refine the sampling space efficiently, thereby accelerating the optimization, especially when the derivative information is not available.

We first employ the subscripts and superscripts to represent the sequence of parameters and iterations, respectively. Then, the fractional objective function of Problem 1 is modified by using Dinkelbach's method [30] to make the iteration task easier:

$$\Pi(x, \hat{x}, \alpha, h^{i-1}) = \sum_{k=1}^{l} \alpha_k \phi_k(\hat{x}) - h^{i-1} \sum_{k=1}^{l} \alpha_k \phi_k(x) \tag{3.42}$$

where i is the iteration index. Then, the following relationship holds:

$$\frac{V(\hat{x})}{V(x)} < h^{i-1} \leftrightarrows \Pi(x, \hat{x}, \alpha, h^{i-1}) = \sum_{k=1}^{l} \alpha_k \phi_k(\hat{x}) - h^{i-1} \sum_{k=1}^{l} \alpha_k \phi_k(x) < 0 \tag{3.43}$$

Hence, an improved solution for α is achieved if and only if

$$h^i = \max_{x,\hat{x}} \left. \frac{V(\hat{x})}{V(x)} \right|_{\alpha} < h^{i-1}$$

which is equivalent to:

$$\max_{x,\hat{x}} \Pi(x, \hat{x})|_{\{\alpha, h^{i-1}\}} < 0 \tag{3.44}$$

We also have $\Pi(\alpha)|_{\{x,\hat{x},h^{i-1}\}} \leq \max_{x,\hat{x}} \Pi(\alpha, x, \hat{x})|_{h^{i-1}}$, which forms a lower bound, for any α and $\{x,\ \hat{x}\}$ respecting Eqs. (3.14) and (3.15). This plays an important role in accelerating the algorithm since any α rendering $\Pi(\alpha)|_{\{x,\hat{x},h^{i-1}\}} > 0$ does not lower the objective value of Problem 1, and thus can be ignored.

It should be also noted that $\Pi(\alpha)|_{\{x,\hat{x},h^{i-1}\}}$, is linear in each component of α; therefore, only one element of α, say, α_k, is chosen as a decision variable in each iteration step. This way, the lower and upper bounds of this single parameter can be estimated easily. Within the confined region, denoted as Λ_k and satisfying the positive definiteness, i.e. (3.15), α_k is sampled and its corresponding Problem 1 is solved to estimate ROA and compare it with a target region.

Since solving Problem 1 or Eq. (3.44) is nontrivial for general form of CLF, grid checking is the most reliable way, but time-consuming. In order to avoid solving Problem 1 frequently, the number of sample points for α_k is reduced by eliminating the region in α_k space with $\Pi(\alpha_k)|_{\{x,\hat{x},h^{i-1}\}} > 0$. However, note that α_k in Problem 2 also needs to respect constraints (3.14), a new lower-bound function, Γ, further considering this constraint, defined in Λ_k, is designed given any x and $\hat{x}$:

$$\Gamma(\alpha_k)|_{\{x,\hat{x},h^{i-1}\}} = \begin{cases} \Pi(\alpha_k)|_{\{x,\hat{x},h^{i-1}\}} & \text{if Eq. (3.14) is satisfied for given } \{x, \hat{x}\} \\ -\beta & \text{Otherwise} \end{cases} \tag{3.45}$$

where β is a large positive number. Γ can serve as a lower bound of $\max_{x,\hat{x}} \Pi(\alpha_k, x, \hat{x})$ for any value of α_k within Λ_k. The lower bound is easy to handle than $\Pi(\alpha_k)|_{\{h^{i-1},x,\hat{x}\}}$. The procedure is depicted in Fig. 3.2. As different pairs of $\{x, \hat{x}\}$ are obtained in solving Problem 1 to evaluate h, corresponding Γ functions are also generated. In order to approach the true objective function more accurately, the $\tilde{\Pi}$ is constructed from the upper envelope union of different $\Gamma(\alpha_k)|_{\{x,\hat{x},h^{i-1}\}}$ over α_k as in Fig. 3.3. This makes it possible to sample α_k leading to an improved solution.

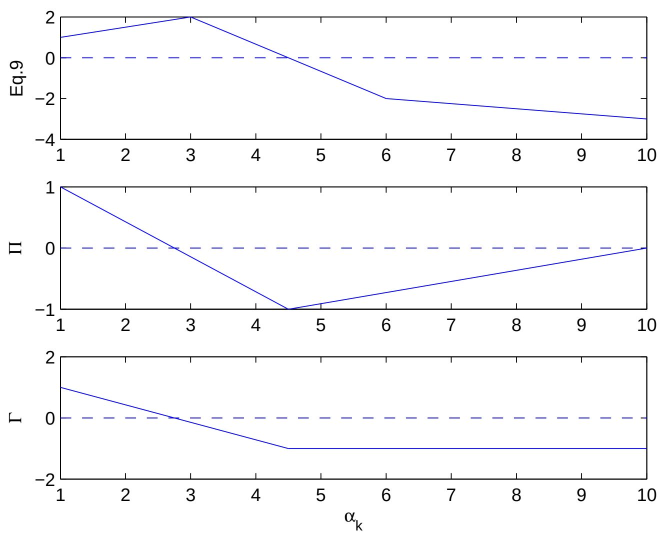

Figure 3.2: Construction of lower bound function with $\beta = 1$

The piecewise linear structure makes the proposed optimization scheme superior to the common coordinate descent search in two aspects. First, the solution of Problem 1, even without improvement, not only rejects the corresponding α_k sample, but also provides a cut that refines the sampling space. Second, the piecewise linear function can be described with a small number of intersection points, which are trivial to store and search. The upper envelope of many piecewise linear functions is also easy to be created.

Once a new value of α_k with enough improvement in the objective function is found, the optimizer switches to another component of α. The algorithm is terminated when there is no further decrease in the objective value by changing any of α_k. Assuming there are l parameters, the whole procedure is summarized below. The detailed implementation of each step is further discussed in the next section.

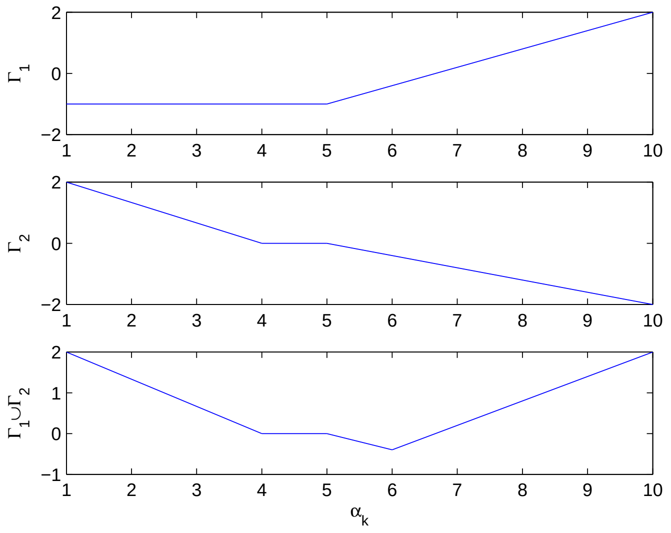

Figure 3.3: Construction of the upper envelope union of lower bound functions

Step 0: Set $k = 1$, $i = 1$, $h^0 = M$, where M is a large positive number.

Step 1: Initialize the lower bound function $\tilde{\Pi} = -\beta$ over the feasible section Λ_k and sample α_k.

Step 2: Given the sampled α_k, solve Problem 1 to obtain x, $\hat{x}$ and h^i, then compare h^i with the h^{i-1}. If $h^i < 1$, update α_k with the sampled value and return α. If $h^{i-1} - h^i > \gamma$, where γ is a positive threshold, update α_k with the sampled value, set $i \leftarrow i + 1$, $k \leftarrow k + 1$, and go to Step 1. Moreover, if $k > l$, then set $k = 1$.

Step 3: Based on $\{x, \hat{x}\}$, compute the piecewise linear function $\Gamma(\alpha_k)|_{\{x,\hat{x},h^{i-1}\}}$ according to Eq. (3.45) and update $\tilde{\Pi}(\alpha_k) = \max(\tilde{\Pi}(\alpha_k), \Gamma(\alpha_k)|_{\{x,\hat{x},h^{i-1}\}})$ in the feasible region Λ_k. Let $q = \min_{\alpha_k} \tilde{\Pi}(\alpha_k)$. If $q \geq \epsilon$, where ϵ is a negative value close to 0, let $k \leftarrow k + 1$, $h^i \leftarrow h^{i-1}$, $i \leftarrow i + 1$ and go to Step 1. In addition, if $k > l$, then set $k = 1$.

Step 4: Refine the set $\Lambda_k \longleftarrow \{\alpha_k | \quad \tilde{\Pi}(\alpha_k) < 0\}$.

Step 5: Select a new value of α_k in Λ_k and go to Step 2.

If the algorithm terminates without finding a solution, the initial guess should be changed or the target region be reduced.

We also note that Step 2 requires solving nonlinear programming and x and $\hat{x}$ can be searched separately. Since there exists no NLP solver that can guarantee the global optimum, grid checking is the most reliable approach. A similar idea is applied successfully in [55], which formulates linear constraints for each mesh point and solves a large-scale linear or quadratic optimization problem. Compared to this method, we only need to evaluate Eq. (3.14) for each grid point. Another similar work is proposed to generate a LMI problem for each point in the scheduling variable space of LPV system [52].

It is worthwhile to note that grid checking is not necessary if nonlinear programming (NLP) solver finds a solution ($\{x, \hat{x}\}$) that makes $\Pi > 0$ and the corresponding α can be ignored. Furthermore, only the points in the subset satisfying $V(x) \leq V(\hat{x})$ needs checking in this scheme. The number of checking points is far less than those of the previous methods in [55] and [52]. Discussions on the tradeoff between accuracy and efficiency for gridding can be found in [55].

3.6 Derivative-free Optimization Approach for RCLF

In this section, the iterative coordinate search is modified to design the RCLF. During the iteration, the solver tries to find a new P that increases the objective function $\Phi(P)$ compared with the value in the last round. Still, the subscripts and superscripts denote the sequence of parameters and iterations.

Assume that $\frac{\min_x x^T P x}{\max_{\tilde{x}} \tilde{x}^T P \tilde{x}} = h^{t-1}$ is the best solution achieved in iteration $t-1$, then in order to facilitate the optimization, the Dinkelbach's approach is employed to modify the fractional objective function of Problem 3:

$$\Pi(P, h^{t-1}) = \min_x x^T P x - h^{t-1} \max_{\tilde{x}} \tilde{x}^T P \tilde{x} \tag{3.46}$$

One can derive that

$$\frac{\min_x x^T P x}{\max_{\tilde{x}} \tilde{x}^T P \tilde{x}} > h^{t-1} \leftrightarrows \Pi(P, h^{t-1}) = \min_x x^T P x - h^{t-1} \max_{\tilde{x}} \tilde{x}^T P \tilde{x} > 0 \tag{3.47}$$

Therefore, the aim of iteration t is to seek a new P such that

$$\Pi(P, h^{t-1}) > 0 \tag{3.48}$$

Since the derivative information of $\Pi(P, h^{t-1})$ is not valid, the pattern search methodology is applied, which samples the P in the parameter space and solve the Problem 3 to see whether an improvement can be attained. Unfortunately, similar with CLF design, the global optimal of Problem 3 should be found by grid checking and the amount of computational time depends on the number of samples.

The detail of the algorithm is as follows. First of all, the constraint (3.28) can be solved by linear matrix inequality (LMI) to get the upper/lower bound of the kth parameter P_k of the matrix, before the tth iteration begins. The sampling space for this parameter is denoted by Λ_k. With the sample of P_k, a pair of $\{x, \tilde{x}\}$ is generated by solving Problem 3 and the following inequality holds for any P_k within Λ^k if the triple $\{x, \tilde{x}, P_k\}$ satisfies constraints (3.24) to (3.26):

$$x^T P x - h^{t-1} \tilde{x}^T P \tilde{x} \geqslant \Pi(P_k, h^{t-1}) = \min_{x} x^T P x - h^{t-1} \max_{\tilde{x}} \tilde{x}^T P \tilde{x} \tag{3.49}$$

The upper bound of the $\Pi(P_k, h^{t-1})$ over P_k, derived from the left side of (3.49), can restrict the sampling space because any P_k yielding a negative value in this bound function will not result in a positive Π, thereby not necessary to sample. Given $\{x, \tilde{x}\}$, this upper bound function can be defined in the entire Λ_k

$$\varphi(P_k)|_{\{x,\tilde{x}\}} = \begin{cases} x^T P x - h^{t-1} \tilde{x}^T P \tilde{x} & \text{if (3.24) holds and } \tilde{x}^T P \tilde{x} = \max_{\tilde{x}} \tilde{x}^T P \tilde{x} \\ \beta & \text{Otherwise} \end{cases} \tag{3.50}$$

where β is a large enough positive number which does not affect the estimation of the upper bound in that area.

To distinguish different cases in (3.50), one can easily identify the region of P_k which renders (3.24) hold because it is piecewise linear over this single parameter proved by Theorem 10. Moreover, for the low dimensional system and simple polyhedron $\tilde{\Theta}$, the number of vertex is small enough, say Y. Thus, it is not difficult to pre-divide Λ_k into Y parts, $\{\tau_1, \tau_2, \ldots, \tau_Y\}$. In the part j, $\forall P_k \in \tau_j$, $\tilde{x}^T P \tilde{x}$ achieves its maximum value in the jth vertex.

Finally, we can also conclude that the φ is piecewise linear over its parameter P_k. Hence, as more samples of P_k are collected, the yielded different piecewise function φ, can be combined together to approximate the true objective value more accurately. This combined upper bound function $\tilde{\Pi}$, can be achieved by calculating the lower envelope of all φ. Here we have to point out that the lower envelope can be obtained efficiently only when the φ is the function of single variable. To derive higher dimensional envelope for multi-variable, one should refer to the computational geometry, which is too complex for this scheme.

Although there are some differences between the RCLF and CLF design, the construction of the bound function make them share the same advantage. Namely, solving the inner problem not only rejects the corresponding P_k sample point, but also refines the sample space Λ_k, thereby enhancing the efficiency of the search. The whole algorithm is summarized as below:

Step 0: Set $t = 1$, $k = 1$, $h^0 = M$, γ, where M and γ are small enough positive number. Specify the upper limit of the iteration times T. Determine the total parameter number N.

Step 1: If $k > N$, set $k = 1$; If $t > T$, terminate the algorithm; Otherwise, solve the LMI for the kth parameter of RCLF, say P_k, to get Λ_k. Initialize the upper bound function $\tilde{\Pi} = \beta$ over the Λ_k.

Step 2: Given the sampled value of P_k, solve Problem 3 to obtain the x and $\tilde{x}$. Then if $\Phi(P) - h^{t-1} > \gamma$, update P_k, h_t with the sampled value and $\Phi(P)$, set $t \leftarrow t+1$,$k \leftarrow k+1$ and go to Step 1.

Step 3: Based on the current $\{x, \tilde{x}\}$, compute the piecewise linear function $\varphi(P_k)$ within Λ_k. Then, update upper bound function, $\tilde{\Pi}(P_k) = \min(\tilde{\Pi}(P_k), \varphi(P_k))$ and compute $q = \max_{P_k} \tilde{\Pi}(P_k)$. If $q \leq \epsilon$, where ϵ is a positive value close to 0, let $h^t \leftarrow h^{t-1}$, $t \leftarrow t+1$, $k \leftarrow k+1$ and go to Step 1.

Step 4: Refine the feasible region: $\Lambda_k \leftarrow \{P_k | P_k \in \Lambda_k \quad \text{and} \quad \tilde{\Pi}(P_k) > 0\}$

Step 5: Sample a new value of P_k in Λ_k and go to Step 2.

This algorithm terminates once the total sampling times exceeds the limit T. In order to enlarge the ROA and reduce the residual set enough, T needs to be large.

3.7 Efficient Implementation of the CLF Design Algorithm

In this section, we illustrate the implementation detail of the CLF design algorithm, including how to locate the region of Λ_k and handle the state constraints. Moreover, considering that grid-checking to solve Problem 1 is time consuming, an accelerate step is proposed by appealing to the nonlinear programming.

3.7.1 Feasible Region for α_k

The feasible region Λ_k is determined in terms of the lower and upper bounds of α_k to respect the following positive definiteness for $\forall\ x \in \Xi$:

$$V(x) = \sum_{t=1}^{l} \alpha_t \phi_t(x) = \alpha_k \phi_k(x) + \sum_{t=1,\neq k}^{l} \alpha_t \phi_t(x) > 0 \tag{3.51}$$

The bounds can be derived as the function of x from Eq. (3.51), and the following optimization problems can be solved:

Problem 5

$$\begin{aligned} \underline{\alpha}_k &= \max_x \frac{-\sum_{t=1,\neq k}^{l} \alpha_t \phi_t(x)}{\phi_k(x)} \\ \text{s.t.} \quad & \phi_k(x) \geq \epsilon \\ & x \in \Xi \end{aligned}$$

Problem 6

$$\begin{aligned} \overline{\alpha}_k &= \min_x \frac{-\sum_{t=1,\neq k}^{l} \alpha_t \phi_t(x)}{\phi_k(x)} \\ \text{s.t.} \quad & \phi_k(x) \leq -\epsilon \\ & x \in \Xi \end{aligned}$$

where ϵ is a small positive number. The constraints keep x from the equilibrium $(x_0 = \{0\})$ to avoid singularity. Due to these constraints, we should further guarantee the positive definiteness around the equilibrium. In other words, α_k should keep $V(x_0) = 0$ as the local minimum. Since we require $\left.\frac{\partial \phi_t(x)}{\partial x}\right|_{x_0} = 0$, there also is

$$\sum_{t=1}^{l} \alpha_t \left.\frac{\partial \phi_t(x)}{\partial x}\right|_{x_0} = 0 \tag{3.52}$$

It is only necessary to have the positive definite Hessian matrix of V:

$$\sum_{t=1}^{l} \alpha_t \begin{pmatrix} \frac{\partial^2 \phi_t}{\partial x_1^2} & \frac{\partial^2 \phi_t}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 \phi_t}{\partial x_1 \partial x_n} \\ \frac{\partial^2 \phi_t}{\partial x_2 \partial x_1} & \frac{\partial^2 \phi_t}{\partial x_2^2} & \cdots & \frac{\partial^2 \phi_t}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 \phi_t}{\partial x_n \partial x_1} & \frac{\partial^2 \phi_t}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 \phi_t}{\partial x_n^2} \end{pmatrix} = \sum_{t=1}^{l} \alpha_t D_t > 0 \tag{3.53}$$

This inequality is easy to solve with the LMI toolbox in MATLAB. In summary, Λ_k can be estimated from Problems 5 and 6 and Eq.(3.53). If a CLF is not positive definite for at least one point of $x \in \Xi$, new bounds of α_k are obtained by solving Problems 5 and 6 with that state point as an initial guess.

3.7.2 Acceleration of Step 2

The most time consuming step in the suggested algorithm is Step 2 where Problem 1 is solved to obtain $\{x, \hat{x}\}$ and check if the sampled α_k yields a larger ROA. Hence, if the sampling range for α_k is reduced, the computational requirement can be further decreased. This section shows that $\{x, \hat{x}\}$ can be smartly selected to identify the range of α_k that does not decrease the objective function. In this range, α_k needs not sampling further.

Eq. (3.14) characterizes no-decay state points for the given α_k, and can be further written as:

$$
\begin{aligned}
& \inf_{u \in U} \frac{\partial V}{\partial x} f(x) + \frac{\partial V}{\partial x} g(x) u && (3.54)\\
= \; & \inf_{u \in U} \alpha^T \frac{\partial \phi(x)}{\partial x} (f(x) + g(x)u) && (3.55)\\
= \; & \alpha^T \frac{\partial \phi(x)}{\partial x} f(x) + \sum_{s=1}^{m} \inf_{u_s} \sum_{j=1}^{n} \alpha^T \frac{\partial \phi(x)}{\partial x_j} g_{js}(x) u_s && (3.56)\\
= \; & \alpha^T \frac{\partial \phi(x)}{\partial x} f(x) - \sum_{s=1}^{m} \frac{(\overline{u}_s - \underline{u}_s)}{2} \left| \sum_{j=1}^{n} \alpha^T \frac{\partial \phi(x)}{\partial x_j} g_{js}(x) \right| \\
+ \; & \sum_{s=1}^{m} \frac{(\overline{u}_s + \underline{u}_s)}{2} \sum_{j=1}^{n} \alpha^T \frac{\partial \phi(x)}{\partial x_j} g_{js}(x) \geqslant 0 && (3.57)
\end{aligned}
$$

where n is the dimension of state; m is the dimension of input; g_{js} is the jth row and sth column item of the matrix $g(x)$. The Eq. (3.57) holds because input is determined by the sign of $\sum_{j=1}^{n} \alpha^T \frac{\partial \phi(x)}{\partial x_j} g_{js}(x)$ in the corresponding channel and only the lower or upper bound signal are used.

The α_k associated with the given state point can be written as a function of x from the equality part of Eq. (3.54):

$$
\begin{aligned}
& \alpha_k(x) \\
= & \left\{ \sum_{t=1,\neq k}^{l} \alpha_t \left[\frac{\partial \phi_t(x)}{\partial x_t}, \ldots, \frac{\partial \phi_t(x)}{\partial x_n} \right] f(x) + \sum_{s=1}^{m} u_s \left(\sum_{t=1,\neq k}^{l} \sum_{j=1}^{n} \alpha_t \frac{\partial \phi_t(x)}{\partial x_j} g_{js}(x) \right) \right\} \\
\bullet & \left\{ \left[\frac{\partial \phi_k(x)}{\partial x_1}, \ldots, \frac{\partial \phi_k(x)}{\partial x_n} \right] f(x) + \sum_{s=1}^{m} u_s \left(\sum_{j=1}^{n} \frac{\partial \phi_k(x)}{\partial x_j} g_{js}(x) \right) \right\}^{-1}
\end{aligned}
$$

and

$$u_s(x) = \begin{cases} \overline{u}_s & \text{if } s \in \psi_1 \\ \underline{u}_s & \text{if } s \in \psi_2 \end{cases}$$

$\psi_1 : \{s| \sum_{t=1}^{l} \sum_{j=1}^{n} \alpha_t \frac{\partial \phi_t(x)}{\partial x_j} g_{js}(x)) < 0\}$
$\psi_2 : \{s| \sum_{t=1}^{l} \sum_{j=1}^{n} \alpha_t \frac{\partial \phi_t(x)}{\partial x_j} g_{js}(x)) \geqslant 0\}$
ψ_1 and ψ_2 can be specified by the user. Then, the following nonlinear feasibility problem is solved to obtain the pair $\{x, \hat{x}\}$. The derived lower bound function $\Gamma(\alpha_k)$ can characterize a segment of α_k that needs not sampling further.

Problem 7

$$\alpha_k(x)\phi_k(\hat{x}) + \sum_{t=1,\neq k}^{l} \alpha_t\phi_t(\hat{x}) - h(\alpha_k(x)\phi_k(x) + \sum_{t=1,\neq k}^{l} \alpha_t\phi_t(x)) \geqslant 0 \tag{3.58}$$

$$\sum_{t=1}^{l}\sum_{j=1}^{n} \alpha_t \frac{\partial \phi_t(x)}{\partial x_j} g_{js}(x) \leqslant 0 \quad s \in \psi_1 \tag{3.59}$$

$$\sum_{t=1}^{l}\sum_{j=1}^{n} \alpha_t \frac{\partial \phi_t(x)}{\partial x_j} g_{js}(x) \geqslant 0 \quad s \in \psi_2 \tag{3.60}$$

$$\alpha_k(x) \in \Lambda_k \tag{3.61}$$

$$\hat{x} \in \Psi \tag{3.62}$$

$$x \in \Xi \tag{3.63}$$

The inequality (3.58) is from the function $\Pi(x, \hat{x}, \alpha_k(x))|_h$ in (3.42), which determines $\alpha_k(x)$ that does not decrease the function $\Phi(\alpha)$ in (3.13). Without a special note, the index of h is omitted for simplicity. Due to the continuity of (3.54) over α_k, we can identify the positive range of the resulting piecewise linear function $\Pi(\alpha_k)|_{\{x,\hat{x},h\}}$ and refine Λ_k further. Eq. (3.61) guarantees that the derived α_k is in the sampling region Λ_k. Although Problem 7 is nonlinear, searching for a feasible solution without global optimal requirement is much faster than the grid checking in Problem 1.

3.7.3 State Constraints

Whereas most existing CLF construction methods do not consider state constraints explicitly, the proposed approach can be extended to handle the following type of state constraints, which are common in chemical processes to represent physically meaningful quantities or safety limits:

$$\underline{x}_i \leqslant x_i \leqslant \overline{x}_i \qquad i \in \{1, 2, \ldots, n\} \tag{3.64}$$

where i represents the component of state vector. The constraints may change the boundary of ROA. Namely, r_{max} is not only determined by (3.14), but also state constraints. In that case, the following feasibility problem should be considered, given a sampled α_k:

Problem 8

$$\sum_{t=1}^{l} \alpha_t \phi_t(\hat{x}) - h \sum_{t=1}^{l} \alpha_t \phi_t(x) \geqslant 0 \tag{3.65}$$

$$x_i = \underline{x}_i \quad i \in \underline{G} \subset \{1, 2, 3, \dots, n\} \tag{3.66}$$

$$x_j = \overline{x}_j \quad j \in \overline{G} \subset \{1, 2, 3, \dots, n\} \tag{3.67}$$

$$\hat{x} \in \Psi \tag{3.68}$$

$$x \in \Xi \tag{3.69}$$

where G is the active constraint set, selected by the user.

In fact, the Problem 8 can be employed to replace Problem 1 in Step2 when state constraints affect ROA. The (3.65) characterizes α_k that cannot improve the ROA due to constraints for $x_i, i \in \underline{G}$ and $x_j, j \in \overline{G}$. The generated $\{x, \hat{x}\}$ can also be used to construct the piecewise linear function $\Pi(\alpha_k)|_{\{x,\hat{x},h\}}$ and refine Λ_k. Because of the simplicity of inequality (3.65) and the fixed value of several variables, Problem 8 is much easier to be solved than Problem 1.

3.8 Guarantee of the Small Control Property

This section provides a sufficient condition for the CLF to have the small control property so that there exists the continuous control at the origin.

Taylor series expansion of $\frac{\partial V}{\partial x}$ around the equilibrium $\{0\}$ becomes

$$\frac{\partial V}{\partial x} = [x_1, x_2, \dots, x_n] P_{v1} + x_{\{2\}}^T P_{v2} \tag{3.70}$$

where $x_{\{2\}} = [x_1^2, \dots, x_1^{t_1} x_2^{t_2} \dots x_n^{t_n}, \dots, x_n^2]^T$; P_{v1} is the Hessian matrix of V,

$$P_{v1} = \begin{pmatrix} \frac{\partial^2 V}{\partial x_1^2} & \cdots & \frac{\partial^2 V}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 V}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 V}{\partial x_n^2} \end{pmatrix} \tag{3.71}$$

$x_{\{2\}}^T P_{v2}$ is the Lagrange form of the remainder. Thus, there is a point $\xi \in (0, x)$

such that

$$P_{v2} = \begin{pmatrix} \frac{\partial^3 V(\xi)}{2!\partial^3 x_1} & \cdots & \frac{\partial^3 V(\xi)}{2!\partial^2 x_1 \partial x_n} \\ \frac{\partial^3 V(\xi)}{t_1!...t_n!\partial^{t_1+1} x_1 ... \partial^{t_n} x_n} & \cdots & \frac{\partial^3 V(\xi)}{t_1!...t_n!\partial^{t_1} x_1 ... \partial^{t_n+1} x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^3 V(\xi)}{2!\partial x_1 \partial^2 x_n} & \cdots & \frac{\partial^3 V(\xi)}{2!\partial^3 x_n} \end{pmatrix}$$
$$t_1 + t_2 + \ldots + t_n = 2,\ t_i \in \{0, 1, 2\}$$

Similarly, the nonlinear system can also be expanded as

$$\dot{x} = f(x) + g(x)u = [F_1(x,u), F_2(x,u), \ldots, F_n(x,u)]^T = Ax + Bu + C\psi + A_{\{2\}}x_{\{2\}} \tag{3.72}$$

where $A = \frac{\partial F(x,u)}{\partial x}|_{x_0=0,u_0=0}$, $B = \frac{\partial F(x,u)}{\partial u}|_{x_0=0,u_0=0}$; $C\psi + A_{\{2\}}x_{\{2\}}$ is the Lagrange form of the remainder:

$$A_{\{2\}} = \begin{pmatrix} \frac{\partial^2 F_1(\xi,\zeta)}{2!\partial^2 x_1} & \cdots & \frac{\partial^2 F_1(\xi,\zeta)}{t_1!t_2!...t_n!\partial^{t_1} x_1 ... \partial^{t_n} x_n} & \cdots & \frac{\partial^2 F_1(\xi,\zeta)}{2!\partial^2 x_n} \\ \frac{\partial^2 F_2(\xi,\zeta)}{2!\partial^2 x_1} & \cdots & \frac{\partial^2 F_2(\xi,\zeta)}{t_1!t_2!...t_n!\partial^{t_1} x_1 ... \partial^{t_n} x_n} & \cdots & \frac{\partial^2 F_2(\xi,\zeta)}{2!\partial^2 x_n} \\ \vdots & \ddots, & \vdots, & \ddots & \vdots \\ \frac{\partial^2 F_n(\xi,\zeta)}{2!\partial^2 x_1} & \cdots & \frac{\partial^2 F_n(\xi,\zeta)}{t_1!t_2!...t_n!\partial^{t_1} x_1 ... \partial^{t_n} x_n} & \cdots & \frac{\partial^2 F_n(\xi,\zeta)}{2!\partial^2 x_n} \end{pmatrix} \tag{3.73}$$

$$C_{i,(n-1)j+k} = \frac{\partial^2 F_i(\xi,\zeta)}{\partial x_j u_k} \tag{3.74}$$

$$\psi = [x_1u_1, x_1u_2, \ldots, x_1u_m, x_2u_1, \ldots, x_nu_m]^T \tag{3.75}$$

where $\{\xi, \zeta\}$ denotes a point between $\{0, 0\}$ and $\{x, u\}$.

Then the following theorem can be established:

Theorem 11. *Suppose* $|\frac{\partial^2 F_i(x,u)}{\partial x^2}| \leqslant M_2$, $|\frac{\partial^3 V(x)}{\partial x^3}| \leqslant M_3$ *and* $|\frac{\partial^2 F_i(x,u)}{\partial x \partial u}| \leqslant M_4$ *hold for the system Eq. (3.1) with input constraint and the CLF of Eq. (3.11). If* P_{v1} *is positive definite and there exists a positive scalar* ν *rendering* Q *positive definite:*

$$Q = -(A^T P_{v1} + P_{v1}A - \nu P_{v1}BB^T P_{v1}) > 0 \tag{3.76}$$

Then V *has the small control property.*

Proof. For simplicity of notation, we assume that the system has an equilibrium point x_0 at the origin, and $x = [x_1, x_2, x_3, \ldots, x_n]^T$ is the state vector. Let $D := \{x | \, ||x||_\infty \leqslant \eta\}$; $F(x,u)$ and $V(x)$ are defined on the set D.

Consider the stabilizable criterion and the control law:

$$u = \frac{-B^T P_{v1} x}{2} \tag{3.77}$$

There we have

$$\begin{aligned}&\frac{\partial V}{\partial x}(f(x)+g(x)u) \qquad (3.78)\\ &= (x^T P_{v1} + x_{\{2\}}^T P_{v2})(Ax+Bu+C\psi+A_{\{2\}}x_{\{2\}})\\ &= x^T P_{v1}(Ax+Bu) + x^T P_{v1}(C\psi + A_{\{2\}}x_{\{2\}}) + x_{\{2\}}^T P_{v2}(Ax+Bu+C\psi+A_{\{2\}}x_{\{2\}})\\ &= -x^T\frac{Q}{2}x + x^T P_{v1}(C\psi + A_{\{2\}}x_{\{2\}}) + x_{\{2\}}^T P_{v2}(Ax+Bu+C\psi+A_{\{2\}}x_{\{2\}})\end{aligned}$$

Give the Q positive definite, there is a positive scalar K satisfying $x^T\frac{Q}{2}x > K||x||_2^2 \geqslant K||x||_\infty^2$, then we have:

$$\begin{aligned}&\frac{\partial V}{\partial x}(f(x)+g(x)u) \qquad (3.79)\\ <\ & -K||x||_\infty^2 + x^T P_{v1}(C\psi + A_{\{2\}}x_{\{2\}}) + x_{\{2\}}^T P_{v2}(Ax+Bu+C\psi+A_{\{2\}}x_{\{2\}})\\ <\ & -K||x||_\infty^2 + ||A_{\{2\}}||_\infty||P_{v1}||_\infty||x||_\infty^3 + \frac{1}{2}||x||_\infty^3||B||_\infty||C||_\infty||P_{v1}||_\infty^2\\ & +||x||_\infty^3\left(||A||_\infty||P_{v2}||_\infty + \frac{1}{2}||P_{v2}||_\infty||P_{v1}||_\infty||BB^T||_\infty\right)\\ & +\frac{1}{2}||x||_\infty^4||B||_\infty||C||_\infty||P_{v1}||_\infty||P_{v2}||_\infty + ||x||_\infty^4||A_{\{2\}}||_\infty||P_{v2}||_\infty\\ <\ & -K||x||_\infty^2 + \left(\frac{n^2+n}{2}M_2||P_{v1}||_\infty + \frac{mnM_4}{2}||B||_\infty||P_{v1}||_\infty^2\right)||x||_\infty^3\\ & +||x||_\infty^3\left(n||A||_\infty M_3 + \frac{1}{2}nM_3||P_{v1}||_\infty||BB^T||_\infty\right)\\ & +\left(\frac{n^3+n^2}{2}M_2M_3 + \frac{n^2m||B||_\infty||P_{v1}||_\infty M_3M_4}{2}\right)||x||_\infty^4\end{aligned}$$

The second strict inequality holds because

$$||x_{\{2\}}||_\infty = ||x||_\infty^2 \qquad (3.80)$$

$$||\psi||_\infty = ||x||_\infty||u||_\infty \leqslant 0.5||x||_\infty^2||B||_\infty||P_{v1}||_\infty \qquad (3.81)$$

The third strict inequality holds because

$$\begin{aligned}||A_{(2)}||_\infty &= \max_{1\leqslant i\leqslant n}\sum_{j=1}^{\frac{n^2+n}{2}}|A_{\{2\}ij}| \leqslant \frac{n^2+n}{2}\times\max\left|\frac{\partial^2 F_i(\xi,\zeta)}{\partial^{t_1}x_1\ldots\partial^{t_n}x_n}\right|\\ &= \frac{n^2+n}{2}M_2 \qquad (3.82)\end{aligned}$$

$$\begin{aligned}||P_{v2}||_\infty &= \max_{1\leqslant i\leqslant \frac{n^2+n}{2}}\sum_{j=1}^{n}|P_{v2ij}| \leqslant n\times\max\left|\frac{\partial^3 V(\xi)}{\partial^{t_1}x_1\ldots\partial^{t_n}x_n\partial x_i}\right|\\ &= nM_3 \qquad (3.83)\end{aligned}$$

and

$$||C||_\infty = \max_{1\leqslant i\leqslant n} \sum_{j=1}^{mn} |C_{ij}| \leqslant mnM_4 \tag{3.84}$$

By further analyzing the third inequality of (3.79), we can see that if

$$||x||_\infty < \frac{-W_2 + \sqrt{W_2^2 - 4W_1W_3}}{2W_1} \tag{3.85}$$

where

$$\begin{aligned}
W_1 &= \frac{n^3+n^2}{2}M_2M_3 + \frac{n^2m||B||_\infty||P_{v1}||_\infty M_3M_4}{2}\\
W_2 &= \frac{n^2+n}{2}M_2||P_{v1}||_\infty + \frac{mnM_4}{2}||B||_\infty||P_{v1}||_\infty^2 + n||A||_\infty M_3\\
&\quad + \frac{1}{2}nM_3||P_{v1}||_\infty||BB^T||_\infty\\
W_3 &= -K
\end{aligned}$$

The right hand of (3.79) is negative.

Now given any bound of input μ, from the control law (3.77), there is

$$||\frac{-B^TP_{v1}x}{2}||\infty \leqslant \mu \Rightarrow ||x||_\infty \leqslant \frac{2\mu}{||B^TP_{v1}||_\infty} \tag{3.86}$$

Finally, we can define the domain

$$||x||_\infty < \min\{\frac{-W_2 + \sqrt{W_2^2 - 4W_1W_3}}{2W_1}, \frac{2\mu}{||B^TP_{v1}||_\infty}, \eta\} \tag{3.87}$$

such that the output of control law in (3.77) is within the bound and the stabilizable criterion is negative. □

3.9 State Feedback Controller Design Based on CLF

A continuous CLF-based feedback control law for the affine system with 2-norm bounded input is proposed in [76]. This work modifies the approach to handle the infinity-norm bounded input. Without loss of generality, the input constraint is assumed to be $||u||_\infty \leq 1$, and let $\varrho \subseteq \Re^2$ denote the open set:

$$\varrho = \{(\hat{a}, \hat{b})|\ \hat{a} < |\hat{b}|\} \tag{3.88}$$

where $\hat{a}$ and $\hat{b}$ are two scalars.

The following Definition and Lemma are introduced from [76] to prove the control law.

Definition 12. *K-continuous [76]: A function $Z(\hat{a}, \hat{b})$ is said to be K-continuous if there is a scalar $\delta > 0$ for all $\epsilon > 0$ such that*

$$[|\hat{b}| < \delta, \hat{a} < \delta|\hat{b}|] \Rightarrow |Z(\hat{a}, \hat{b})| < \epsilon \tag{3.89}$$

Lemma 13. *A scalar function:*

$$Z(\hat{a}, \hat{b}) = \begin{cases} -\frac{\hat{a}+\sqrt{\hat{a}^2+\hat{b}^4}}{\hat{b}(1+\sqrt{1+\hat{b}^2})} & \text{if } \hat{b} \neq 0 \\ 0 & \text{if } \hat{b} = 0, \end{cases}$$

which is defined in ϱ, has the following properties [76]:

1. $Z(\hat{a}, \hat{b})$ *is* K-continuous
2. $|Z(\hat{a}, \hat{b})| < 1$ *for all* $(\hat{a}, \hat{b}) \in \varrho$
3. $\hat{a} + \hat{b}Z(\hat{a}, \hat{b}) < 0$ *for all* $(\hat{a}, \hat{b}) \in \varrho$

Now define the $a(x)$ and $b(x)$ as:

$$a(x) = \frac{\partial V}{\partial x} f(x) \tag{3.90}$$

$$b(x) = [b_1(x), b_2(x), \ldots, b_m(x)] = \frac{\partial V}{\partial x} g(x) \tag{3.91}$$

In the following, the controller and its property are discussed:

Theorem 14. *Given the control Lyapunov function V satisfying small control property and the feedback law:*

$$u_i(x) = \begin{cases} K_i(x)b_i(x) & \text{if } |b_i(x)| \neq 0 \\ 0 & \text{if } |b_i(x)| = 0 \end{cases}$$

It is assumed that $|u_i| \leqslant 1, \ \forall \ i \ \in \ \{1, 2, \ldots, m\}$. Let $R_i(x) = \frac{|b_i(x)|a(x)}{\sum_{j=1}^m |b_j(x)|}$ and $K_i(x) = -\frac{R_i(x)+\sqrt{R_i(x)^2+(|b_i(x)|)^4}}{|b_i(x)|^2[1+\sqrt{1+(|b_i(x)|)^2}]}$, then the system (3.1) is stable by the input within the constraint, and the u is continuous at the origin.

Proof. Given the proper CLF, there are:

$$\begin{aligned} &\inf_{u \in U} \frac{\partial V}{\partial x} f(x) + \frac{\partial V}{\partial x} g(x)u < 0 \quad \forall x \in \Sigma \neq 0 \\ &\longrightarrow a(x) - \sum_{j=1}^{m} |b_j(x)| < 0 \quad \forall x \in \Sigma \neq 0 \\ &\longrightarrow \frac{|b_i(x)|a(x)}{\sum_{j=1}^m |b_j(x)|} - |b_i(x)| < 0 \quad \forall x \in \Sigma \neq 0, i \in \{1, 2, \ldots, m\} \\ &\longrightarrow R_i(x) < |b_i(x)| \quad \forall x \in \Sigma \neq 0, i \in \{1, 2, \ldots, m\} \end{aligned} \tag{3.92}$$

where m is the number of inputs and Σ is the decay region of the given CLF, see (3.3). Thus, we know that the pair (R_i, b_i) is in the set ϱ defined by Eq. (3.88) and u_i can be simplified as:

$$u_i(x) = \begin{cases} -\frac{R_i+\sqrt{R_i^2+b_i^4}}{b_i(1+\sqrt{1+b_i^2})} & \text{if } |b_i(x)| \neq 0 \\ 0 & \text{if } |b_i(x)| = 0, \end{cases}$$

which is similar with Z defined in Lemma 13. According to Lemma 13, there is $R_i + b_i u_i(R_i, b_i) < 0$ for each input component, so $a + \sum_{i=1}^m b_i u_i < 0$ which guarantees the Lyapunov function is decreasing while following the state trajectory, i.e. stability; By the same Lemma, there is also $|u_i| < 1$ satisfying the input constraint. It only remains to prove the continuity of u. Similar with the proof in [76], we want to show that for each $\epsilon > 0$, there is an ϵ' such that if $0 < ||x||_\infty < \epsilon'$ then $||u||_\infty < \epsilon$.

From the *K-continuous* property of u_i, for the ϵ and each input component, there is a δ_i such that $[|b_i| < \delta_i, R_i < \delta_i|b_i|] \Rightarrow |u_i| < \epsilon$. For any $\delta_i > 0$, we can select the bound ϵ' such that $||x||_\infty < \epsilon'$ leading $|b_i(x)| < \delta_i$ by the continuity of $b_i(x)$. Then we define $\delta = \min\{\delta_i\}$. According to the small control property, the ϵ' is also chosen such that any state in $||x||_\infty < \epsilon'$ can be stabilized by one of the inputs within $||u||_\infty < \delta$, i.e.:

$$a + bu < 0 \Rightarrow a < \delta \sum_{i=1}^{m} |b_i| \Rightarrow R_i < \delta |b_i| \Rightarrow R_i < \delta_i |b_i|$$

This completes the proof. $\square$

Although the Theorem 14 provides a feedback control law that can stabilize the system even with the input constraints, the modern control system usually employs the discrete regulation fashion, which is not applicable to this continuous policy. Fortunately, if the sampling time or holding time is small enough, the CLF/RCLF based discrete controller can guarantee the system converge to an invariant set containing equilibrium. In next chapter, we will follow the idea in [83, 84] to prove this proposition for RCLF controller.

3.10 Case Study

3.10.1 Case1: Disturbance-free CSTR

Consider the continuous stirred tank reactor example in [84], whose model takes the form of:

$$\dot{C}_A = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{-E/RT_R} C_A \tag{3.93}$$

$$\dot{T}_R = \frac{F}{V}(T_{A0} - T_R) + \frac{-\Delta H}{\rho C_p} k_0 e^{-E/RT_R} C_A + \frac{Q_\sigma}{\rho C_p V} \tag{3.94}$$

where $0 < C_{A0} < 2kmol/m^3$ and $|Q_\sigma| < 0.0167kJ/min$ are two inputs. The other parameters are shown in Table 3.1. In this case we supposed that all states are observed.

Table 3.1: Parameters of the process model

$V = 0.1m^3$	$R = 8.314kJ/kmolK$
$C_{A0s} = 1.0kmol/m^3$	$T_{A0s} = 310.0K$
$\Delta H = -4.78 \times 10^4 kJ/kmol$	$k_0 = 72 * 10^9 min^{-1}$
$E = 8.314 \times 10^4 kJ/kmol$	$C_p = 0.239kJ/kgK$
$\rho = 1000kg/m^3$	$F = 0.1m^3/min$
$T_{Rs} = 395.33K$	$C_{As} = 0.57kmol/m^3$

The control objective is to stabilize the target region of $0.6kmol/m^3 < C_A < 0.62kmol/m^3$, $397.5K < T_R < 398K$ and drive the system to the unstable steady state: $T_{Rs} = 395.33\ K$, $C_{As} = 0.57\ kmol/m^3$. Note that the model is not polynomial, the SOS cannot be applied directly. Here the quadratic CLF is employed to stabilize the system. Solving the Ricatti equation for the linearized system to get the initial guess:

$$P_{initial} = \begin{bmatrix} 14.6402 & 0.8897 \\ 0.8897 & 0.1041 \end{bmatrix}$$

The ROA with the initial CLF is shown in Fig. 3.4. However, it cannot incorporate the whole target region. A new CLF: $V = x^T P x$, where $x = \{Ca - 0.57, T_R - 395.33\}$, can be calculated by the proposed method. In the nonlinear search step, for solving Problem1, 15 initial points were selected randomly to help estimating the global optimal. Moreover, after each optimization step, we also use grids to check its optimality.

$$P_{final} = \begin{bmatrix} 23.433 & 1.0185 \\ 1.0185 & 0.0532 \end{bmatrix}$$

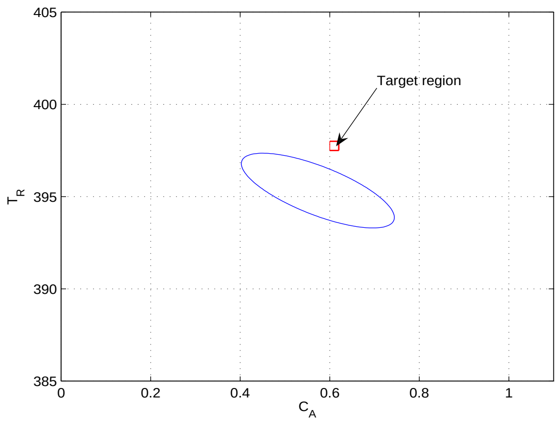

Figure 3.4: Region of attraction for matrix $P_{initial}$

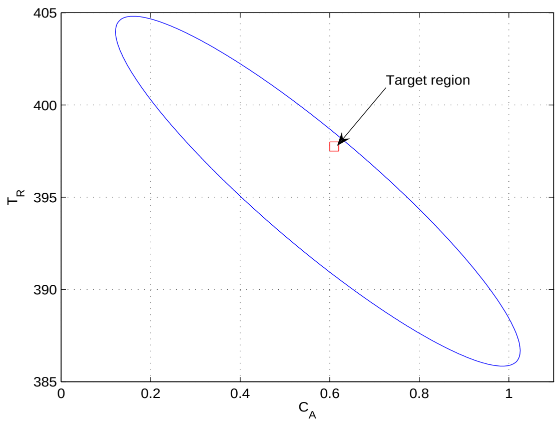

Figure 3.5: Region of attraction for matrix P_{final}

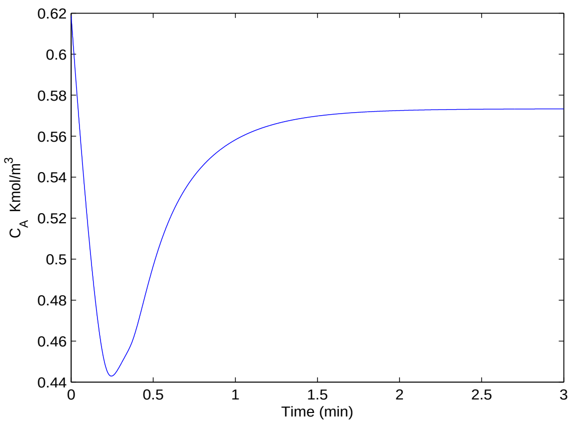

Figure 3.6: The evolution of state C_A in response to input trajectories in Fig.3.8 and 3.9

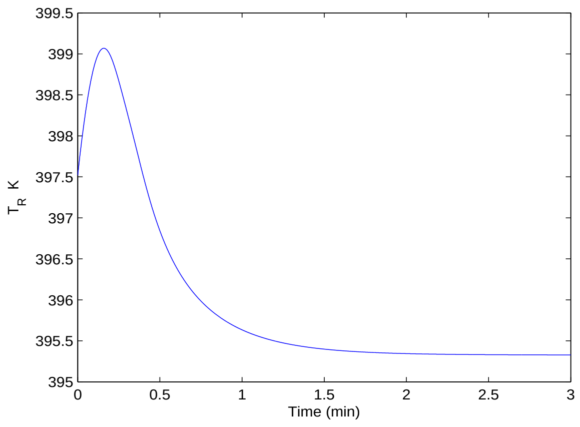

Figure 3.7: The evolution of state T_R in response to input trajectories in Fig.3.8 and 3.9

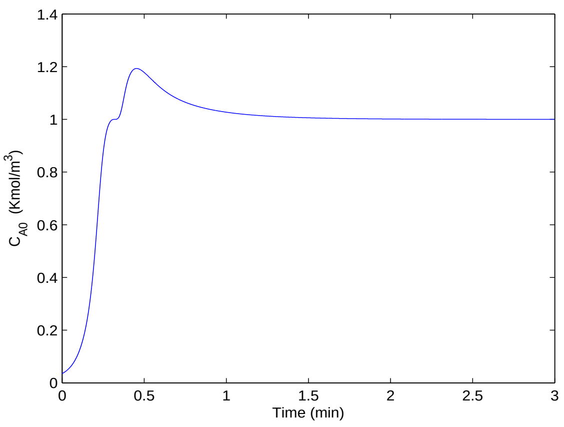

Figure 3.8: The input C_{A0}

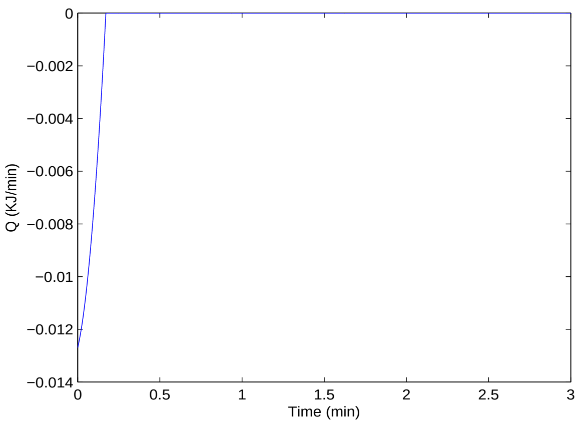

Figure 3.9: The input Q

The ROA and target area for the initial guess $P_{initial}$ is shown in Fig. 3.4. For P_{final}, its ROA and specified region is shown in Fig. 3.5. Figs. 3.6 and 3.7 show that the system reach the steady state smoothly and Figs. 3.8 and 3.9 show the input trajectories are within the constraints.

Moreover, it is not difficult to verify that $P_{v1} = P_{final}$ for the quadratic CLF. Assume $\nu = 1$, there is

$$Q = -(A^T P_{v1} + P_{v1}A - P_{v1}BB^t P_{v1}) = \begin{bmatrix} 328.1626 & 13.8728 \\ 13.8728 & 0.6224 \end{bmatrix}$$

which is a positive finite matrix. Thus, the derived CLF has the small control property. This conclusion is easy to be checked by the Fig. 3.6 $\sim$ 3.9.

3.10.2 Case2: CSTR with Bounded Uncertainties

In this case, we still employ the CSTR model. However, the real process can be affected by many disturbances. Here two sources of uncertainties are mainly considered: (1) The concentration fluctuation of the inlet flow ϵC_{A0}. (2) The disturbance μ affecting $\dot{T}_R$. In summary, the true model can be express as:

$$\dot{C}_A = \frac{F}{V}(C_{A0}(1+\epsilon) - C_A) - k_0 e^{-E/RT_R} C_A \tag{3.95}$$

$$\dot{T}_R = \frac{F}{V}(T_{A0} - T_R) + \frac{-\Delta H}{\rho C_p} k_0 e^{-E/RT_R} C_A + \frac{Q_\sigma}{\rho C_p V} + \mu \tag{3.96}$$

The input is $0 < C_{A0} < 2kmol/m^3$ and $|Q_\sigma| < 90kJ/min$. Here we also assumed that all states can be observed.

In order to derive the RCLF, the CSTR model can be rewrote as the typical control affine system and the terms are:

$$f = [-\frac{F}{V}C_A - k_0 e^{-E/RT_R} C_A, \frac{F}{V}(T_{A0} - T_R) - \frac{\Delta H}{\rho c_p} k_0 e^{-E/RT_R} C_A]^T \tag{3.97}$$

$$\Delta f = [0, \mu]^T \tag{3.98}$$

$$g = [\frac{F}{V}, 0; 0, \frac{1}{\rho c_p V}] \tag{3.99}$$

$$\Delta g = [\frac{\epsilon F}{V}, 0; 0, 0] \tag{3.100}$$

Here we assume that disturbances are $\epsilon C_{A0} \in [-0.2C_{A0}, 0.2C_{A0}] kmol/m^3$, $\mu \in [-3.5, 3.5] K/min$. For simplicity, the $x = [x_1, x_2]^T = [C_A, T_R]^T$ and $u = [u_1, u_2]^T = [C_{A0}, Q_\sigma]$ represent the state vector and input variables, respectively. Let x_0 denote the equilibrium. Then the constraint (3.24) is:

$$\begin{aligned}
&\inf_{u\in U}\ \sup_{\Delta f(x),\Delta g(x)}\ \nabla V(x)[f(x)+\Delta f(x)+(g(x)+\Delta g(x))u] \qquad (3.101)\\
&=(x-x_0)^TPf+\min_{u_2}(x-x_0)^TP[0,\frac{1}{\rho c_pV}]^Tu_2+\max_{\Delta f}(x-x_0)^TP\Delta f\\
&+\frac{\underline{u}_1-\overline{u}_1}{2}||((x-x_0)^TP)_1\frac{0.2F}{V}|+((x-x_0)^TP)_1\frac{F}{V}|\\
&+\frac{\underline{u}_1+\overline{u}_1}{2}(|((x-x_0)^TP)_1\frac{0.2F}{V}|+((x-x_0)^TP)_1\frac{F}{V})\\
&=(x-x_0)^TPf-90|(x-x_0)^TP[0,\frac{1}{\rho c_pV}]^T|+3.5|((x-x_0)^TP)_2|\\
&+\frac{\underline{u}_1-\overline{u}_1}{2}||((x-x_0)^TP)_1\frac{0.2F}{V}|+((x-x_0)^TP)_1\frac{F}{V}|\\
&+\frac{\underline{u}_1+\overline{u}_1}{2}(|((x-x_0)^TP)_1\frac{0.2F}{V}|+((x-x_0)^TP)_1\frac{F}{V})
\end{aligned}$$

where the sub-index also represents the term in the vector. (3.101) is still a piecewise function in terms of the single parameter P_{ij}, thereby facilitating our algorithm.

We also define the $\tilde{\Theta}=\{x|0.56kmol/m^3\leqslant x_1\leqslant 0.58kmol/m^3, 395K\leqslant x_2\leqslant 395.5K\}$. Then for the quadratic RCLF: $V(x)=(x-x_0)P(x-x_0)^T$, the initial guess of P is derived from linearization of the nonlinear system and solving the Ricatti equation:

$$P_{initial}=\begin{bmatrix}20.1669 & 1.5958\\ 1.5958 & 0.1844\end{bmatrix}$$

The ROA is shown in Fig. 3.10. There is $\Phi(P_{initial})=2.1168$. It is important to note that Θ derived from the problem 3 is required to cover the $\tilde{\Theta}$. However, the true residual set Θ can be smaller than the solution of this problem.

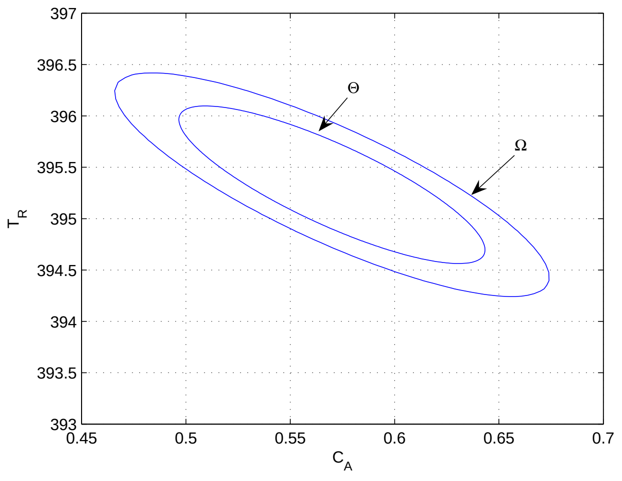

Figure 3.10: The region of attraction for $P_{initial}$.

By using the proposed algorithm, the new P is yielded:

$$P_{final} = \begin{bmatrix} 51.6785 & 2.3359 \\ 2.3359 & 0.1245 \end{bmatrix}$$

There is $\Phi(P_{final}) = 18.89$. The Ω and the residual set derived from problem 3 is shown in Fig. 3.11. Compare the ROA of these two RCLFs, it is clear that the proposed method enlarge the stabilizable region dramatically. Due to the non-convexity of the optimization formulation, we admit that this is not the largest ROA among all possible RCLFs. Changing the initial guess and increasing the sampling points in each iteration may improve the solution.

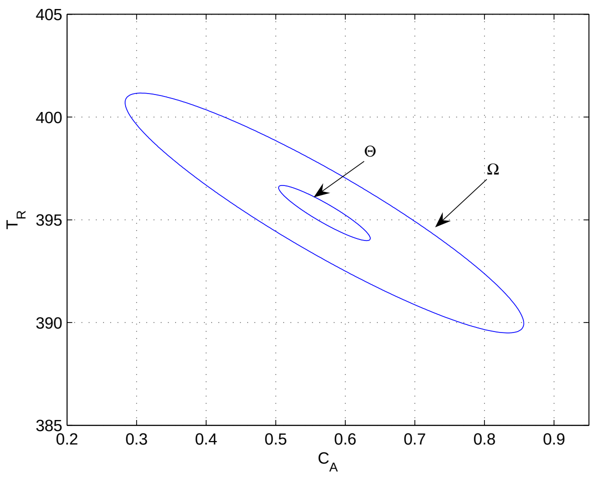

Figure 3.11: The region of attraction for P_{final}.

Given this RCLF, there always is a feedback control law that can drive this process towards the specified equilibrium as long as the initial point is within this ROA, regardless of the uncertainty.

3.10.3 Case3: Coupled Tanks System

Consider the coupled tanks system, which has been studied in [2, 94]. The general dynamic model of this system can be written as:

$$\dot{z}_1 = -a_1\sqrt{z_1} + a_2\sqrt{z_2} \tag{3.102}$$

$$\dot{z}_2 = a_1\sqrt{z_1} - 2a_2\sqrt{z_2} + \frac{u}{C} \tag{3.103}$$

where $z_1 = h_2 > 0cm$, $z_2 = h_1 - h_2 > 0cm$, $0 \leqslant u \leqslant 50cm^3/s$ is the input. The aim of the control is keeping the height of tank 2, h_2 in the specified value. The parameters of the system are shown in Table 3.2.

Table 3.2: Parameters of the process model

$a_1 = 0.0511\sqrt{cm}/s$	$a_2 = 0.1234\sqrt{cm}/s$	$C = 208.2cm^2$

Actually, this system can be regulated by backstepping, feedback linearization or slide mode control. However, the conventional CLF design approach may be powerless because of the non-integer power and the state constraint. Thus, we employ this example to show the competence of our algorithm to overcome those difficulties. Moreover, the non-quadratic CLF is also used here:

$$\begin{aligned} V &= \alpha_1(z_1 - z_{10})(\sqrt{z_1} - \sqrt{z_{10}}) + \alpha_2(z_2 - z_{20})(\sqrt{z_2} - \sqrt{z_{20}}) \\ &\quad + \alpha_3(z_1 - z_{10})(z_2 - z_{20}) \end{aligned} \tag{3.104}$$

where $z_{10} = 6cm$ and $z_{20} = 1.0273cm$ are the specified steady state. It is worthwhile to note that how to design the formulation of the CLF is a broad topic and beyond the scope of this paper. Thus, we just specified this form and it is not difficult to verify that $\phi_k(\{z_{10}, z_{20}\}) = 0$ and $\frac{\partial \phi_k}{\partial z}|_{z_{10},z_{20}} = 0$. Here we consider two different target region: $\Psi_1 : \{3cm \leqslant z_1 \leqslant 3.5cm, 0.8cm \leqslant z_2 \leqslant 1cm\}$ and $\Psi_2 : \{10cm \leqslant z_1 \leqslant 10.5cm, 2cm \leqslant z_2 \leqslant 2.2cm\}$

The initial guess of the parameters are

$$\alpha_1 = 15 \quad \alpha_2 = 3 \quad \alpha_3 = -1 \tag{3.105}$$

In Figs. 3.12 and 3.13, both target areas are far away from the scope of the ROA for the initial CLF. Note that in this case, the physical boundary of the z_2 constrained the size of the ROA. By employing the proposed approach, the new parameters to stabilize Ψ_1 are:

$$\alpha_1 = 12.4873 \quad \alpha_2 = 42.7937 \quad \alpha_3 = -1 \tag{3.106}$$

We also verify the small control property, there is

$$P_{v1} = \begin{bmatrix} 3.9255 & -1 \\ -1 & 31.9118 \end{bmatrix}$$

and let $\nu = 1$

$$Q = -(A^T P_{v1} + P_{v1}A - P_{v1}BB^t P_{v1}) = \begin{bmatrix} 0.1027 & -0.7045 \\ -0.7045 & 7.9152 \end{bmatrix}$$

which is a positive definite matrix. The Fig. 3.12 showed the ROA for the initial CLF and the derived CLF. The track of the process under the proposed control law is also presented.

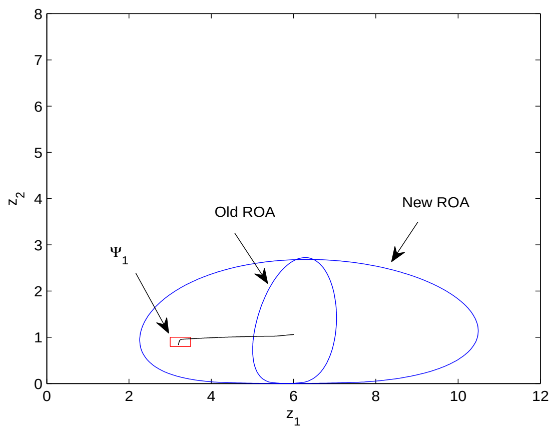

Figure 3.12: The comparison of ROA to stabilize Ψ_1

By the same way, for Ψ_2, there are

$$\alpha_1 = 9.4903 \quad \alpha_2 = 40.4300 \quad \alpha_3 = -3.3520 \tag{3.107}$$

To check the small control property:

$$P_{v1} = \begin{bmatrix} 3.0078 & -3.3520 \\ -3.3520 & 30.1628 \end{bmatrix}$$

and let $\nu = 1$

$$Q = -(A^T P_{v1} + P_{v1} A - P_{v1} B B^t P_{v1}) = \begin{bmatrix} 0.1328 & -0.9428 \\ -0.9428 & 7.7732 \end{bmatrix}$$

which is a positive definite matrix. The Fig. 3.13 showed the ROA for the initial CLF and the derived CLF. The system finally can reach the steady state by the proposed control algorithm.

3.11 Conclusion

In this work, a systematic design method for the control Lyapunov function (CLF) and robust control Lyapunov function (RCLF), which can stabilize control affine system with input and state constraints, is proposed. The problem for obtaining such a CLF/RCLF is formulated as a min-max style optimization problem and a derivative-free optimization scheme is proposed to obtain the optimal CLF/RCLF starting from a simple initial guess. The bounded control law is also modified to handle infinity norm constraint for the input. The condition for small control property is also presented. Future work includes further acclerating the optimization procedure and the design respect to the robust performance.

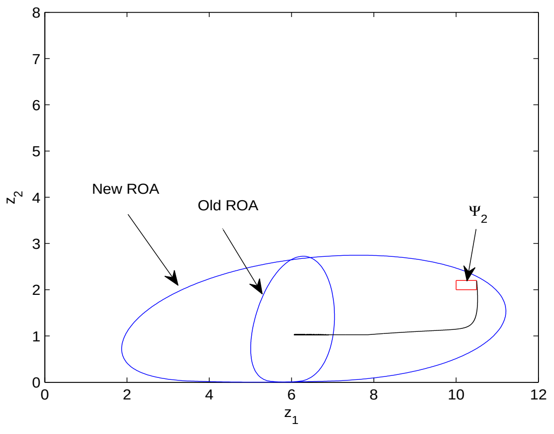

Figure 3.13: The comparison of ROA to stabilize Ψ_2

Chapter 4

Advanced Control by Approximate Dynamic Programming

4.1 Introduction

Currently, as most of processes need to be operated under tighter performance specifications and more constraints, the nonlinear model predictive control (NMPC) has become an attractive strategy for nonlinear systems in process industry. Despite its advantages of using nonlinear models and handling constraints explicitly, conventional NMPC has outstanding issues for practical applications; among them are excessive online computational burden, stability, and robustness under uncertainty. In the remainder of this chapter, we will not distinguish two terminologies: NMPC and MPC. Both of them refer to the same methodology.

Since MPC strategy employs the nonlinear differential algebra equation (DAE) model in solving a constrained finite horizon optimal control problem for the current state of the plant at each sampling time, the resulting online optimization problem becomes a large-scale nonlinear program. This requires excessive computational time, which renders conventional MPC algorithms unsuitable for many practical applications. In favor of the online computation, a fast min-max MPC employing linear programs formulation is developed [81], but it is only applicable for the 1-norm cost function. Explicit MPC divides the state space into several regions and computes an explicit control law off-line for each region [8]. It is very difficult to extend this approach to the high dimension case. A robust constrained

*A brief version of this chapter has been accepted by the 11th International Conference on Control, Automation and Systems.

MPC algorithm with explicit control law is developed, based on the concept of asymptotically stable invariant ellipsoid [125]. A neuro-fuzzy network is trained to approximate the input-output relationship of a MPC, thereby arranging the online optimization to be done off-line [49].

Due to the receding horizon nature of MPC implementation, the closed-loop stability can be guaranteed under particular conditions only. A significant survey about the stability issue of the MPC is provided in [80]. A computationally expensive equality terminal state constraint is added to MPC design [58]. MPC combined with the terminal penalty instead of the difficult equality constraint is proposed to ensure that the process finally enters the region of attraction in the absence of model uncertainty [117]. Sorts of endpoint penalty for the MPC are also developed in [21, 87, 78], respectively. The LMPC method [84], incorporates the control Lyapunov function (CLF) as the constraint into the optimization formulation to guarantee the close-loop trajectory of system dynamics follow the descent direction of the Lyapunov function. This, however, requires the CLF with large enough region of attraction (ROA). It is indicated that the prior CLF or the terminal cost, should be viewed as the approximation of the infinite horizon value function rather than the penalty [53]. A time varying terminal constraint set also can be used to achieve the stability in relative short horizon [126]. The input-to-state stability for the hybrid system is further considered by [68].

As for the model uncertainties, several examples to illustrate non-robust MPC are provided in the [44]. The plant uncertainties are explicitly incorporated into the model and the linear matrix inequality (LMI) is used to obtain the solution [63]. In [67], a piecewise affine control law is derived to maintain the controlled trajectory in a tube against the disturbances. The worst case MPC is also proposed to reject the disturbances and applied to a pilot plant [45].

Recently, the approximate dynamic programming (ADP), or called the reinforcement learning in artificial intelligence community, has been extended to the process control field to overcome the model uncertainty and meet the online computation requirements [72, 88]. The main idea of the ADP-based control approach is to encode the multistage objective values under a particular control policy into the value function because the Bellmans optimality equation gives us a mechanism for solving the intractable multi-stage optimization problems in a simple and elegant way. The approximate value function is iteratively updated given the observation from operation or simulation until its convergence. The converged value function is deemed close to the true optimal value function, if the approximation error is properly controlled in the learning stage. Theoretically, the MPC outperforms the ADP if the accurate model is known. However, since the accurate model usually is very difficult to obtain due to the lack of knowledge on the process, the ADP has tremendous potentials in some practical cases provided the operational data is enough. Moreover, the ADP only needs to solve the single step optimization, which can remarkably reduce computational burden compared to MPC.

A major issue in application of ADP to control problems is its closed-loop stability, which is the most essential requirement of the regulation, is not guaranteed theoretically in the existing literature. One of the main reasons is because the ADP is a data-based approach and mainly applied in the model-free applications.

In this work we propose a mixed control strategy where an ADP controller can be employed reliably to improve the performance of MPC for some particular states. A robust control Lyapunov function (RCLF) is generated by the proposed algorithm in last chapter and embed into the MPC scheme. Then, the ADP control strategy is developed based on the operational data of the MPC policy. Moreover, a data description technique is introduced to characterize the explored regions in the state space to prevent undue extrapolation in using the ADP controller and MPC is employed for the rest of regions. The merit of the proposed method is that the worst case is only considered for the stability analysis and the performance improvement in MPC can be achieved via ADP. By exploring the historical data in real time, the ADP control policy not only can learn the desirable behavior of the current control law, but also adaptively finds a better control policy.

The rest of this chapter is organized as follows: Section 4.2 gives the control framework of MPC combined with the RCLF; Section 4.3 develops the ADP and MPC mixed control strategy; The case study is illustrated in Section 4.4 to show the performance improvement of this mixed method in contrast to the conventional MPC. The conclusion is drawn in Section 4.5.

4.2 Main Result

In this chapter, we mainly focus on the control affine system with uncertainties, such that:

$$\dot{x} = F(x, u, \Delta f, \Delta g) = f(x) + \Delta f + (g(x) + \Delta g)u \tag{4.1}$$

where $\Delta f,\ \Delta g \in \mathcal{W}$ are bounded uncertainties. One should note that these specifications are exactly same with the last chapter for the RCLF design.

4.2.1 The MPC Control Law based on Robust Control Lyapunov Function

Most of MPC applications suffer from the stability issue because of the incompatibility between the receding horizon scheme and the asymptotic process behavior under the presence of uncertainties. In order to overcome this shortcoming of the MPC, the RCLF is combined with the conventional MPC formulation to enhance its stability. In this section, two of MPC approaches based on the prior RCLF is presented briefly. A discrete implementation of Lyapunov based controller is first introduced.

Discrete Implementation of Lyapunov based Control

Given a RCLF, let us first define the Lyapunov derivative $L_F V$:

$$L_F V(x, u, \Delta f, \Delta g) := \nabla V(x) \bullet F(x, u, \Delta f, \Delta g) \tag{4.2}$$

Define the u^*, such that

$$u^*(x) = \arg \inf_{u \in U} \sup_{\Delta f \in \mathcal{W}, \Delta g \in \mathcal{W}} \nabla V(x) \bullet F(x, u, \Delta f, \Delta g) \tag{4.3}$$

Namely, the u^* minimizes $L_F V$ under the worst uncertainties determined by the following Lemma. Here the "worst" means the $\{\Delta f, \Delta g\}$, maximizing $L_F V(x, u)$.

Lemma 15. *For any state x and input u, the worst uncertainties of Δf and Δg in (3.78) are determined by the sign of each component in $\nabla V(x)$ and $\nabla V(x)u$, respectively.*

Proof. Let the $\Delta f^*(x, u)$ and $\Delta g^*(x, u)$ denote the worst uncertainties maximizing $L_F V(x, u)$, given state x and input u. Then, for each component of uncertainty vector and matrix, we have

$$\Delta f^*(x, u)_i = \arg \max_{\Delta f_i} \nabla V(x)_i \Delta f_i \tag{4.4}$$

and

$$\Delta g^*(x, u)_{ij} = \arg \max_{\Delta g_{ij}} \nabla V(x)_i \Delta g_{ij} u_j \tag{4.5}$$

Also, the bounds of these components are provided in **Definition 9**. Thus, their values can be derived explicitly.

$$\Delta f_i^* = \begin{cases} \Delta_i & \text{if } \nabla V_i > 0 \\ -\Delta_i & \text{if } \nabla V_i \leqslant 0 \end{cases}$$

$$\Delta g_{ij}^* = \begin{cases} \Upsilon_{ij} & \text{if } \nabla V_i u_j > 0 \\ -\Upsilon_{ij} & \text{if } \nabla V_i u_j \leqslant 0 \end{cases}$$

□

Then the theorem for discrete controller is proposed as below:

Theorem 16. *For the system (4.1) and rclf: V with Ω and residual set Θ, let initial state $x(0) \in \Omega \backslash \Theta$. Then, given a set Π_R, such that $\Theta \subset \Pi_R \subset \Omega$, there always exist a set Ψ_R, satisfying $\Pi_R \subset \Psi_R \subset \Omega$, and a positive time τ with control law: $u(t) = u^*(x(0))$, such that: (1)$L_F V(x(t), u(t), \Delta f, \Delta g) < 0, \forall t \in [0, \tau)$, $\forall \Delta f, \Delta g \in \mathcal{W}$, if $V(x(0)) > C_{\Pi_R}$ (2) $V(x(t)) \leqslant C_{\Psi_R}, \forall t \in [0, \tau)$ if $V(x(0)) \leqslant C_{\Pi_R}$, where C_{Π_R} and C_{Ψ_R} are the set level of Π and Ψ, respectively.*

Proof. We follow the idea in [83] to prove this theorem. First of all, since $x(0) \in \Omega \backslash \Theta$, according to (3.8) and (4.3), one can get

$$\nabla V(x(0))\{f(x(0)) + \Delta f^*(x(0), u^*) + [g(x(0)) + \Delta g^*(x(0), u^*)]u^*\} < -\alpha_V(x(0)) \tag{4.6}$$

where $u^* := u^*(0)$. Then, in time point τ_1, the Lyapunov derivative for the worst uncertainties, i.e. $L_F V(x(\tau_1), u^*, \Delta f^*(x(\tau_1), u^*), \Delta g^*(x(\tau_1), u^*))$ can be described as follows:

$$\begin{aligned} &\nabla V(x(\tau_1))\{f(x(\tau_1)) + \Delta f^*(x(\tau_1), u^*) + [g(x(\tau_1)) + \Delta g^*(x(\tau_1), u^*)]\, u^*\} \qquad (4.7) \\ &= \nabla V(x(0))\{f(x(0)) + \Delta f^*(x(0), u^*) + [g(x(0)) + \Delta g^*(x(0), u^*)]u^*\} + \\ &\quad \nabla V(x(\tau_1))f(x(\tau_1)) - \nabla V(x(0))f(x(0)) + [\nabla V(x(\tau_1))g(x(\tau_1)) - \nabla V(x(0))g(x(0))]u^* \\ &+ [\nabla V(x(\tau_1))\Delta g^*(x(\tau_1), u^*) - \nabla V(x(0))\Delta g^*(x(0), u^*)]u^* \\ &+ \nabla V(x(\tau_1))\Delta f^*(x(\tau_1), u^*) - \nabla V(x(0))\Delta f^*(x(0), u^*) \end{aligned}$$

Considering that system (4.1) is continuous and Ω as well as Θ are bounded, there exists a number K_1, for all $x(0) \in \Omega \backslash \Theta$, Δf, $\Delta g \in \mathcal{W}$ and $t \in [0, \tau_1)$, satisfying $||x(t) - x(0)|| \leqslant K_1 \tau_1$. Since function $\nabla V \bullet f$ and $\nabla V \bullet g$ are both continuous, there are real positive numbers K_2 and K_3, for all $x(0) \in \Omega\backslash\Theta$, Δf, $\Delta g \in \mathcal{W}$ and $t \in [0, \tau_1)$, such that

$$||\nabla V(x(t))f(x(t)) - \nabla V(x(0))f(x(0))|| \leqslant K_2K_1\tau_1 \tag{4.8}$$

$$||[\nabla V(x(t))g(x(t)) - \nabla V(x(0))g(x(0))]u^*|| \leqslant K_3K_1\tau_1 \tag{4.9}$$

For the continuous function $\nabla V(x)_i$, there is also a positive real number M_i, for all $x_0 \in \Omega\backslash\Theta$, Δf, $\Delta g \in \mathcal{W}$ and $t \in [0, \tau_1)$, satisfying

$$||\nabla V(x(t))_i - \nabla V(x(0))_i|| \leqslant M_iK_1\tau_1 \tag{4.10}$$

Then we need to discuss two cases.

Case1: $\nabla V(x(0))_i\nabla V(x(t))_i > 0$. According to Lemma15, $\Delta f^*(x(t), u^*)_i = \Delta f^*(x(0), u^*)_i$ and $\Delta g^*(x(t), u^*)_{ij} = \Delta g^*(x(0), u^*)_{ij}$. Then,

$$||\nabla V(x(t))_i\Delta f^*(x(t), u^*)_i - \nabla V(x(0))_i\Delta f^*(x(0), u^*)_i|| \leqslant M_i\Delta_iK_1\tau_1 \tag{4.11}$$

$$||[\nabla V(x(t))_i\Delta g^*(x(t), u^*)_{ij} - \nabla V(x(0))_i\Delta g^*(x(0), u^*)_{ij}]u^*_j|| \leqslant M_i\Upsilon_{ij}K_1\tau_1u^*_j \tag{4.12}$$

Case2:$\nabla V(x(0))_i\nabla V(x(t))_i \leqslant 0$. According to Lemma15, the worst uncertainties switch to the different bound during $[0, t]$, namely, $\Delta f^*(x(t), u^*)_i = -\Delta f^*(x(0), u^*)_i$ and $\Delta g^*(x(t), u^*)_{ij} = -\Delta g^*(x_0, u^*)_{ij}$. However, in this scenario, we have

$$||\nabla V(x(0))_i + \nabla V(x(t))_i|| \leqslant |\nabla V(x(0))_i - \nabla V(x(t))_i|| \leqslant M_iK_1\tau_1 \tag{4.13}$$

Therefore, Eqs. (4.11) and (4.12) still hold.

Combining Eqs. (3.8), (4.8), (4.9), (4.11) and (4.12), the upper bound of (4.7) can be derived:

$$\begin{aligned} & \nabla V(x(t))\{f(x(t)) + \Delta f^*(x(t), u^*) + [g(x(t)) + \Delta g^*(x(t), u^*)]u^*\} \\ < \quad & -\alpha_V(x(0)) + K_2K_1\tau_1 + K_3K_1\tau_1 + M^T\Delta K_1\tau_1 + M^T\Upsilon u^*K_1\tau_1 \end{aligned} \tag{4.14}$$

To specify the holding time, as long as $\tau_1 < \frac{\alpha_V(x(0))}{K_2K_1+K_3K_1M^T\Delta K_1+M^T\Upsilon u^*K_1}$, one can guarantee that $\forall t \in [0, \tau_1]$, $L_FV(x(t), u^*, \Delta f^*(x(t), u^*), \Delta g^*(x(t), u^*)) < 0$.

If $V(x(0)) \leqslant C_{\Pi_R}$, we first define $d_{\Pi,t} = \max_{u\in U,x(0)\in\Pi_R,\Delta f\in\mathcal{W},\Delta g\in\mathcal{W}} V(x(t))$. Then due to $C_\Pi < C_\Gamma$ and the continuity of system, there is a positive time τ_2, such that $\forall t \in [0, \tau_2]$ $d_{\Pi_R,t} < C_\Omega$. We can specify $\tau = min\{\tau_1, \tau_2\}$ and $C_{\Psi_R} = d_{\Pi_R,\tau}$. Then it implies that $V(x(t)) \leqslant C_{\Psi_R}$, $\forall t \in [0, \tau)$.

If $V(x(0)) > C_{\Pi_R}$, because of $C_\Theta < C_{\Pi_R} < C_\Omega$ and $\tau < \tau_1$, $\forall t \in [0, \tau)$, there is

$$L_F V(x(t), u^*, \Delta f^*(x(t), u^*), \Delta g^*(x(t), u^*)) < 0$$
$$\longrightarrow L_F V(x(t), u^*, \Delta f, \Delta g) < 0, \forall \Delta f, \Delta g \in \mathcal{W}$$

$\square$

The **Theorem 16** claims the existence of holding time τ for the discrete controller implementation. Moreover, it can conclude that increasing negativity margin α_V or reducing uncertainties bound can directly prolong the allowable holding time. Alternatively, one can also shrink Π_R or enlarge permissible Ψ_R to get longer τ.

Lyapunov based Model Predictive Control (LMPC)

Given a small enough holding time τ, the formulation of Lyapunov based model predictive control (LMPC) is proposed as below:

$$\min_{u(0),u(\tau),\dots,u(h_c\tau)} \sum_{i=0}^{h_c} x(i\tau)^T Q x(i\tau) + u(i\tau)^T R u(i\tau) \tag{4.15}$$

$$\text{s.t.} \quad \dot{x} = f(x) + g(x)u \tag{4.16}$$

$$\dot{\hat{x}} = f(x) + \Delta f^*(x(0), u(0)) + (g(x) + \Delta g^*(x(0), u(0)))u(0) \tag{4.17}$$

$$\text{if} \quad V(x(0)) \geqslant C_{\Pi_R} :$$

$$V(\hat{x}(\tau)) < V(x(0)) - \epsilon_R \tag{4.18}$$

$$\text{if} \quad V(x(0)) \leqslant C_{\Pi_R} :$$

$$V(\hat{x}(\tau)) \leqslant C_{\Psi_R} \tag{4.19}$$

$$u(0), u(\tau), \dots, u(h_c\tau) \in U \tag{4.20}$$

where h_c is the control horizon; Q and R are strictly positive definite matrix to weight the cost of the control; U is the admissible input; V is the RCLF; ϵ_R is a small enough positive real number. This formulation is only used for initial states within the ROA of V.

In this formulation, the one step estimation of worst state $\hat{x}(\tau)$ in terms of V is maintained. Since the **Theorem16** guarantees that the Lyapunov derivative is negative during $t \in [i\tau, (i+1)\tau)$, $\forall \Delta f, \Delta g \in \mathcal{W}$, provided that the holding time is small enough, the constraint (4.18) is easy to be satisfied. ϵ_R is employed to force V decreasing fast. Although this constraint only renders the Lyapunov function along the process trajectory decline in one step, the robust stability is still reserved because only the first action is implemented and the control action sequence will be re-calculated in the new state. Moreover, considering that worst case occurs rarely in practice and the multi-step prediction of the worst scenario is very time consuming and inaccurate, there is no reason to spend many computations on the conservative solution. Constraint (4.19) is also derived from **Theorem16** to ensure Ψ be an invariant set. Solving this formulation online, it yields an input to drive the system finally falling into the region Ψ_R from any initial state within the ROA. However, parameters τ, ϵ_R, C_{Π_R} and C_{Ψ_R} should be well-tuned to guarantee the performance, which may not be easy.

Unconstrained receding horizon control

Despite employing the RCLF as the hard constraint guarantees the stability for the nonlinear control in the presence of disturbances, this scheme suffers at least two drawbacks: The first, adding more constraints renders the optimization more difficult to be solved; secondly, it can not stabilize the state outside the ROA. To address these issues, the unconstrained finite horizon optimal control framework is proposed in [53]. This method uses a prior CLF as the terminal cost to approximate the tail of the infinite horizon cost-to-go function and it is shown that the stability can be guaranteed if the system can enter the ROA in the last stage of the control horizon and that CLF is an appropriate upper bound of the real cost-to-go. For the system with uncertainties, no conclusion is made to guarantee the robust stability for the states outside the ROA. However, treating RCLF as the terminal cost is still useful to enhance the stability.

$$\min_{u(0),u(\tau),\ldots,u(h_c\tau)} V(x((h_c+1)\tau)) + \sum_{i=0}^{h_c} x(i\tau)^T Q x(i\tau) + u(i\tau)^T R u(i\tau) \tag{4.21}$$

$$\text{s.t.} \quad \dot{x} = f(x) + g(x)u \tag{4.22}$$

$$u(0), u(\tau), \ldots, u(h_c\tau) \in U \tag{4.23}$$

where V is a CLF. Although the formulation (4.21) is not computationally expensive, to determine a suitable V as a terminal cost is not trivial. An excessively large V dominates the optimization and degrades the control performance. People who interests the detail work of CLF based unconstrained NMPC can refer to [53].

4.2.2 The Hybrid Control Strategy

Value Function Approximation

Despite the considerable success achieved by the MPC for the control of plant in the chemical industry, there is still some room to be improved especially when the initial state is outside of the ROA. In that case, the MPC needs the long controlled and predictive horizon to guarantee the final state enter the ROA, whereas increases the number of decision variables. The resulting formulation is computationally prohibitive, large-scale nonlinear program.

In order to reduce the computational complexity of MPC, an approximate dynamic programming (ADP) scheme is proposed in this work. As mentioned in the introduction part, the dynamic programming (DP) is an effective algorithm to generate the optimal control law by computing the optimal value function $J^*(x)$ for each state and has two major advantages over MPC; the online computational requirement of DP is much less than that of MPCs, and a closed-loop optimal control policy can be derived using DP for systems with uncertainties [72]. Classical DP algorithms, including value iteration and policy iteration, require all possible states to be involved in iteratively solving for the optimal value function. The computational burden exponentially increases in state dimensions, which makes it difficult to apply these algorithms to continuous state systems. Central to the DP is obtaining the optimal value (cost-to-go) function, which is usually too complex to be figured out exactly for nonlinear system with constraints. Thus an approximation scheme is a practical choice to solve DP.

Let us first consider the true optimal value function. With the presence of the disturbances, the practical process should be described as a stochastic system. The optimal control policy should minimize the expected cost over the infinite horizon:

$$J^*(x(0)) = \min_{u(i\tau)} E\{\sum_{i=0}^{T} r(x(i\tau), u(i\tau), w(i\tau))\} \tag{4.24}$$

where $r(x, u)$ is the stage-wise cost; T is the terminal step; τ is the holding time of discrete controller; $J^*(x_0)$ is the optimal value function for state x_0; w is the random disturbance. Here we assume the existence of the terminal state, thereby only considering the un-discounted case. Based on the Bellman optimal principal, in any t^{th} step, the (4.24) can be rewrote as:

$$J^*(x(t\tau)) = \min_{u(t\tau)} E\{r(x(t\tau), u(t\tau)) + J^*((t+1)\tau)\} \tag{4.25}$$

where $x((t+1)\tau)$ is the successive state of $x(t\tau)$ under the input $u(t\tau)$. Note that the expectation operator inside the minimization is a major source of computational complexity. Hence, we employ the post-decision state x^p to avoid the minimization over the expectation. Specifically, the x^p indicates the state after the external input acts on the system immediately but before the uncertainties are realized. For the discrete control affine system with uncertainties,

$$x((t+1)\tau) = f_d(x(t\tau)) + \Delta f_d(x(t\tau)) + g_d(x(t\tau))u + \Delta g_d(x(t\tau))u \quad (4.26)$$

The $x^p(t\tau)$ and $x((t+1)\tau)$ can be expressed by decomposing the (4.26)

$$\begin{aligned} x^p((t+1)\tau) &= f_d(x(t\tau)) + g_d(x(t\tau))u &(4.27)\\ x((t+1)\tau) &= x^p((t+1)\tau) + \Delta f_d(x(t\tau)) + \Delta g_d(x(t\tau))u &(4.28)\end{aligned}$$

Then, we define the optimal value function on the post-decision state:

$$J^*(x^p((t+1)\tau)) = E\{J^*(x((t+1)\tau))|x^p((t+1)\tau)\} \quad (4.29)$$

Based on (4.29), optimal input is much easier to be calculated:

$$\begin{aligned} u^*(t\tau) &= arg\min_{u(t\tau)} E\{r(x(t\tau), u(t\tau)) + J^*(x((t+1)\tau))\} &(4.30)\\ &= arg\min_{u(t\tau)} r(x(t\tau), u(t\tau)) + E\{J^*(x((t+1)\tau))\} &(4.31)\\ &= arg\min_{u(t\tau)} r(x(t\tau), u(t\tau)) + J^*(x^p((t+1)\tau)) &(4.32)\end{aligned}$$

Here we suppose that the stage-wise cost is deterministic.

As a result, different from the traditional function approximation scheme, we need to construct the approximator to approach the value function of the post-decision state. Given the simulation or operational data, every trajectory leading to the ROA provides several sample states. Suppose that there is a sequence of data under a fixed control law π,

$$\{x(0), u(0)\}, \{x(\tau), x^p(\tau), u(\tau)\}, \ldots, \{x(T\tau), x^p(T\tau), u(T\tau)\} \quad (4.33)$$

where terminal step T depends on the earliest time of process entering the ROA; The superscript p indicates the post-decision state, derived from the nominal model. Since the initial state $x(0)$ is determined, the post-decision state sequence begins at the second sampling time and we define the instance based post-decision state value function for $x^p((T-1)\tau)$ as:

$$\hat{J}^\pi(x^p((T-1)\tau)) = x((T-1)\tau)^T Q x((T-1)\tau) + u((T-1)\tau)^T R u((T-1)\tau) \quad (4.34)$$

where $k \leqslant T-1$. The recursive formulation is:

$$\hat{J}^{\pi}(x^p(k\tau)) = x(k\tau)^T Q x(k\tau) + u(k\tau)^T R u(k\tau) + \hat{J}^{\pi}(x^p((k+1)\tau)) \tag{4.35}$$

where Q and R are the weighting matrix for the state and input, respectively. The terminal cost, namely, the value function of states in the ROA is specified as zero, which means no stage-cost incurred after step T. That is because the regulator can be switched to the LMPC after that time point, with guaranteed stability.

The choice of the appropriate basis function has received much attention in approximating value function because it fundamentally determines the error of the approximate function. In the proposed approach, the whole post-decision state space is divided into finite number of regions which is implicitly defined by a data clustering method. A local approximator for the state region corresponding to each cluster is estimated. The advantage of this scheme over the conventional global approximation lies in its accuracy. Moreover, considering that some of states are never related according to the available control policy, incorporating all these states into the same approximation framework is not reasonable. For each cluster, the polynomial function is adopted as the basis because it is shown that the solution of Hamilton-Jacobi-Bellman (HJB) equation can be accurately represented by polynomials via the Taylor series [22]. As most of existing methodologies, we also consider the linear approximate value function:

$$\tilde{J}^{\pi}(x) = \sum_{h=0}^{k} \sum_{j=1}^{m_h} \theta_{hj} \sum_{i=1}^{n} x_i^{\alpha_i} \tag{4.36}$$

where k is the highest order of the polynomial; m_h is the total number of different h degree monomials; n is the number of state dimensions; θ_{hj} is the coefficient; x_i denotes the ith component of the state vector. Also, there is $\sum_{i=1}^{n} \alpha_i = h$. For simplicity, we assume that the equilibrium is $\{0\}$. Since the instance based value function and the post-decision state can be derived for each data, the parameter θ is estimated off-line by the least square method.

Support Vector Data Description (SVDD)

Usually, the function approximation based controller encounters two issues [71]. The first, the errors of the approximation is hard to control. Whether the approximation is reliable in those unexperienced region is not fully predictable. Moreover, the pure ADP controller may be aggressive and more likely to drive the system to the boundary of the feasible region, where the system is running with high risks but the value function is supposed to be small according to the $\tilde{J}$. A similar problem has been reported in [71] where the nonparametric local averager is used to estimate the value function. They ascribed it to the excessive extrapolation and designed the data density penalty to control the value function in those unexperienced region.

Secondly, the unstable trajectory also provides the useful information to keep process from the risk but hard to be incorporated into the approximation scheme because the choice of the suitable value function for these regions is not an easy task. Assigning a constant, high value for the regions in the unstable trajectories may render the resulting approximator non-smooth, thereby amplifying the error.

In order to address these problems, we introduce the support vector data description (SVDD) to characterize those reliable region, i.e. the sampled and stable area. The SVDD is developed mainly for the one class classification to distinguish target and outlier data. In its scheme, the stable and sampled states are labeled as "target" and the unstable data is viewed as "outlier". The aim is to develop a suitable hypersphere to contain all those targets and exclude outliers. To this end, an optimization problem is solved to yield the hypersphere [119]:

$$\min_{R,\xi_i,\zeta_l,a} R^2 + \sum_i \xi_i + \sum_i \zeta_l \tag{4.37}$$

$$\text{s.t.} \quad ||z_i - a||^2 \leqslant R^2 + \xi_i \tag{4.38}$$

$$||z_l - a||^2 \geqslant R^2 - \zeta_l \tag{4.39}$$

$$\xi_i \geqslant 0 \quad \zeta_l \geqslant 0 \tag{4.40}$$

where R is the radius of the hypersphere; z_i and z_l are the target and outlier data, respectively; ξ and ζ are the slack variables which allow a few of samples be identified incorrectly; a is the center of the hyperplane. To determine whether a new data z to be an outlier or target, it is only necessary to compare $||z - a||$ with R. In practice, for the purpose of making the data description more flexible, it is common to use kernel function, such as Gaussian, to replace the inner product. Details for implementation of SVDD can be found in [119].

Control Scheme

Once the sampled and stable area is determined by SVDD, the corresponding boundary constraints can be added to the Bellman equation to select the optimal input. The formulation of this optimization can be derived as:

$$\min_{u,i} J = r(x,u) + \tilde{J}_i(x^p) \tag{4.41}$$

$$\text{s.t.} \quad \dot{x}^p = f(x) + g(x)u \tag{4.42}$$

$$||x^p - a_i||^2 \leqslant R_i^2 \tag{4.43}$$

$$u \in U \tag{4.44}$$

$$i \in \{1, 2, 3, \ldots, M\} \tag{4.45}$$

where U is the admissible input. $\tilde{J}_i$, a_i and R_i are the approximate value function for the previous control law, hypersphere's center and radius for the i^{th} cluster, respectively. M is the total number of clusters. A kernel function in SVDD can be used to substitute for the inner product in constraint (4.43). Since the value function is constructed based on the post-decision state space, only the nominal model is employed to predict the process dynamics.

It is worth noting that (4.41) is not a common optimization problem, because the $\tilde{J}_i$ and constraint (4.43) can vary with the cluster i. However, since we can make the one step prediction to find all possible x^p, it is not difficult to test all possible successive clusters, usually in a small proportion of M, and select the optimal solution. Moreover, we can also borrow the idea from classical "Branch and Bound" method. Namely, $L_i = \min_{x^p} \tilde{J}_i(x^p)$ for each cluster is pre-calculated as the lower bound value function. Then one can rearrange the cluster index by the ascending order of L_i, i.e. $L_1 < L_2 \ldots, < L_N$. Then, in the online calculation, we need to solve (4.41) for all clusters sequentially and maintain the best solution J, until $L_i > J$.

If the optimization problem (4.41) is feasible, the new post-decision state will be in the reliable region under the yielded input. In case the (4.41) is infeasible, which means no subsequent state in the reliable region, the unconstrained NMPC can be used to determine the input. Thus, we call this a hybrid control strategy. The detail of this scheme is as follows:

Step 1: If the current state is within ROA, employ the LMPC until the process enters Θ; otherwise go to Step2.

Step 2: Calculate all possible successive clusters and incorporate them by set S.

Step 3: Initialize a large enough objective function J; set $i = 0$.

Step 4: Repeat $i \leftarrow i + 1$ until $i \in S$. If $i > |S|$, go to step 8, where $|S|$ is the number of elements in S.

Step 5: If $L_i < J$, solve the optimization problem (4.41) for cluster i; Otherwise, terminate the procedure and return the optimal input.

Step 6: If (4.41) is infeasible, return to Step 4.

Step 7: If (4.41) is feasible for cluster i with the solution J^i, u^i, such that $J^i < J$, let $J \leftarrow J^i, u \leftarrow u^i$. Go to Step 4.

Step 8: After checking all successive clusters, if there is no feasible solution, the unconstrained NMPC should be used; otherwise, return the optimal input u.

Online learning vs. Off-line learning

The ADP algorithm, no matter model free or model based, is featured by its online learning scheme. Although this scheme can adjust the controller's behavior quickly to accommodate the dynamic environment, it may not be suitable to the process industry. Specifically, the parameters of the value function in the online learning keep on changing and are considerably affected by the new data. As a result, the control performance may fluctuate dramatically due to the disturbances. Moreover, the learning factor is also hard to be determined because the performance of the algorithm is sensitive to its choice.

On the contrary, off-line is chosen in our case because its stable performance. Note that DP can be used to improve the control policy by the Bellman optimality operator, once the sufficient data is gathered under the current control law, above value function approximation and update procedure actually can be applied again to refine the control strategy further. Even this method is slower than the online learning, the performance of the plant is kept constant during the time between two learning iterations, which is much favorable to the process industry. Moreover, as more data is collected, the transition probability and stage-wise cost can be estimated more accurately. Thus, the corresponding control law is more reliable.

4.3 Case Study

The CSTR model is studied for the stabilization of uncertainty system, with the exactly same scenario with the last chapter. The control objective is to drive the system to the unstable steady state: $x_0 = [T_{Rs} : 395.33\ K,\ C_{As} : 0.57\ kmol/m^3]$. The parameters are shown in Table 4.1. We mainly consider two sources of noises: (1) the concentration fluctuation of the inlet flow ϵC_{A0} and (2) the disturbance μ in $\dot{T}_R$. Thus, the true model is:

$$\dot{C}_A = \frac{F}{V}(C_{A0}(1+\epsilon) - C_A) - k_0 e^{-E/RT_R} C_A \tag{4.46}$$

$$\dot{T}_R = \frac{F}{V}(T_{A0} - T_R) + \frac{-\Delta H}{\rho C_p} k_0 e^{-E/RT_R} C_A + \frac{Q_\sigma}{\rho C_p V} + \mu \tag{4.47}$$

Table 4.1: Parameters of the process model

$V = 0.1 m^3$	$R = 8.314 kJ/kmolK$
$C_{A0s} = 1.0 kmol/m^3$	$T_{A0s} = 310.0K$
$\Delta H = -4.78 \times 10^4 kJ/kmol$	$k_0 = 72 * 10^9 min^{-1}$
$E = 8.314 \times 10^4 kJ/kmol$	$C_p = 0.239 kJ/kgK$
$\rho = 1000 kg/m^3$	$F = 0.1 m^3/min$
$T_{Rs} = 395.33K$	$C_{As} = 0.57 kmol/m^3$

4.3.1 MPC Design

As mentioned before, the states within the ROA can be stabilized by LMPC directly. Different from DP, the MPC is a receding horizon framework. Therefore, we assume the control horizon as $h_c = 5$ and holding time $\tau = 0.04$ min in this example. Let $x = [C_A, T_R]$, then one can employ the following optimization:

$$\min_{u(0),u(\tau),\ldots,u(5\tau)} \sum_{t=0}^{5} (x(t\tau) - x_0)^T W (x(t\tau) - x_0) + u(t\tau)^T R u(t\tau) \tag{4.48}$$

$$\text{s.t.} \quad \dot{x} = f(x) + g(x)u \tag{4.49}$$

$$\dot{\hat{x}} = f(x) + \Delta f^*(x(0), u(0)) + (g(x) + \Delta g^*(x(0), u(0)))u(0) \tag{4.50}$$

$$\text{if} \quad V(0) \geqslant C_{\Pi_R} : V(\hat{x}(\tau)) \leqslant V(x(0)) - \epsilon_R \tag{4.51}$$

$$\text{if} \quad V(0) \leqslant C_{\Pi_R} : V(\hat{x}(\tau)) \leqslant C_{\Psi_R} \tag{4.52}$$

$$u(0), u(\tau), \ldots, u(5\tau) \in U \tag{4.53}$$

$$W = \begin{bmatrix} 200 & 0 \\ 0 & 3 \end{bmatrix}$$

$$R = \begin{bmatrix} 10 & 0 \\ 0 & 0.01 \end{bmatrix}$$

where W and R are the weights for the state and input; $V(x) = (x - x_0)^T P (x - x_0)$ is the pre-designed RCLF in the last chapter. An important issue in the LMPC

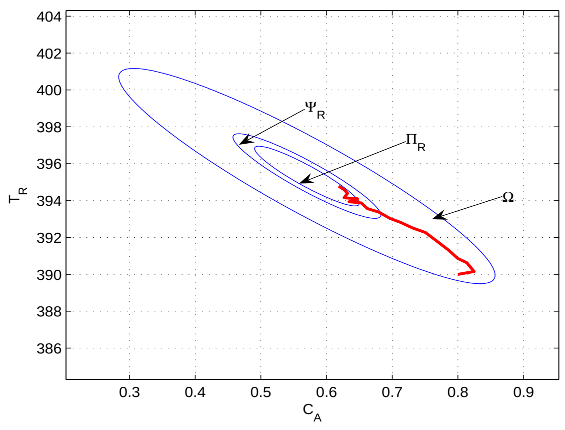

Figure 4.1: The evolvement of process under the LMPC

control scheme is the choice of parameters including $\tau, \epsilon_R, C_{\Pi_R}, C_{\Psi_R}$ as mentioned in **Theorem16**. With a reasonable sampling time τ, a smaller C_{Ψ_R} is more favorable to the control in terms of the accuracy. In the implementation for this example, the ϵ_R, C_{Π_R} and C_{Ψ_R} are specified as 0.02, 0.05 and 0.1 respectively. The Fig. 4.1 shows the effect of the LMPC control, in which the process can be driven to the Ψ_R successfully.

The LMPC can only guarantee the robust stability for states within ROA. However, the regulation of system starting from states outside the ROA is more challengeable. For those states, the objective of the control is driving the process into the ROA safely, quickly and economically. The possible options are unconstrained MPC or ADP frameworks.

We first construct the unconstrained MPC. The RCLF in last chapter can be employed as the terminal penalty in the objective function:

$$\min_{u(0),\ldots,u(8\tau)} \sum_{t=0}^{8} (x(t\tau)-x_0)^T W (x(t\tau)-x_0) + u(t\tau)^T R u(t\tau) + \omega V(x(9\tau)) \tag{4.54}$$

$$\text{s.t.} \quad \dot{x} = f(x) + g(x)u \tag{4.55}$$

$$u(t\tau) \in U \tag{4.56}$$

where $\omega = 5$ is the weight of the terminal cost; $h_c = 8$ is the control horizon. Without the control Lyapunov function constraint, the sampling time is released as $\Delta_t = 0.1$ min. Note that although the nominal model is hired in this formulation, the extened Kalman filter (EKF) can be applied to estimate the unknown parameters

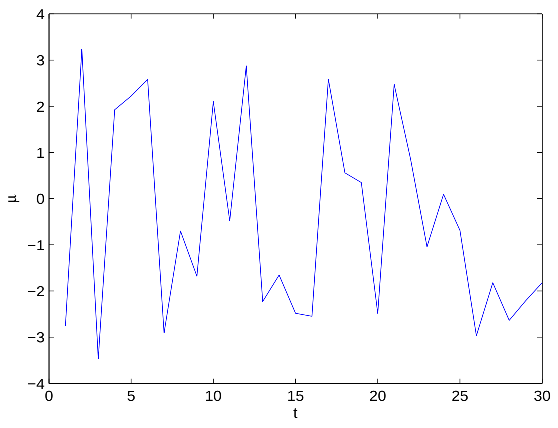

Figure 4.2: μ satisfies the uniform distribution

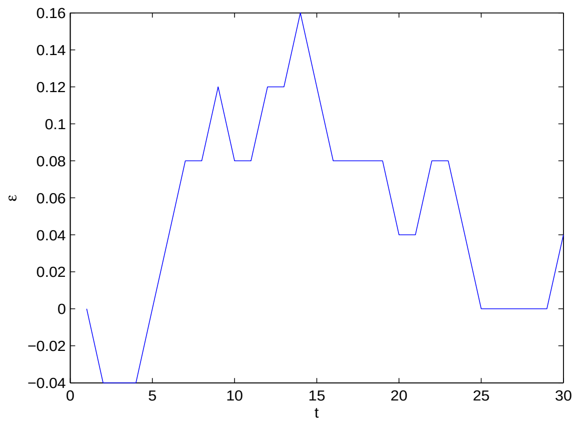

Figure 4.3: The concentration fluctuation of the inlet flow (%)

in this case. However, without the Gaussian distribution assumption, it is hardly to track the signal of the μ and ϵ accurately.

The simulations for unconstrained MPC have run a large number of times with different disturbance scenarios. Unfortunately, even with the end point penalty, the system cannot be stabilized for some states outside of ROA because of (1) disturbance (2) input constraint (3) limitation of the optimization solver. Moreover, for the stabilized cases, the control performance also can be improved. Hence, the ADP control strategy is designed in the next section based on the data provided by the MPC cases to further reduce the control cost and drive the process to the ROA faster.

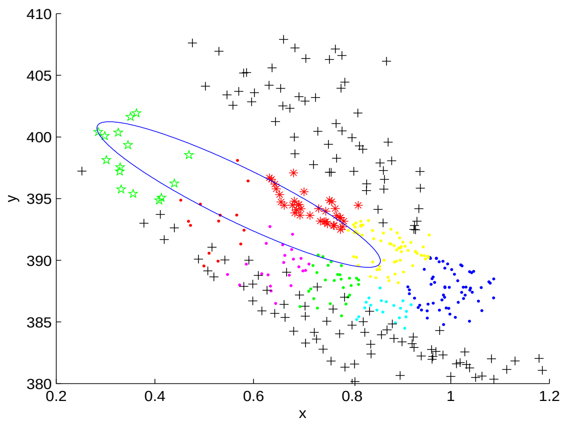

Figure 4.4: Clustered data and the ROA. The "+" represents data in the unstable trajectory

4.3.2 ADP Regulator Design

ADP controller is mainly to improve the control performance and stability for the initial state outside the ROA. We first collect the data from the unconstrained MPC by running the plant 100 times with various scenarios, i.e. initial states and disturbances. The value function for each state in the stable trajectory is calculated as illustrated above. Any states within ROA can be viewed as the terminal state, with value function zero.

The K-mean is applied in the data set to construct 10 clusters, see Fig.4.4. To balance the simplicity and approximation accuracy, 3rd order polynomial is applied for the n^{th} cluster $n \in \{1, 2, \ldots, N\}$, i.e.

$$\begin{aligned}\tilde{J}_n^{MPC}(x) &= \theta_{n1}(x_1 - x_{01})^3 + \theta_{n2}(x_2 - x_{02})^3 + \theta_{n3}(x_1 - x_{01})^2(x_2 - x_{02}) \\ &+ \theta_{n4}(x_2 - x_{02})^2(x_1 - x_{01}) + \theta_{n5}(x_1 - x_{01})^2 + \theta_{n6}(x_2 - x_{02})^2 \\ &+ \theta_{n7}(x_1 - x_{01})(x_2 - x_{02}) + \theta_{n8}(x_1 - x_{01}) + \theta_{n9}(x_2 - x_{02}) + \theta_{n10}\end{aligned} \quad (4.57)$$

where θ are unknown parameters, identified from the MPC data by the least square. x_1 is the C_A and x_2 is the T_R. $x_0 = [x_{01}, x_{02}]^T$ is the equilibrium. The K-mean is applied in the data set to construct 10 clusters, see Fig. 4.4.

It is common that some of data points are within the ROA because the post-decision state is being dealt with. One interesting issue is that most of data in those unstable trajectories is aggregated in the cluster 1 and 6. As a result, it is not necessary to derive their approximators and those regions should be forbidden

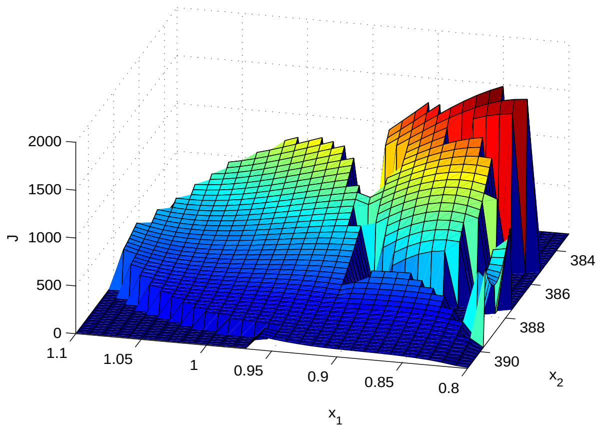

Figure 4.5: Approximate value function of the MPC control policy

in the optimization formulation. The SVDD is adopted in this scheme to detect the reliable region and isolate those unstable states.

Although the local value function scheme can promote the accuracy of the approximation dramatically, there is still a practical issue impeding the application. Due to the disturbance, the sampled value function may show considerable diversity for the closer state point under the similar input. These non-smooth data poses difficulty on the approximation. Thus, before calculating the parameters, the data should be pre-processed to reflect the expected value function for each sample state. To this end, the k-nn averager is introduced.

k-nn sample average: Given the operational data and the instance based value function $\hat{J}^{MPC}(x)$, the smoothed value function is

$$\bar{J}^{MPC}(x) = \beta_0 \hat{J}^{MPC}(x) + \sum_{i=1}^{k-1} \beta_i \hat{J}^{MPC}(X_i) \tag{4.58}$$

where the X_i is the ith nearest stable state to the current x; β_i is their weight, usually determined by the distance. Then the $\bar{J}^{MPC}$ can be used as the target value function, approximated by the $\tilde{J}^{MPC}$.

The Fig. 4.5 shows the reliable approximate value function of the MPC control law in the region $\{0.8kmol/m^3 \leqslant x_1 \leqslant 1.1kmol/m^3, 383K \leqslant x_2 \leqslant 391K\}$. Some states with zero value function in this area only means that it is outside the reliable region, whose value function can not be inferred.

4.3.3 Control Result for ADP1

By the investigation of operational data, it is found that stabilizing the state in the region $\{0.6kmol/m^3 \leqslant x_1 \leqslant 1.1kmol/m^3, 385K \leqslant x_2 \leqslant 390K\}$ by MPC is very expensive. Therefore, 60 different initial states in these areas are sampled to test proposed NMPC and ADP hybrid controller, named ADP1. The optimal control action of ADP1 is determined by using Bellman optimality operator to the value function estimated from the MPC data. The holding time is set $\tau = 0.1$ min as well. The comparison of ADP1 and pure MPC is illustrated in Table 4.2.

Table 4.2: Comparison of ADP1 and MPC. The improvement for each test is $\frac{J^{MPC} - J^{ADP1}}{J^{MPC}}$

Controller	Unstable cases	Average Improvement
MPC	9	N/A
ADP1	9	4.58%

The Fig. 4.6 shows an example trajectory of ADP1 and MPC. In the early stages, the ADP1 employs almost same actions with the MPC, however, it diverges from the MPC trajectory in $t = 3$ and goes towards the ROA directly. Although the system eventually reaches the ROA under both types of controller, the ADP1 can drive system into the ROA much faster, thereby saving more cost. Fig. 4.7 illustrates all improvement ratio based on the stable scenarios in the 60 tests, where the negative value indicates that the ADP controller is inferior to the MPC in that case. It is also possible because the ADP controller focuses on the optimization of the expected performance, which may result in the loss for some special cases.

4.3.4 Control Result for ADP2

Since DP is featured by its policy improving procedure, one of raised questions is whether one can repeat the whole algorithm based upon the data from ADP1 to yield a better control law. In order to answer this question, the data from the 60 test set under the controller ADP1 is gathered and re-clustered in the state space, see Fig. 4.8. Since the initial states and their trajectories in the previous test are concentrated on the particular region, the number of nodes is smaller than ADP1. Totally 7 clusters are used to separate the state space and the 5^{th} node is full of unstable data, see Fig. 4.8. Also, the SVDD determines the reliable zone. The 3rd order polynomial approximator is used for estimating value function of each cluster.

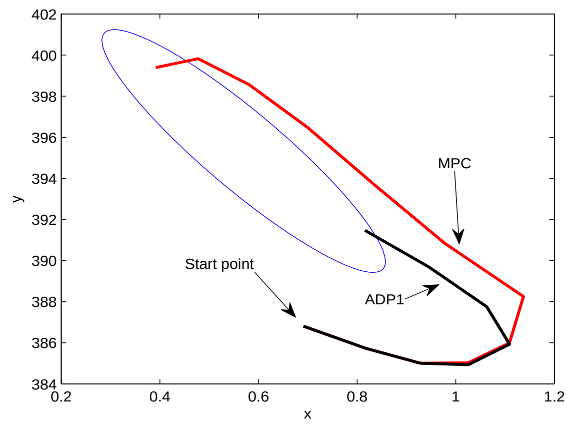

Figure 4.6: The trajectory of MPC and mixed ADP1

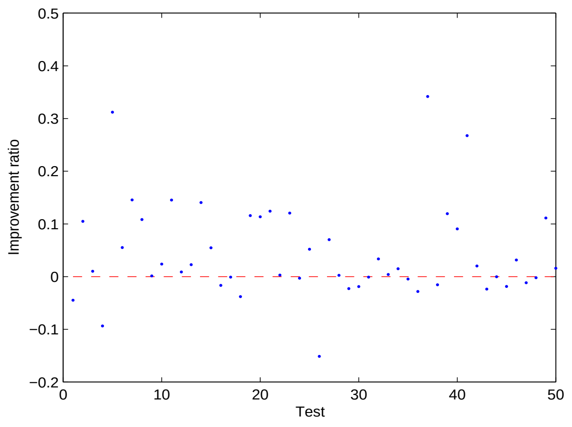

Figure 4.7: The improvement ratio of ADP1 compared with MPC

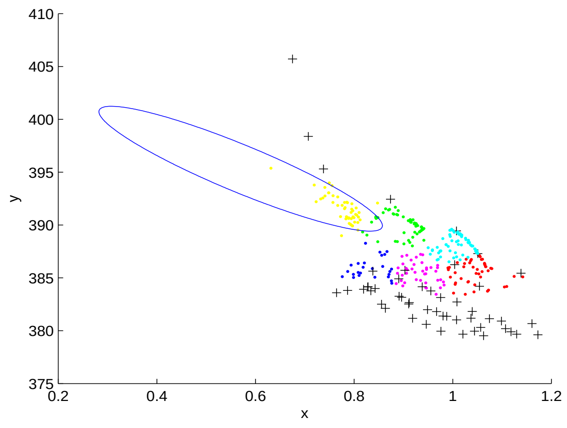

Figure 4.8: New cluster data and the ROA. The “+” represents data in the unstable trajectory

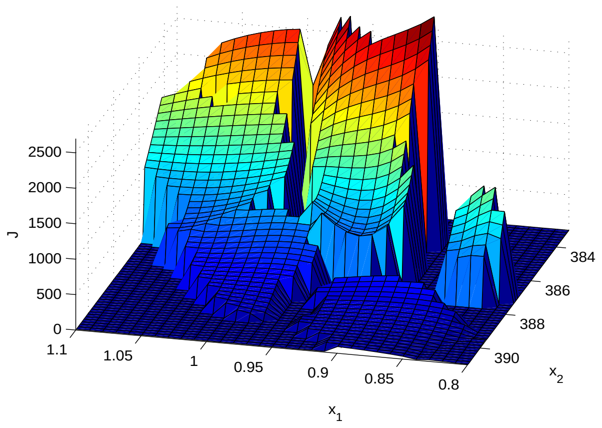

Figure 4.9: Approximate value function of the ADP1 control policy

The Fig. 4.9 shows the approximate value function of the ADP1 control law in the region $\{0.8kmol/m^3 \leqslant x_1 \leqslant 1.1kmol/m^3, 383K \leqslant x_2 \leqslant 391K\}$. Compare the Fig. 4.5 and Fig. 4.9, one can see that the approximate value function of ADP1 is higher than the MPC in part of state space. This is not surprising because the mixed strategy may drive the process into some unexperienced region under the NMPC policy, such as $\{0.9kmol/m^3 \leqslant x_1 \leqslant 0.95kmol/m^3, 384K \leqslant x_2 \leqslant 385K\}$, and these new sample data in the extreme region may pull up the whole estimated value function. Moreover, the function approximation scheme, highly depending on the sample data, cannot guarantee the monotonic decreasing of the value function for every state by policy improvement.

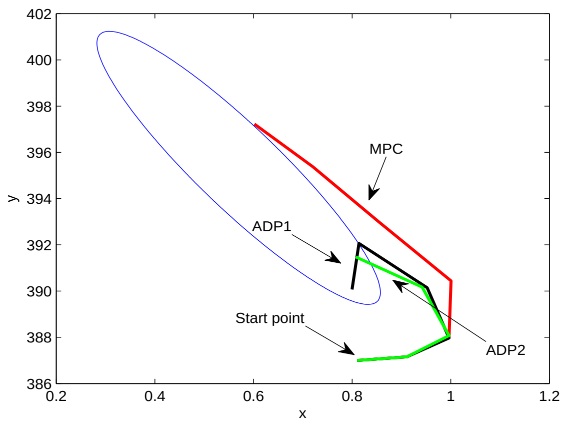

Figure 4.10: The trajectories of MPC, ADP1 and ADP2

Using the Bellman optimality operator again to improve the ADP1, the new control law ADP2 is yielded. A new set of scenarios, totally 60 different initial states with different disturbances, is tested for comparison of MPC, ADP1 and ADP2. The results are displayed in Table 4.3. It is shown that the ADP2 can further reduce the operational cost as well as keeping the similar unstable ratio with the MPC. Some tougher scenarios, such as initial states sampled far away from ROA, are tested in this set, thus leading to more unstable cases and ADP1's performance is degraded. Fig. 4.10 shows example trajectories for MPC, ADP1 and ADP2. One can see that the ADP2 makes better actions in some stages than ADP1. Figs. 4.11 and 4.12 illustrate the improving ratio based on the stable scenarios of these three control schemes.

Table 4.3: Comparison among MPC, ADP1 and ADP2

Controller	Unstable cases	Average Improvement
MPC	18	N/A
ADP1	20	2.67%
ADP2	17	6.03%

4.4 Concluding Remarks

In this chapter, we present an integrated control framework for the nonlinear system with uncertainty. An ADP and MPC mixed control strategy are employed to achieve robust stability and performance improvement. The CSTR example demonstrates the efficiency of the proposed control framework and compared with the conventional

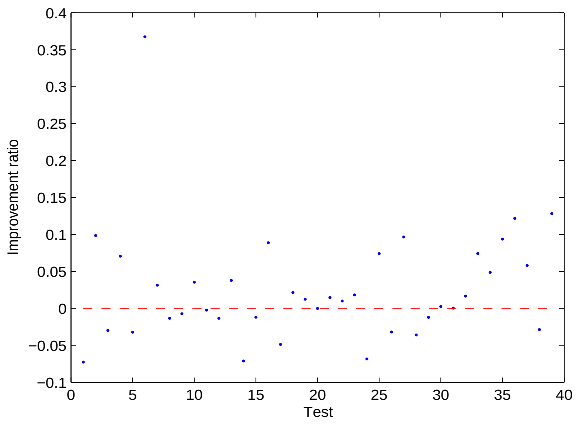

Figure 4.11: The improvement ratio of ADP1 compared with MPC (new test)

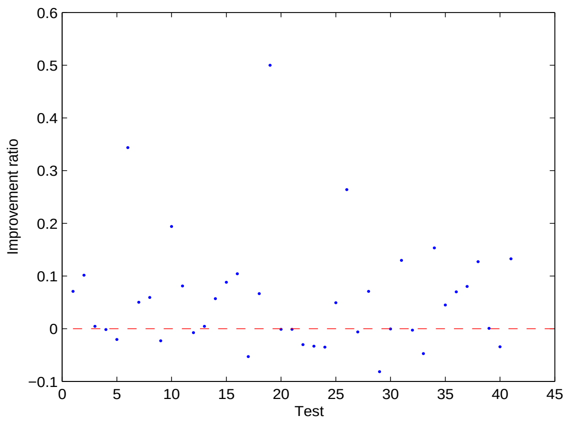

Figure 4.12: The improvement ratio of ADP2 compared with MPC

MPC. The future work includes the adaption of the approximate value function online and the extension to the general nonlinear system control.

Chapter 5

Probabilistic Modeling and Real Time Optimization for Plant-wide Operation

5.1 Background

With an ever increasing need for improving process efficiency and product quality in today's complex manufacturing environment, real-time optimization (RTO) is considered an essential technique for the operation of complex chemical plants. The role of RTO is to coordinate operating conditions (e.g., set points) of all units in a plant so that the plant remains near an economic optimum in the face of disturbances and other external/internal changes.

Steady-state model-based RTO has been the most prevalent technique due to its simplicity and suitability for online optimization [28, 79, 82]. However, steady-state RTO may leave significant room for performance improvement because it ignores the dynamic degrees of freedom such as unused capacity of a buffer tank and provides infeasible operating strategies during certain time periods. On the other hand, full-scale dynamic plant-wide models were used for rigorous online dynamic optimization based on a nonlinear programming method [13]. This approach may not be scalable to practical plant-wide optimization problems due to its excessive computational requirements. Alternatively, the idea of using linear model predictive control (MPC) formulation has been suggested [132]. However, this approach is restricted to relatively simple plants, such as an oil blending plant, where a linear

*A full version of this chapter has been published in the *Computers and Chemical Engineering* [130].

†An abbreviated version of this chapter was presented at 58th Canadian Chemical Engineering Conference (CSChE), 2008, Ottawa.

model is suitable to describe the plant and the settling time is relatively small. One common difficulty associated with the model-based methods is obtaining a reliable plant-wide dynamic model in the form of algebraic/differential equations, apart from its relevance to online optimization.

Another key issue of a model-based approach is that uncertainties from various sources can seriously degrade the performance. For example, model uncertainty can lead to RTO solutions that are far from the true plant optimum, or even an infeasible solution with respect to actual process constraints. Furthermore, once such solutions implemented and early symptoms go unnoticed, abnormal situations may arise and lead to significant periods of off-spec products or plant shutdowns. In most RTO systems, model uncertainty is addressed through some form of online model updating. This can be too late to contain the abnormal situations or prevent substantial loss of profitability. Several approaches have been proposed to handle uncertainties more systematically within RTO. Results analysis uses multivariable statistical hypothesis testing to decide if RTO predictions represent statistically valid changes with respect to the current plant operation [85]. This approach cannot explicitly address the effect of measurement uncertainty because it is a post-optimization evaluation method. Stochastic optimization-based approach was developed to incorporate uncertainty into the optimization formulation by using recourse functions and chance constraints [133]. Though this class of methods has been adopted for several applications [107, 74], uncertainties are incorporated into either steady state models or process constraints only. Recently, a measurement-based approach was developed that adjusts a model relating manipulated inputs to necessary conditions of optimality [56]. The procedure requires physical insight and experience about the process for successful implementation.

In general, there are two standard approaches to address the problem of multi-stage decision making under uncertainty. The first is multistage stochastic programming with recourse and the second is dynamic programming (DP). The stochastic programming usually recast the problem as a superstructure of large-scale deterministic equivalents corresponding to all the realizable scenarios [17]. Whereas this approach has been used to solve two-stage problems with relatively small number of scenarios, the computational complexity increases exponentially with the number of stages and the number of uncertain parameters, limiting its general appeal. DP offers a unified approach to solving problems of stochastic dynamic optimization [10]. Based on the principle of optimality [7], the DP algorithm decomposes a stochastic dynamic optimization problem into a sequence of single-period subproblems that are solved recursively backward in time. Central to the methodology is the “value” function, which is obtained via solving Bellman’s equation. The domain of the value function is the state space of the system to be controlled, and DP algorithms compute and store a table consisting of one value per state. Unfortunately, the size of a state space typically grows exponentially in the number of state variables. Known as the “curse of dimensionality”, this phenomenon renders DP intractable for the problems of practical scale.

Recently, the approach of approximate dynamic programming (ADP) has received much attention in the community of operations research [109] and has been introduced to process systems engineering field [69]. In most ADP methods, the value table is constructed only for a small subset of states and a function approximator is used to estimate values for unvisited states. The sample points are determined by running simulations with some known or random policies. The goal of this contribution is to develop a novel ADP-based RTO framework suitable for a large-scale plant under uncertainty. The proposed method constructs a probability transition matrix to describe plant-wide dynamics as a Markov Decision Process (MDP), which is a general mathematical formulation of multi-stage decision making problems under uncertainty. Hence, the approach is particularly relevant for a process where a plant-wide dynamic model, either it is empirical or first-principle, is difficult to obtain and uncertainty has a significant impact on profitability and reliability of the process operation.

This chapter is organized as follows. In Section 5.2, a brief review on self-organizing map is provided. Section 5.3 explains the proposed modeling approach, which consists of construction of Markov chain model using a self-organizing map (SOM). After establishing a discrete state model and its transition probability, further model refinement step identifying outliers is introduced. Section 5.3 also explains the proposed value function-based optimization framework where a failure node is designed to strike a balance between profitability maximization and risk management. Section 5.4 illustrates comparison studies of different RTO strategies applied to a reactor-storage-separator system connected via a material recycle loop and Tennessee Eastman challenge problem. Finally, Section 5.5 provides the conclusions of this study.

5.2 Preliminary

5.2.1 Self-organizing Map

Self-organizing map (SOM) is a unsupervised learning approach developed by Kohonen for nonlinear dimension reduction, data clustering and visualizing without any prior knowledge. In this thesis, we mainly focus on the application of SOM for high dimensional data clustering. Since the SOM usually consists in several neurons in two dimension grid, each neuron can be viewed as the cluster for the data. The whole structure is a two-layer neural network: input layer and output layer. Each neuron in these layers has two important values: the position in the grid and the weight in the data space. In the SOM process, there are three phases: competition, cooperation and synaptic adaption.

In the competitive stage, each node is given an initial weight randomly. Then, for the data set, each data is assigned to the node with minimal distance (usually Euclidean metric) between the data vector and weight of the node. This node is called the best matching unit (BMU).

In the cooperative phase, the objective is to determine the correlation among each node. In this part, we define the neighborhood function $h_{j,i}$ for node i and j:

$$h_{j,i} = \exp \frac{-d_{i,j}^2}{2\sigma^2(t)} \tag{5.1}$$

$$\sigma^2(t) = \sigma_0^2 \exp \frac{-t}{\tau} \tag{5.2}$$

$$d_{i,j} = ||r_i - r_j|| \tag{5.3}$$

where t is the number of iterations; τ is the time constant to control decay rate. r_i is the position of cluster i in the discrete lattice.

In the final stage, the weight of each node is updated by the input data. The data x is first assigned to the node and then not only this node, but also the neighboring nodes' weight are adapted by this input vector according to the following formulation:

$$w_j(t+1) = w_j(t) + \eta(t) h_{j,i(x)}(x - w_j(t)) \tag{5.4}$$

where $w_j(t)$ is the weight of node j at iteration t; $\eta(t)$ is the learning rate at iteration t; $i(x)$ is the BMU for the data x. According to this way, the node can span across the whole space and be adjusted by the data. As the increase of the index t, the neighborhood function decreases exponentially, which guarantees the convergence of the algorithm.

5.3 Proposed RTO Framework

5.3.1 Construction of Markov Chain Model

The first step in the proposed approach is to model underlying behavior of plant-wide dynamics under uncertainties as a series of “representative” discrete states evolving over time without any predetermined equations. The underlying idea is similar to *cell mapping* [48, 46], which approximates system’s dynamics as transitions between discrete points. In order to characterize such representative states, we cluster process data using SOM [61]. SOM classifies data set into different groups such that each observation is assigned to a single class and the observations within a class are more similar, in some metric, to each other. Though such classification can be achieved by other clustering methods including K-means, fuzzy C-means, etc., there are some advantages in using SOM. SOM creates an ordered set of states in one or more dimensions. This offers topological structure of data, which means data points that are closer to each other in high dimensional space will be projected to the neighboring clusters (or nodes) in low dimensional space. This property can simplify the procedure of searching for neighboring data points in the proposed framework, which will be discussed later. Moreover, the clustered and ordered data sets are more interpretable than those clustered points in other clustering methods. A successful application of SOM with cell mapping for extracting transition dynamics of a film deposition process has also been reported [89].

By assuming all states of the plant can be observed, the historical data set is divided into representative nodes, the probability transition matrix P and one step reward are computed. It is also assumed that each unit can take a finite number of set-point values. Under a particular action vector u (i.e., a particular set points), the transition probability from a discrete node i to j is estimated as

$$P^u_{ij} = \frac{N^u_{ij}}{\sum_{k=1}^{n_T} N^u_{ik}} \tag{5.5}$$

where N^u_{ij} is the number of switches from the states belonging to the node i to those belonging to j under action u and n_T is the total number of discrete nodes.

The one step reward R^u_{ij} denotes the reward incurred during the transition from i to j under action u. The expected reward over the entire data points moving from node i to node j under action u is computed as follows:

$$R^u_{ij} = \frac{\sum_{x_{ij}} r(x_{ij}, u)}{M} \tag{5.6}$$

where x is the state point that made a transition from node i to j under action u and M is the total number of transitions.

To check the model, the whole historical data can be divided into the training part and validation part. The derived transition probability is used to predict the successive node in the validation data set to check its accuracy.

A special node, F, is also designed to represent "failure states" such as shutdown, reaction termination, constraint violation, etc. As system reaches these states, operation stops and needs restarting. Hence, this node actually behaves like an *absorbing* state meaning that the system stays at the same state with probability of one. Moreover, a negative value should be assigned to this node due to the loss of economic performance. In a later section, we will show that this additional design assists the ADP scheme in providing a risk-averse policy.

Once the representative states and their transition probabilities are obtained, the optimal value for each cluster is computed by iteratively solving the following Bellman equation:

$$V^*(i) = \max_u E\left\{R_{ij}(i,\ u) + \gamma V^*(j)\right\} \tag{5.7}$$

where $i,\ j\ \in\ \{1,\ 2,\ \ldots,\ n_T\}$ are the nodes obtained from SOM and j is a successor node of i.

5.3.2 Model Refinement

Treating Outliers

Though a probabilistic model and its associated value function can be derived from historical data by the simple steps described in the previous section, the main premise was that the data points belonging to a same node will show a similar performance. However, this assumption may not hold for systems with complex dynamics. Furthermore, assigning data points to a certain cluster near the boundaries is non-trivial and may affect the model accuracy.

Since the goal of RTO is to improve plant-wide performance, the process state characterization should be linked with some performance index. For example, a certain process state associated with process shutdown should not be classified into the same group with a process state under normal operations simply because those two process states are "close" in terms of Euclidean distance metric. In the proposed method, the value under a certain policy (i.e., Q value) associated with a process state serves as a metric to determine the similarity among the process states.

In order to identify a group of data points with distinctive values from the majority of data points inside each node, Q value of each process state (x_t) and action (u_t) pair is first estimated using

$$Q(x_t, u_t) = r(x_t, u_t) + \gamma V^*(j) \tag{5.8}$$

where j is a cluster containing x_{t+1} and $V^*(j)$ is the value obtained by solving Eq. (5.7). Depending on the transition probability and single-stage reward r, each data point inside the same node can take a different Q value. Inside each cluster, subsets of data under the same actions (set points) are examined to detect the "outliers." The outliers are defined as the data points with Q values falling outside the following range [123]:

$$[Q_1 - 1.5 \cdot \mathrm{IQR},\ Q_3 + 1.5 \cdot \mathrm{IQR}] \tag{5.9}$$

where Q_1 and Q_3 denote the first and the third quartiles, respectively. IQR is the interquartile range defined as

$$\mathrm{IQR} = Q_3 - Q_1 \tag{5.10}$$

Since the outliers based on Q values reveal the "uniformity" of data points in each cluster, the trade-off between the numbers of nodes and outliers is used in refining the SOM structure including the number of clusters and grid size. In order to keep the generalization capability, a minimum number of data points is specified in each node first, and the number of outliers is minimized in determining the optimal structure of SOM.

As the outliers are detected successively in each node, simple hyper-spheres are constructed to represent the area of outliers for fast classification in real-time applications. The center of each hyper-sphere is the current outlier, and the radius is the distance between the outlier and its nearest "normal" data point. Any data point falling into the sphere is treated as the outlier of each cluster as seen in Fig. 5.1. It should be also noted that the closest normal data point can be found in a different cluster when the outlier lies near the boundary. This can render the hyper-sphere construction step computationally expensive. However, the topology preservation of SOM can reduce the computation dramatically because only the data points in neighboring nodes can be examined. For example, if the clusters are arranged in a 6×10 grid, any node has six neighbors at most as seen in Fig. 5.2. This can save about 75% of computation compared to the exhaustive search if the number of data points is the same for all the nodes.

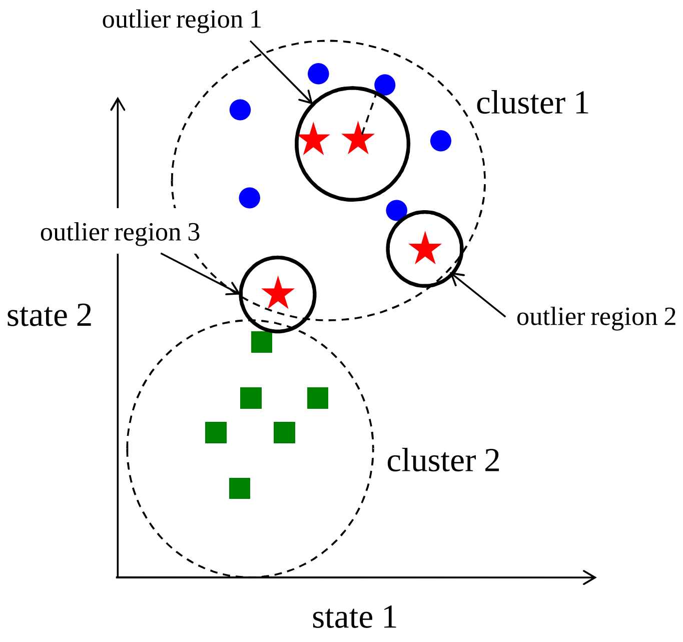

Figure 5.1: Domain of outliers: stars denote outliers and the other points are normal data.

Update of Q Values for Outliers

Since the probabilistic model is not valid for outliers any more, a model-free approach is employed to estimate a new value for each outlier. Once the radius and centroid of hyper-sphere is defined, the data points belonging to the area are counted and their action-values $Q(x, u)$ are computed by the following model-free update scheme:

$$Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \alpha \{r(x_t, u_t) + \gamma Q(x_{t+1}, u_{t+1}) - Q(x_t, u_t)\} \qquad (5.11)$$

where $\alpha \in [0, 1)$ is a step-size parameter. Eq. (5.11) is referred to as SARSA [102] mentioned in Chapter 2. It updates Q values from a state trajectory in the absence of a model. Then the average of Q values of all outliers in the region represents the Q value of the outlier region in a cluster.

The model refinement step can be performed both off-line (i.e., using the historical data only) and online (i.e., when the optimizer is put to work). The

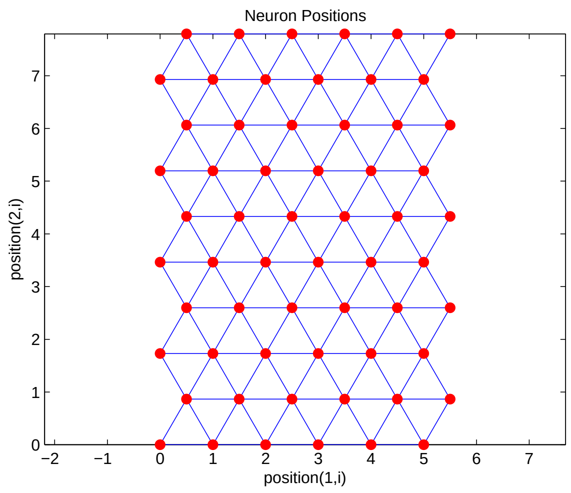

Figure 5.2: Cluster positions of SOM in 2-D (6×10) grid. SOM projects closer data points in higher dimension onto neighboring nodes.

transition matrix, optimal value for each node, and action-values for outliers are updated online as a new observation is available at each time step. The following steps are taken:

- If a new observation belongs to an outlier region, update its Q function by Eq. (5.8).
- If a new observation belongs to a node, Eq. (5.11) is used to update the Q value and determine if the new data point should be classified as an outlier. If it is an outlier, create a new outlier region. Otherwise, update the transition matrix, one-step reward, and the corresponding value function after a predefined number of operation episodes are completed.

Remark: Although conventional DP can handle millions of state points [10], the "curse of dimensionality" still defies its applications to practical problems with continuous state space. Unlike simple discretization of the continuous state space, the proposed scheme dramatically reduces the number of state points and alleviate the "curse of dimensionality" by representing the continuous state space as discrete nodes. Moreover, the model-free scheme for handling sparse data regions is not limited by the state dimension. However, the number of nodes and storage requirement of historical data will increase with the data dimension. In such a case, data compression techniques such as principal component analysis (PCA) will be useful as shown in [89]

5.3.3 Online Implementation

General Procedure

With the learned value ($\tilde{V}^*$) and action-value functions ($\tilde{Q}^*$), the online decision is implemented as follows:

1. Given a process state at current time (x_t), find its Best Matching Unit (BMU) using the trained SOM.

2. Compute the distance to the center of each outlier inside the BMU.

3. Compare the distance with corresponding radius.

4. If the data falls into an outlier domain, calculate the action (set point of each unit) according to
$$u_t^* = \arg\max_{u_t} \tilde{Q}^*(x_t, u_t) \tag{5.12}$$
Since the number of data points in outlier domains may be small in the beginning, the sample-based model-free learning may still require further information for better decision making. For further information gathering in the outlier regions, ϵ-greedy method is employed [115]. When the system stays in an outlier domain, the probability of performing exploration is set as a small number ϵ. If we determine to explore, a random action will be selected among the candidates with an equal probability for each. This strategy is useful for relatively small action space. With a large number of action candidates or for risk-sensitive processes, a small action set with high profits and low risks should be selected instead.

5. If the data is classified as a normal data in the BMU, calculate the action according to
$$u_t^* = \arg\max_{u_t} E\left\{r(x_t, u_t) + \gamma\tilde{V}^*(j)\right\} \tag{5.13}$$
where j is a cluster to which x_{t+1} belongs. At the next sample time calculate the Q value of (x_t, u_t^*) by Eq. (5.11) and determine if it is an outlier.

Effect of the Failure-node on Policy Selection

In chemical process optimization, it is more desirable to prevent abnormal situations rather than simply maximize the averaged total reward. Only a few in the literature discuss the risk-sensitive decision making within the setting of dynamic optimization.

[40] employs the error state to denote risky cases and the most profitable policy in the set of policies whose probability of entering undesirable states less than a preset value. However, the weighting factors for the risk and reward terms should be adapted online in this approach. This trial-and-error approach is not acceptable in chemical process operations.

As discussed in Section 5.3.1, our proposed method designs a special node, F, to represent failure states. Since DP computes a greedy policy with respect to total reward, a negative value V_F assigned to the node F will derive a policy striking a balance between total reward and the risk of entering undesirable states. This property can prevent the process states violating some important constraints or more important, avoiding the instability. However, one cannot arbitrarily assign a large negative value to V_F because it may lead to a very conservative policy. In this section, a systematic guideline on how to assign V_F is provided.

First, a quantified risk term under a policy π is defined as the discounted sum of probability for entering F:

$$W^{\pi} = \sum_{k=1}^{\infty} P_k \gamma^k \tag{5.14}$$

where γ is the discount factor; P_k is the probability of reaching F in k steps.

Then, the value of any state under policy π is given as

$$V^{\pi} = \sum_{k=1}^{\infty} P_k \gamma^k V_F + \sum_{k=1}^{\infty} \gamma^{k-1} \left(1 - \sum_{\ell=1}^{k} P_\ell\right) \hat{r}_k \tag{5.15}$$

where $\hat{r}_k$ is the expected reward in k^{th} step, on the condition that process doesn't enter F.

For simplicity in the proof of Theorem 18, the average reward R^{π} is also defined as

$$R^{\pi} = \frac{\sum_{k=1}^{\infty} \gamma^{k-1} \left(1 - \sum_{\ell=1}^{k} P_\ell\right) \hat{r}_k}{\sum_{k=1}^{\infty} \gamma^{k-1} \left(1 - \sum_{\ell=1}^{k} P_\ell\right)} \tag{5.16}$$

R^{π} also satisfies

$$\underline{r} \leq R^{\pi} \leq \overline{r} \tag{5.17}$$

where $\underline{r}$ and $\overline{r}$ are lower and upper bounds of single-stage reward, respectively.

Given V_F, the proposed method strikes a balance between the reward and the risk. Without loss of generality, we consider the case where the optimizer compares two different policies, π with a higher risk and κ with a lower risk. Two difference quantities are defined as

$$\delta = W^{\pi} - W^{\kappa} > 0 \tag{5.18}$$

$$\beta = R^{\pi} - R^{\kappa} \tag{5.19}$$

where δ is the difference in the risk terms of two policies; β is the difference between their average rewards. From Eq. (5.17), the maximum of β is $\bar{r} - \underline{r}$.

We first derive the upper bound of the W under a control policy,

Lemma 17.

$$W^{\pi} = \sum_{k=1}^{\infty} P_k \gamma^k < \gamma$$

Proof. Define the probability of reaching failure node at i^{th} sample time as $P(i)$. Since P_k is the probability of entering the failure node in k sample times,

$$P_k = P(k) \prod_{m=1}^{k-1} (1 - P(m)), \quad k \geq 2 \tag{5.20}$$

Then a finite sum of n terms is considered as follows:

$$\begin{aligned}
\sum_{k=1}^{n} P_k \gamma^k &= \sum_{k=1}^{n-1} P_k \gamma^k + P_n \gamma^n \\
&= \sum_{k=1}^{n-1} P_k \gamma^k + \prod_{m=1}^{n-1} (1 - P(m)) P(n) \gamma^n \\
&\leqslant \sum_{k=1}^{n-1} P_k \gamma^k + \prod_{m=1}^{n-1} (1 - P(m)) \gamma^n \qquad (5.21) \\
&= \sum_{k=1}^{n-2} P_k \gamma^k + \prod_{m=1}^{n-2} (1 - P(m)) P(n-1) \gamma^{n-1} + \prod_{m=1}^{n-1} (1 - P(m)) \gamma^n \\
&< \sum_{k=1}^{n-2} P_k \gamma^k + \prod_{m=1}^{n-2} (1 - P(m)) P(n-1) \gamma^{n-1} + \prod_{m=1}^{n-2} (1 - P(m))(1 - P(n-1)) \gamma^{n-1} \\
&= \sum_{k=1}^{n-2} P_k \gamma^k + \prod_{m=1}^{n-2} (1 - P(m)) \gamma^{n-1} \qquad (5.22)
\end{aligned}$$

Comparing Eqs. (5.21) and (5.22) and repeating the same procedure yield

$$\sum_{k=1}^{n} P_k \gamma^k < P_1 \gamma + (1 - P_1) \gamma^2 < \gamma$$

Letting $n \to \infty$ proves Lemma 17. $\square$

With the above, the optimizer selects a better policy between π and κ as follows:

Theorem 18. *If $\beta < 0$, the optimizer will select the policy κ. If $\beta > 0$ and δ satisfies*

$$\delta > \frac{\gamma - W^{\pi}}{D} \tag{5.23}$$

where D is a user-specified parameter, the optimizer will select the policy κ given the following value function of the failure node:

$$V_F \leq \frac{(D+1)\underline{r} - D\bar{r}}{\gamma(1-\gamma)} \tag{5.24}$$

Proof. In order to prove Theorem 18, Eq. (5.15) is rewritten by exchanging the summation indices and using notation W:

$$\begin{aligned}
V^{\pi}(s) &= W^{\pi}V_F + \left\{\frac{1}{1-\gamma} - \sum_{k=1}^{\infty}\sum_{\ell=1}^{k}\gamma^{k-1}P_{\ell}\right\}R^{\pi} \\
&= W^{\pi}V_F + \left\{\frac{1}{1-\gamma} - \sum_{\ell=1}^{\infty}\sum_{k=\ell}^{\infty}\gamma^{k-1}P_{\ell}\right\}R^{\pi} \\
&= W^{\pi}V_F + \left\{\frac{1}{1-\gamma} - \frac{\sum_{\ell=1}^{\infty}\gamma^{\ell}P_{\ell}}{\gamma(1-\gamma)}\right\}R^{\pi} \\
&= W^{\pi}V_F + \left\{\frac{1}{1-\gamma} - \frac{W^{\pi}}{\gamma(1-\gamma)}\right\}R^{\pi}
\end{aligned} \tag{5.25}$$

Similarly, the value function under policy κ is given by

$$V^{\kappa}(s) = W^{\kappa}V_F + \left\{\frac{1}{1-\gamma} - \frac{W^{\kappa}}{\gamma(1-\gamma)}\right\}R^{\kappa} \tag{5.26}$$

Using δ and β in Eqs. (5.18) and (5.19), Eq. (5.25) can be written as

$$V^{\pi}(s) = (W^{\kappa}+\delta)V_F + \left\{\frac{1}{1-\gamma} - \frac{W^{\kappa}+\delta}{\gamma(1-\gamma)}\right\}(R^{\kappa}+\beta) \tag{5.27}$$

Eq. (5.27) - Eq. (5.26) gives

$$\begin{aligned}
V^{\pi}(s) - V^{\kappa}(s) &= \delta V_F + \frac{\beta}{1-\gamma} - \frac{(W^{\kappa}+\delta)\beta + \delta R^{\kappa}}{\gamma(1-\gamma)} \\
&= \delta V_F + \frac{\beta}{1-\gamma} - \frac{W^{\pi}\beta + \delta R^{\kappa}}{\gamma(1-\gamma)}
\end{aligned} \tag{5.28}$$

Eq. (5.28) shows that the selection of a policy relies on V_F, δ, β, and W. Each case is examined in the followings:

- $\beta < 0$: From Lemma 17, we have

$$\frac{\beta}{1-\gamma} - \frac{W^{\pi}\beta}{\gamma(1-\gamma)} < 0 \tag{5.29}$$

Considering Eq. (5.24), we also have

$$\delta V_F - \frac{\delta R^\kappa}{\gamma(1-\gamma)} < 0 \tag{5.30}$$

Substitution of Eqs. (5.29) and (5.30) into Eq. (5.28) yields $V^\pi(s) - V^\kappa(s) < 0$ and thus the policy κ is selected.

- $\beta > 0$: This is a non-trivial case because the policy π has a higher probability of leading the process to the undesirable state while the expected reward is also higher than that of κ.

  From Eq. (5.28), we have

$$\begin{aligned} V^\pi(s) - V^\kappa(s) &= \delta \left\{ V_F + \frac{\beta}{\delta(1-\gamma)} - \frac{W^\pi \beta + \delta R^\kappa}{\delta\gamma(1-\gamma)} \right\} \\ &= \delta \left\{ V_F - \left[\frac{R^\kappa}{\gamma(1-\gamma)} - \left(\frac{1}{1-\gamma} - \frac{W^\pi}{\gamma(1-\gamma)} \right) \frac{\beta}{\delta} \right] \right\} \end{aligned} \tag{5.31}$$

  Let us consider the minimum achievable value of $\frac{R^\kappa}{\gamma(1-\gamma)} - \left(\frac{1}{1-\gamma} - \frac{W^\pi}{\gamma(1-\gamma)}\right)\frac{\beta}{\delta}$ for any state. From Eq. (5.23), let δ approach the lower bound $\frac{\gamma - W^\pi}{D}$ which is always positive showed by Lemma 17. As for β, let $\beta = \bar{r} - \underline{r}$ from Eqs. (5.17) and (5.19). Moreover, let $R^\kappa = \underline{r}$. The minimum is achieved as

$$\min \left[\frac{R^\kappa}{\gamma(1-\gamma)} - \left(\frac{1}{1-\gamma} - \frac{W^\pi}{\gamma(1-\gamma)} \right) \frac{\beta}{\delta} \right] = \frac{(D+1)\underline{r} - D\bar{r}}{\gamma(1-\gamma)} \tag{5.32}$$

  Given the range of V_F in **Theorem 17**, it always satisfies $V^\pi(s) - V^\kappa(s) < 0$. Thus, optimizer will select the policy κ.

□

The implication of Theorem 18 is that when the policy with a higher risk, π, has a lower profit, the less risky and more profitable policy κ is selected. When π has a higher profit compared to κ, V_F is selected at the upper bound of Eq. (5.24) with the user-defined value of D. In this case, the less risky policy κ is selected if the risk difference δ is greater than the right hand side of Eq. (5.23).

D is a scaling factor for the difference between the discount factor and the risk. The larger D implies a lower threshold, which chooses a safer policy. If one has the information on the bounds of the risk term, as in the chance constraint of stochastic programming [40, 133], the lower bound of D can be given as

$$D > \frac{\gamma - W_{\max}}{W_{\max} - W_{\min}} \tag{5.33}$$

where $W_{\min} \leq W^{\mu} \leq W_{\max}$ and μ is any feasible policy. Choosing D satisfying Eq. (5.33) guarantees that any policy π with a higher risk than $W_{\max}$ will not be selected. For example, consider a policy π that satisfies $W^{\pi} > W_{\max}$. Then, we have

$$\delta = W^{\pi} - W_{\min} \geq W_{\max} - W_{\min} \geq \frac{(\gamma - W^{\pi})(W_{\max} - W_{\min})}{\gamma - W_{\max}} \geq \frac{\gamma - W^{\pi}}{D} \tag{5.34}$$

$W_{\max}$ can be specified easily by a user because it reflects the conservativeness of operation. $W_{\min}$ can be simply computed offline by the conventional DP with the stage-wise reward of:

$$r(x_t, u_t) = \begin{cases} -1 & \text{if } x_{t+1} \text{ belongs to the failure node.} \\ 0 & \text{otherwise} \end{cases} \tag{5.35}$$

and

$$W_{\min} = - \max_{u_0,\, u_1,\, \ldots} E\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \middle| \, x_0 \right] \tag{5.36}$$

Detailed procedure of computing $W_{\min}$ using DP can be found in [40].

5.4 Case Study

In this section two examples are provided to illustrate the efficacy of the proposed framework. The first example considers an integrated process studied in [121] that involves a CSTR, a storage tank, and a flash tank with a material recycle stream. Then, the Tennessee Eastman challenge process [31] is tested in the second example.

5.4.1 Reaction-storage-separation Network

Problem Description

An integrated process composed of a CSTR, a storage tank, and a flash tank with a material recycle stream is shown in Fig. 5.3. A fresh feed stream F_0 consisting of pure component 1 is fed to the reactor, where two irreversible reactions $1 \xrightarrow{k_1} 2 \xrightarrow{k_2} 3$ take place to produce the desired product 2 and undesired product 3. The reactor product stream F_R enters the storage tank, of which the downstream flow enters the flash tank. The most volatile component 1 goes up to the overhead and is condensed into liquid returning to reactor via the recycle loop. The component 3 is assumed nonvolatile. The material balance is described by the following 12 equations:

$$\dot{H}_R = \frac{1}{\rho A_R}(F_0 + D - FR)$$

$$
\begin{aligned}
\dot{x}_{1R} &= \frac{F_0(x_{10} - x_{1R}) + D(x_{1D} - x_{1R})}{\rho A_R H_R} - k_1 x_{1R} \\
\dot{x}_{2R} &= \frac{-F_0 x_{2R} + D(x_{2D} - x_{2R})}{\rho A_R H_R} + k_1 x_{1R} - k_2 x_{2R} \\
\dot{x}_{3R} &= \frac{-(F_0 + D)x_{3R}}{\rho A_R H_R} + k_2 x_{2R} \\
\dot{H}_M &= \frac{1}{\rho A_M}(F_R - F_M) \\
\dot{x}_{1M} &= \frac{F_R}{\rho A_M H_M}(x_{1R} - x_{1M}) \\
\dot{x}_{2M} &= \frac{F_R}{\rho A_M H_M}(x_{2R} - x_{2M}) \\
\dot{x}_{3M} &= \frac{F_R}{\rho A_M H_M}(x_{3R} - x_{3M}) \\
\dot{H}_B &= \frac{1}{\rho A_B}(F_M - B - D) \\
\dot{x}_{1B} &= \frac{1}{\rho A_B H_B}[F_M(x_{1M} - x_{1B}) - D(x_{1D} - x_{1B})] \\
\dot{x}_{2B} &= \frac{1}{\rho A_B H_B}[F_M(x_{2M} - x_{2B}) - D(x_{2D} - x_{2B})] \\
\dot{x}_{3B} &= \frac{1}{\rho A_B H_B}[F_M(x_{3M} - x_{3B}) + D x_{3B}]
\end{aligned}
$$

where H_R, H_M, and H_B represent liquid levels of reactor, storage tank, and the flash tank respectively. x_{ij} represents the molar liquid fraction of each component i in stream j. In the recycle flow the following constant relative volatility expression is used for equilibrium relationship:

$$
\begin{aligned}
y_{1D} &= \frac{x_{1D}}{x_{2D} + x_{1D}} \\
y_{2D} &= \frac{x_{2D}}{x_{2D} + x_{1D}} \\
x_{1D} &= \frac{\alpha_A y_{1D}}{1 + (\alpha_A - 1)y_{1D} + (\alpha_B - 1)y_{2D}} \\
x_{2D} &= \frac{\alpha_B y_{2D}}{1 + (\alpha_A - 1)y_{1D} + (\alpha_B - 1)y_{2D}}
\end{aligned}
$$

Tables 5.1 and 5.2 show operation ranges and nominal parameter values, and Table 5.3 shows the operation constraints. Though high yields can be theoretically achieved by maintaining a high ratio of the recycle flow to the fresh feed flow, high recycle-to-feed ratio can induce a large variation in the process even with a small change in the feed stream. This “snowball effect” makes the plant-wide optimization challenging.

A model predictive controller (MPC) is designed to stabilize the liquid level of each unit while keeping the component ratio of each production within the operation

Table 5.1: Operation ranges of reaction-storage-separation network

Variables	Range
F_R	[8,47]
F_M	[8,47]
B	[0.67,3]
F_0	1.66
D	[8,45]

Table 5.2: Nominal parameters of reaction-storage-separation network model

Parameters	Value		
Liquid density	$\rho = 1$		
Rate constant	$k_1 = 0.0167$	$k_2 = 0.0167$	
Vessel area	$A_R = 5$	$A_M = 10$	$A_B = 5$
Relative volatilities of A	$\alpha_A = 90$		
Relative volatilities of B	$\alpha_B = 1$		

Table 5.3: Operation constraints of reaction-storage-separation network

Variables	Lower bound	Upper bound
x_{2R}	0	0.3
x_{3R}	0	0.025
x_{1B}	0	0.5
x_{3B}	0.5	1

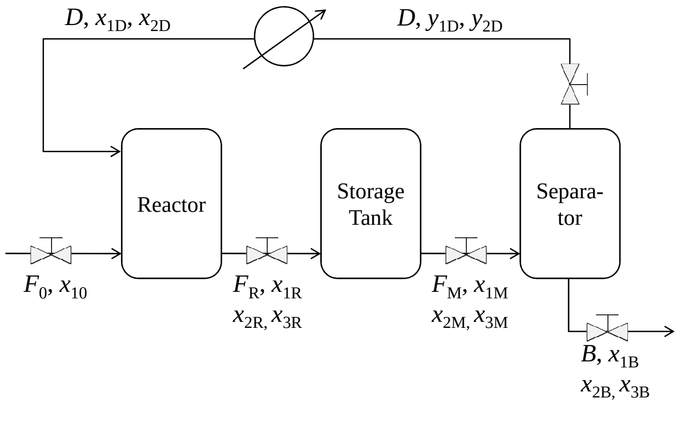
D, x_{1D}, x_{2D}
D, y_{1D}, y_{2D}
Reactor
Storage Tank
Separa-tor
F_0, x_{10}
F_R, x_{1R} x_{2R}, x_{3R}
F_M, x_{1M} x_{2M}, x_{3M}
B, x_{1B} x_{2B}, x_{3B}

Figure 5.3: Schematic of the reaction-storage-separation network.

range. A successive linearization-based MPC is used where the nonlinear model is linearized at each time and the resulting quadratic programming (QP) is solved. In order to find a feasible solution, the operation range is treated as soft constraints and the violation of each constraint, i.e. ε, is penalized:

$$\min_{u} \sum_{i=1}^{h} (H_{sp} - H_i)^T (H_{sp} - H_i) + 10^3 \varepsilon^T \varepsilon \tag{5.37}$$

subject to

$$\Delta u_{min} < \Delta u < \Delta u_{max} \tag{5.38}$$

$$u_{min} < u < u_{max} \tag{5.39}$$

$$H_{min} < H < H_{max} \tag{5.40}$$

where h, u and H denote the prediction horizon, flow rate and liquid level respectively. The detailed design of MPC is shown in Table 5.4.

Table 5.4: Design of MPC for level control in reaction-storage-separation network

Controlled variables	H_R, H_M, H_B
Manipulated variables	F_R, F_M, D, B
Sample time	5 min
Predictive horizon	8
Control horizon	6
max Δu	F_R: 3, F_M : 3, D: 3, B: 0.3
min Δu	F_R: -3, F_M: -3, D: -3, B: -0.3

The objective of plant-wide optimization is to maximize the production of desired product formulated as

$$\max_{H_{sp,i}} \sum_{i=0}^{\infty} 0.9^i \left(\sum_{t=60(i-1)+1}^{60i} B(t) x_{2B}(t) \right) \tag{5.41}$$

The decision variables are the level set points in each unit. $i \in \{0, 1, 2, \cdots\}$ is the time index where the set point is computed. The time interval of dynamic optimization is set as 60 minutes as suggested in [121]. The set point space is discretized by $4 \times 4 \times 4$ grids for simplicity. For each unit, the level set point can be 10, 15, 20, and 25. It is noted that the proposed approach can also handle a large action space as opposed to conventional multi-stage dynamic optimization methods (e.g., MPC) thanks to the single-stage optimization formulation.

Collecting Historical Data

In order to show performance improving capability of the proposed scheme, closed-loop data sets under the conventional steady state RTO and a linearized model-based dynamic RTO proposed in [121] were collected separately. In each episode of simulation, the fresh feed flow (F_0), reaction constants (k_1 and k_2), and relative volatility (α_A and α_B) were altered between lower and upper bounds randomly at every 50–200 minutes. Figure 5.4 shows one sample realization of F_0. The lower and upper bounds of F_0, k, and α were [1.46 1.86], [0.012 0.021], and [80 90], respectively.

The total simulation runs for steady state RTO and the dynamic RTO were 50 and 440 respectively, with each episode running for 1200 minutes. The dynamic RTO solved the following optimization problem and computed the corresponding set point trajectory every 60 minutes:

$$\max_{u_i} \sum_{i=1}^{20} \sum_{t=60(i-1)+1}^{60i} B(t)x_{2B}(t) \tag{5.42}$$

subject to

$$\Delta u_{\min} < \Delta u < \Delta u_{\max} \tag{5.43}$$

$$u_{\min} < u < u_{\max} \tag{5.44}$$

$$H_{\min} < H < H_{\max} \tag{5.45}$$

The prediction and control horizons were set as 20 and 10 respectively. For fair comparison, a set point closest to the one computed by the steady state RTO or dynamic RTO was implemented.

Markov Chain and Value Function Learning

In learning SOM, the grid structure 6×10 was selected based on the outlier criterion with the minimum number of 10 data points in each cluster. The number of outliers was 5.7% of the total number of data points. The single-stage reward and the transition matrices were calculated as described in Section 5.3.1. A special node was also added to represent failure states, where the storage tank is totally drained. Thus, the final transition matrix for each group of data has the size of 60×61.

To specify the value, V_F, for the failure node, the upper ($\bar{r}$) and lower ($\underline{r}$) bounds of single-stage reward were estimated from the data as 132 and 28.6 respectively.

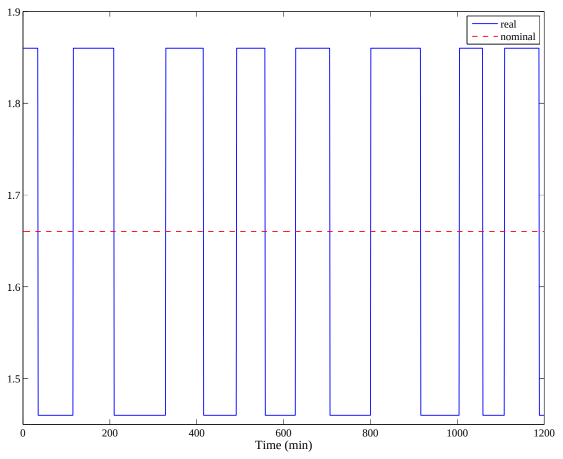

Figure 5.4: A sample realization of uncertainties in the fresh feed flow rate of reaction-storage-separation network.

In order to specify D, with γ set as 0.9, $W_{\min}$ was computed as 0, and $W_{\max}$ was set as 0.25. Then D should be greater than 2.6 by Eq. (5.33). Given a possible plant-model mismatch, the lower bound of risk may be greater than 0. Thus, D was specified as 4. Finally, V_F was calculated as -4277 by Eq. (5.24). All parameters for proposed RTO are shown in Table 5.7.

Simulation Result

Although the soft-constraint MPC can balance the set point tracking and constraint violation, inappropriate set points lead to frequent constraint violations. Whereas both steady-state and dynamic RTO's leave the risk management to MPC, the proposed scheme reduces the frequency of constraint violations and increases the expected rewards at the RTO level. Hence, the ratio of constraint violation and the total reward accumulated over 1200 minutes are used as comparison criteria.

For online testing of three different RTO schemes, a hundred fresh disturbance scenarios were used. The test results are shown in Table 5.5. Compared to the dynamic and steady-state RTO's, the proposed method reduced the number of constraint violations and obtained a comparable level of productivity. Though the production under the proposed RTO scheme is not superior to that of dynamic RTO's, the small difference in the production (0.4%) is acceptable considering the significant reduction of constraint violations. Moreover, due to the model mismatch in MPC controller, no method could prevent constraint violations completely.

Fig. 5.5 shows the responses of x_{3R} under a same disturbance sequence. Under the steady-state and dynamic model-based RTO methods, x_{3R} violates the upper bound of 0.025. The proposed RTO could always keep the x_{3R} within the bounds regardless of the uncertainties. Figs. 5.6 and 5.7 show the prescribed set points by RTOs and the corresponding production respectively. Figs. 5.8–5.10 show the state trajectories of x_{1M} and x_{3R} under different RTO schemes. Whereas the steady-state RTO and dynamic RTO violate the constraint and drive the system to the failure state, the proposed RTO detects the latent risk earlier and selects proper set points to keep the process away from the failure state.

Table 5.5: Production and constraint violation ratio in reaction-storage-separation network

	Steady state RTO	Dynamic RTO	Proposed RTO
Constraint violation ratio	0.39	0.36	0.24
Average production[‡]	1549	1612	1604

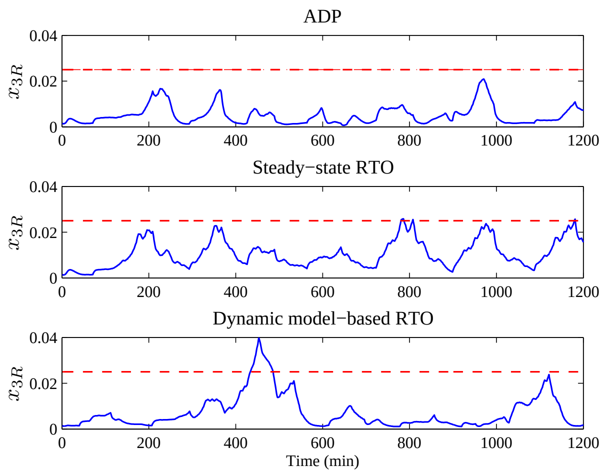

Figure 5.5: Sample responses of x_{3R} under three different RTO schemes. The dashed line is the constraint.

5.4.2 Tennessee Eastman Challenge Process

Problem Description

The Tennessee Eastman (TE) challenge process [31] is a realistic simulation of an integrated chemical plant that has been widely used in the real time optimization and fault detection studies. The Fortran code represents this process is available, but

‡ $\frac{1}{100}\sum_{i=1}^{100}\sum_{t=0}^{1200} B_i(t)x_{2B,i}(t)$

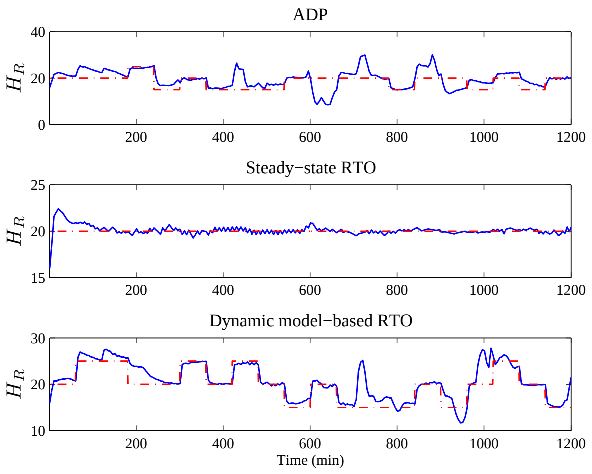

Figure 5.6: Sample responses of H_R under three different RTO schemes. The dashed line is the set point trajectory calculated by each RTO scheme.

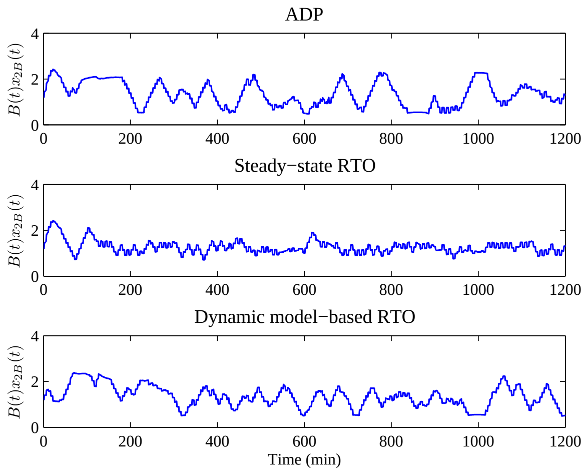

Figure 5.7: A sample plot of production under three different RTO schemes.

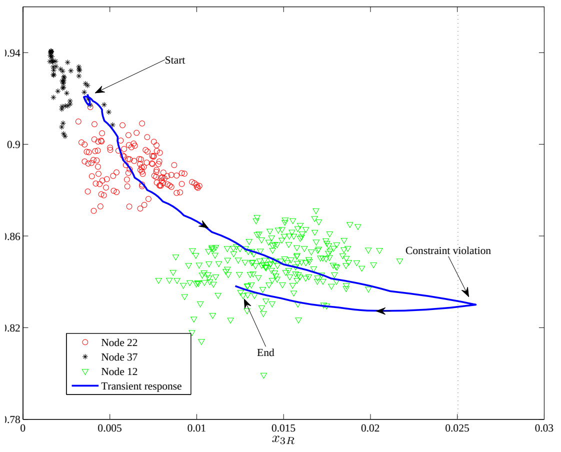

Figure 5.8: State transitions under steady state RTO.

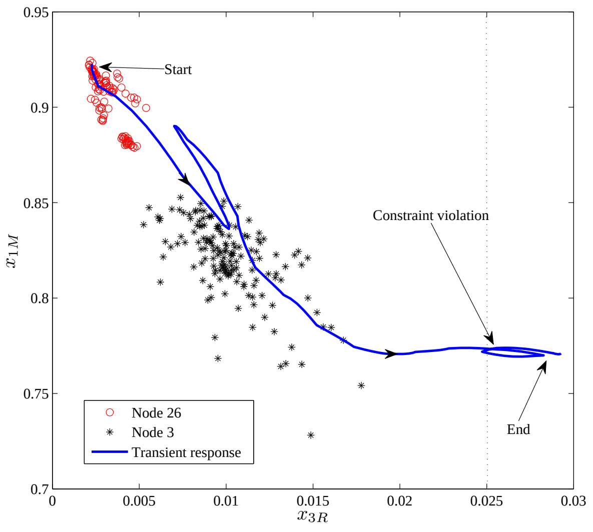

Figure 5.9: State transitions under dynamic model-based RTO.

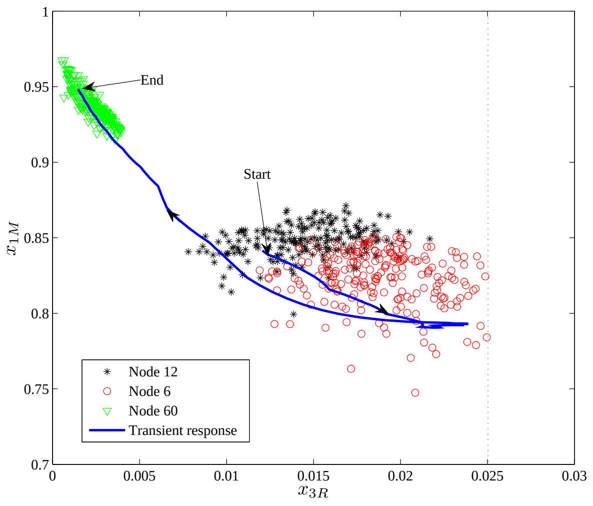

Figure 5.10: State transitions under proposed RTO.

not published. However, we know that all models can be expressed as the first order nonlinear differential equation. In another word, it satisfies the Markov property. The whole system consists in an exothermic two-phase reactor, a flash separator, a condenser, a reboiled stripper, and a recycle compressor. There are 41 measured outputs with added measurement noise. The 19 composition measurements are sampled at two different rates with time delay. Totally 12 manipulated variables can be used to regulate the process. The P&ID of the TE process is shown in Fig. 5.11.

The TE process produces two products (G and H) and one undesired byproduct(F) from four reactants (A, C, D and E), according to the following four reactions:

$$\begin{aligned} A+C+D &\longrightarrow G \quad Product1 \\ A+C+E &\longrightarrow H \quad Product2 \\ A+E &\longrightarrow F \quad Byproduct \\ 3D &\longrightarrow 2F \quad Byproduct \end{aligned}$$

[31] provides six specified operating modes in terms of the ratio of two products G and H. In this work, we consider the mode 1 where G:H mass ratio is 50:50. The objective of plant-wide optimization is to minimize the following operation cost:

$$\sum_{i=0}^{\infty} \gamma^i (-r_i) \tag{5.46}$$

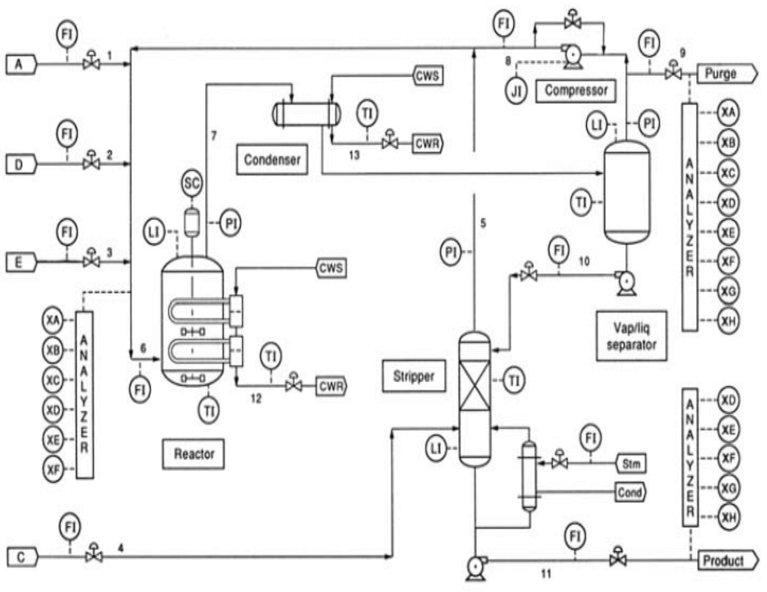

Figure 5.11: The Tennessee Eastman Process [31]

where r_i is the operation cost per stage. The operation cost is defined in the TE simulator [31] as

$$
\begin{aligned}
-r =& \text{purge cost} \times \text{purge rate} + \text{product stream cost} \times \text{production rate} + \\
& \text{compressor cost} \times \text{compressor work} + \text{steam cost} \times \text{steam rate}
\end{aligned}
$$

Many optimization schemes for TE process have been proposed with satisfactory results, e.g., see [43, 54, 32]. However, many of these schemes are focused on the conventional steady-state RTO. In addition, previous dynamic RTOs for TE process require an accurate plant-wide model or ignore disturbances, which limits their applications. Considering that the TE process is governed by the first order DAEs, it can be viewed as a Markov decision process. Thus, the proposed RTO method can be used to learn from the closed-loop data and bring evolutionary improvements of performance based on existing policies.

The nonlinear MPC controllers proposed by [101] are used as local controllers for closed-loop simulations. Since there is no available information for the selection of set points, five sets of set points were chosen within acceptable regions as shown in Table 5.6.

Table 5.6: Candidate sets of set points for TE process optimization

Set point	1	2	3	4	5
Production rate (kmol/h)	339	339	339	339	339
Reactor pressure (kPa)	2400	2420	2405	2390	2400
A in react feed (mol%)	36.4	36.4	36.4	33.4	34.4
E in react feed (mol%)	17	15	17.5	17	17
B in purge (mol%)	15.9	19.9	16.8	16.5	17
G in product (mol%)	53.7	53.7	53.7	53.7	53.7
Separator liquid level (%)	50	60	45	58	65
Stripper liquid level (%)	50	60	45	58	80

Disturbance

In the original problem, there are 15 disturbances [31]. In this work, only three of them are considered: A feed loss, A, B, C feed composition in stream 4, and drift in reaction kinetics. The reason is that more disturbances may result the system highly instable under the controller provided by [101]. Although the disturbance sequences provided by the TE simulator are fixed, the random open-close signals as in Fig. 5.12 are employed. Therefore, the behavior of system is still stochastic.

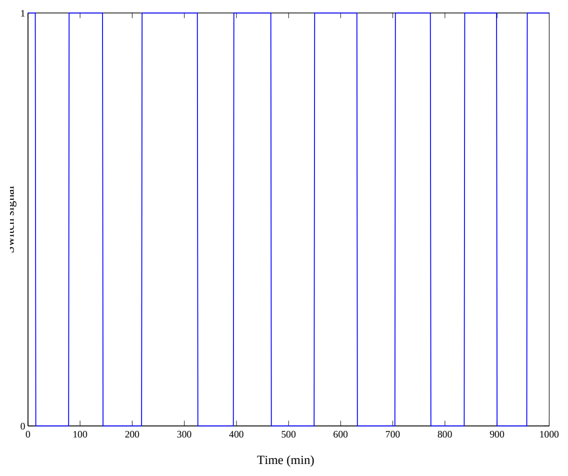

Figure 5.12: Switch signals for a disturbance input

Collecting Historical Data

Under the five different sets of set points, closed-loop simulations were performed 146,156,139,160, and 144 times respectively. For better perturbation, 225 different sequences with set point changes were also tested. In each episode of closed-loop simulations, total simulation duration was 3.5 hours and the time interval for optimization was set as 24 minutes. As a result, a total of 7864 data points were collected for Markov chain modeling.

Since the number of initial policies was too small to explore the large state space, five different ADP policies with different SOM structures were employed for further explorations. The initial policies derived from the original data set did not perform well, sometimes even worse than the steady-state RTO. This is mainly because the initial data points were coming from 5 scenarios with fixed set point, thus, hardly to describe the dynamics of the process properly. Moreover, the sparse data distribution leaves small room for exploration, and the resulting transition probability may not be accurate for deriving an improved policy. However, one advantage of the proposed scheme is that further refinement of model and policy is still possible by exploring the state space under the ADP policies. Thus, the first ADP policies were employed to generate more data and were continually updated until no appreciable improvement was observed.

Markov Chain and Value Function Learning

Based on the original data set, five different SOM structures with 54, 42, 43, 40, and 60 clusters were employed to obtain five more different policies for exploration. In the actual learning step, 6×10 grid was chosen based on the outlier criterion with the minimum number of ten data points in each cluster.

The upper bound of shutdown ratio was specified as 25%, and the minimum risk was calculated by conventional DP as 4.32% based on transition probability matrix. Given the inequality of Eq. (5.33) and the model uncertainty, D was set as 4. The value for the failure node was obtained as $V_F = -222470$ with $\overline{r} = -3827$ and $\underline{r} = -7066$. The learning parameters of ADP-based RTO is shown in Table 5.7.

Table 5.7: Learning parameters for the proposed RTO. N is the number of data in the outlier region.

Parameters	Reactor-storage-separator	Tennessee Eastman
γ	0.9	0.9
SOM structure	6×10	6×10
V_F	-4277	-3708
D	4	4
ϵ	0.4	0.15
α	$\frac{1}{N+1}$	$\frac{1}{N+1}$

Simulation Result

A hundred fresh disturbance switch sequences were used to test and compare different operating policies. Three important variables were considered for plant shutdown: reactor pressure, separator liquid level, and stripper liquid level. (See Table 5.8).

Table 5.8: Constraints considered in TE process

Process variable	Lower bound	Upper bound
Reactor pressure	none	2950 kPa
Separator liquid level	1%	100%
Stripper liquid level	1%	100%

The constraint violation ratio and operation costs are shown in Table 5.9. The first 5 results with fixed setpoint show that an aggressive operation manner has the lower cost but higher violation ratio. The first ADP policies, policy 1–5, are worse than the best steady policy (set point 4). However, ADP-based policy based on these initial data from the random policies as well as static set point policies yielded the best performance with the minimum number of constraint violation and mild operation cost. This shows that the data-driven based optimization scheme has evolutionary learning capability as more data points are available. Compared to the other policies, the learned policy finds the best tradeoff between the risk and the profitability. Further reduction on the cost may be achieved if the $|V_F|$ decreases appropriately, considering that the bound (5.24) is not very tight.

Table 5.9: Comparison of RTO performances

Policy	Constraint violation ratio (%)	Operation cost/ 3.5 hrs [$]
Setpoint1	46	623.2
Setpoint2	62	611.9
Setpoint3	52	620.2
Setpoint4	27	636.0
Setpoint5	33	646.2
Policy1 (6×9)	38	630.2
Policy2 (6×7)	42	630.0
Policy3 (4×8)	32	636.0
Policy4 (5×8)	37	637.0
Policy5 (6×10)	41	630.4
Final policy (6×10)	17	649.1

Figs. 5.13 - 5.15 show a sample of test results. Since the original MPC controller has very small weights on the levels of stripper and separator compared to the

production term, the liquid levels under the MPC do not follow the set points perfectly. This setting may lead the system to shutdown under the disturbances. In this case, both of the tanks are almost drained during 1.5–2.5 hours. The proposed RTO scheme increases the set points of the tank levels accordingly at the cost of production amount. Fig. 5.16 also shows the state transitions during 1.5–2.0 hours.

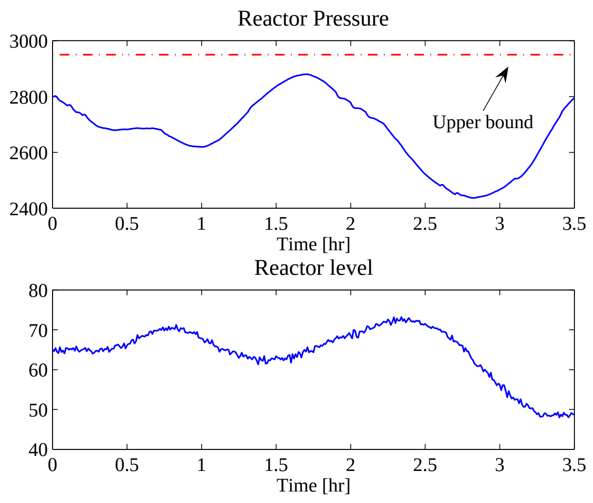

Figure 5.13: Reactor pressure and level

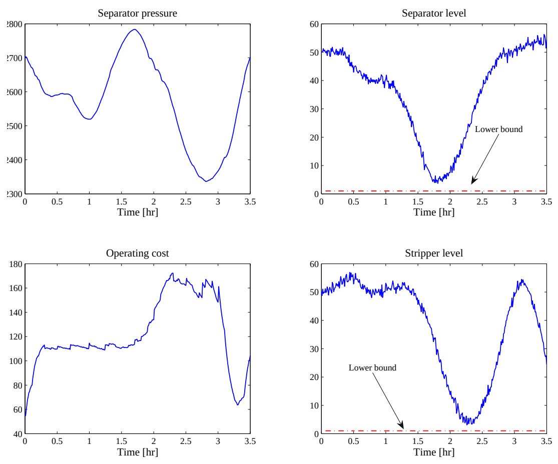

Figure 5.14: Separator pressure and level, stripper level, and operation cost.

5.5 Summary

Dynamic optimization based on a Markov chain model and approximate dynamic programming can provide a more robust and risk-averse policy compared to other

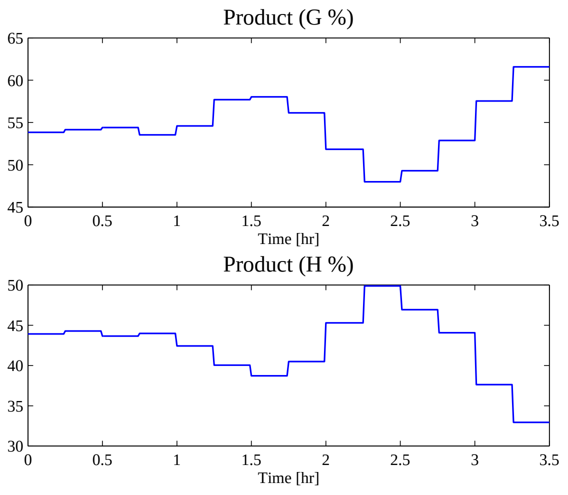

Figure 5.15: G and H in product.

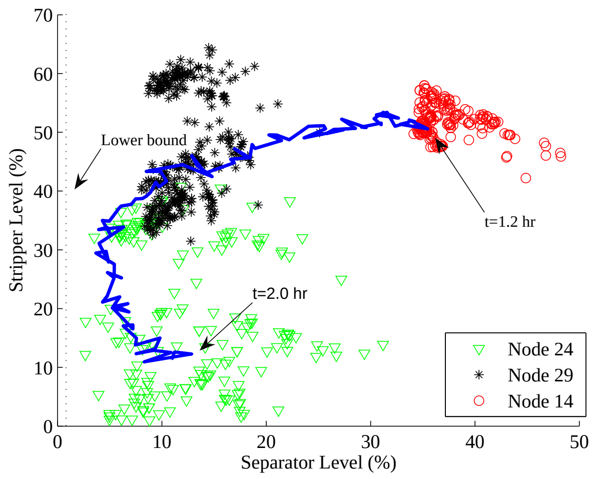

Figure 5.16: State transitions during 1.2 − 2.0 hour.

deterministic model-based approaches. The proposed approach can be applied to a class of dynamic optimization problems where a process model is difficult to obtain, which is likely for large-scale problems, and uncertainties affect operations significantly. The other advantage is that proposed scheme only solves a single stage optimization problem, thus reducing online computational burden significantly. A systematic guide for deriving a reward-to-go of the undesirable state was provided by considering the failure node as an absorbing state. Case studies on an integrated plant and the TE process show improved operations under the proposed scheme compared to the other conventional RTO strategies.

It is worth noting that all the state variables were assumed to be observed directly, which does not hold for many practical cases. However, previous work by [86] showed that such a Markov chain model can be constructed with observed data only, possibly using a higher-order one. Moreover, modeling such a process as a partially observed Markov decision process (POMDP) can be worth investigating further in such a case .

Chapter 6

Decomposition Strategy for Large Scale Scheduling Problems

6.1 Background

Proper production planning, storage location, transportation and scheduling are critical to energy saving and productivity enhancement of chemical process industries in the global economy. Since these real world optimization problems often lead to mixed integer linear models, development of new algorithms that allow for the solution to mixed integer linear programming (MILP) in acceptable times has recently attracted much attention. Though commercial softwares, such as CPLEX, provide effective platforms to solve MILP, there are still certain problems for which those softwares require an excessive amount of time to yield the optimal solution.

Benders decomposition was originally developed for solving "mixed-variables programming" problems where the decision variables can be decomposed into easy and complicating variable sets, which are mutually exclusive [9]. If the complicating variables, e.g., integer variables, are fixed at certain values, the remaining problem becomes a linear programming, which is relatively easy to solve. In his seminal work, [9] solved a small-sized MILP with 29 continuous variables and 27 integer variables.

Since the original decomposition strategy is not necessarily limited to MILP, Benders decomposition has also been suggested as a solution procedure for a broad range of optimization problems. For instance, "L-shaped" method based on Benders cut was first applied to stochastic programming [112]. The nested Benders method

[*]A full version of this chapter has been submitted to the *Computers and Chemical Engineering*

can handle the case where the deterministic equivalent linear program to a stochastic program is very large [16]. In solving dynamic programming, Benders cut is employed to mitigate the "curse of dimensionality" [92]. Benders decomposition is generalized to the problem whose subproblem could be nonlinear [41], and its convergence properties are provided [106]. A Benders hierarchical decomposition approach is also proposed to handle non-convexity for power transmission network design [15].

In particular, a class of MILP problems with "ladder-structure" constraints can be effectively solved by Benders decomposition. It first decomposes the original formulation into the master problem (MP) involving complicating variables and several sub-problems (SPs) having relatively easy variables only, and solves them in an iterative manner. The solution of MP is used as the parameter vector of the SPs, and the resulting solution of each SP determines a new cutting plane, i.e., a constraint added to the MP. It is proven that repeating this procedure can eventually find the optimal solution [57]. For a certain class of problems such as network design and multi-commodity scheduling, this method has shown its efficacy and achieved remarkable successes [77].

However, previous studies also show that Benders decomposition converges very slowly for certain applications. Thus, how to speed up the Benders decomposition algorithm has attracted much attention last several decades. There are two different kinds of approaches to enhance the convergence rate. The first methodology notices that the most computational burden comes from MP involving complicating variables. Since the MP is usually an integer programming (IP) problem, which is not tractable, the focus has been on solving IP approximately. One simple approach is to find a feasible integer solution instead of an optimal one [26]. Genetic algorithm along with a heuristic method is also applied to obtain a feasible solution of MP [93].

Another way to circumvent the difficulty in solving MP is reducing the number of iterations by seeking for a better cut. A well known method falling into this category, called Pareto-optimal cut, finds the cutting plane that is not dominated by other constraints [77]. This way, the tighter cut can decrease the number of iterations by restricting the objective value of MP more efficiently. Since it mainly focuses on the case where a SP has the multi-bounded-optimal solution, it is applicable to some special examples including network optimization problems. Furthermore, it cannot handle infeasible SPs. It is also noted that a host of infeasible points can be generated by solving MP repeatedly. This eventually leads to generation of many feasibility cuts from SPs, which becomes computationally expensive [104, 100, 90].

The maximum feasible subsystem cut generation strategy is proposed to overcome the feasibility issue [104]. Because an infeasible linear programming problem can be relaxed to have a feasible solution, this method makes the minimum number of modifications to its infeasible solution space and yields a new optimality cut in each iteration step. Nevertheless, it is difficult to quantify or prove mathematically how the new constraint can improve the efficiency. Compared with the conventional Benders decomposition, which only generates one constraint from each SP in every iteration step, the covering cut bundle (CCB) strategy yields more constraints at a time [105]. Although this method can reduce the number of iterations thereby saving much time in their example, the potential risk lies in the increment of the time in each iteration step. How to balance them is still an open question. A local branching technique replaces some of the feasibility cuts with local cuts [100]. All these works are concerned with generating a tighter cutting plane to enhance the algorithmic efficiency in solving MP.

The second category of approaches considers the cases where the structure of SPs is extremely complex. This arises when the original formulation can be only decomposed into MP and one SP. Consequently, this SP may become a large-scale LP problem. Since commercial solvers usually employ the interior point method for large-scale LPs, whose solution is not an extreme point, Benders decomposition may converge to a wrong solution. To address this issue, a condition is provided that guarantees the convergence for inexact cuts [131].

In this chapter, the first category is considered where SPs become infeasible given the solution of MP. A novel Benders decomposition approach is proposed where a bilinear optimization problem is solved sequentially to assist in selecting the tightest constraint for the region located by specified point among all suitable cutting planes. This method can be particularly relevant in chemical engineering applications, which often have to consider the continuous and integer decision variables. In addition, in those problems, these two sorts of variables usually can be divided into several exclusive set. We notice that [35, 25] have proposed similar ideas with our method, however, [35] decides to minimize the cardinality heuristically to find a better cut and [25] devotes to search the facets of of the polyhedron passing through the project point. Both of them are different from our algorithm in essence.

This chapter is organized as follows: the preliminary of the conventional Benders decomposition method is reviewed in Section 6.2; next a new strategy for choosing constraints is discussed in Section 6.3. Then a scheduling of multi-product batch plants example is presented in Section 6.4 to show the computational efficiency of the proposed method. Finally, the concluding remarks are provided in Section 6.5.

6.2 Preliminary

Consider the following MILP problem where the constraint matrix has the ladder block structure:

$$\begin{aligned} \max_{x,y}\ & c^T x + f^T y && \\ s.t.\ & Ax && \leqslant b \\ & Bx + Dy && \leqslant d \\ & x \in Z^+,\ y \geqslant 0 && \end{aligned} \tag{6.1}$$

where $y \geqslant 0$ means each component of y is non-negtive.

Problem (6.1) has the ladder block form where the complicating variables (x) exist in both constraint blocks and other variables (y) are shown in the "sub-block." Thus Benders decomposition can be applied by dividing the original formulation into MP and SP.

Master problem (MP):

$$\begin{aligned} \max_{x,z}\ & c^T x + z \\ s.t.\ & Ax \leqslant b \\ & x \in Z^+,\ z \leqslant W \end{aligned} \tag{6.2}$$

where W is a large enough positive number to bound the z. Let $\{\hat{x}, \hat{z}\}$ denote a solution to MP. Then SP becomes

Subproblem (SP):

$$\begin{aligned} z(\hat{x}) = \max_{y}\ & f^T y \\ s.t.\ & Dy \leqslant d - B\hat{x} \\ & y \geqslant 0 \end{aligned} \tag{6.3}$$

If $\hat{x}$ is not a feasible solution for the original problem, (6.3) does not have a feasible solution. Hence, the following dual problem is considered:

$$\begin{aligned} z(\hat{x}) = \min_{p} \;& p^T(d - B\hat{x}) \\ s.t. \;& D^T p \geqslant f \\ & p \geqslant 0 \end{aligned} \tag{6.4}$$

There are two possibilities in solving the dual SP. First, the solution from the MP makes the primal SP feasible then a bounded solution is obtained by the solution of the dual SP. In this case we compare $z(\hat{x})$ and z. If $z(\hat{x}) < \hat{z}$, the new constraint, named optimality cut $p^T(d - Bx) \geqslant z$ needs to be added to the MP. Otherwise, it indicates that the true solution is found. Second, if the primal SP is infeasible, (6.4) becomes unbounded. The solver will return an extreme ray r of the dual solution space. In this case, the feasibility cut $r^T(d - Bx) \geqslant 0$ is added to the MP to remove the infeasible solutions of MP. The procedure is briefly described in Fig. 6.1. Here multiple subproblems is formulated if decision variables can be divided into several blocks according to the ladder structure.

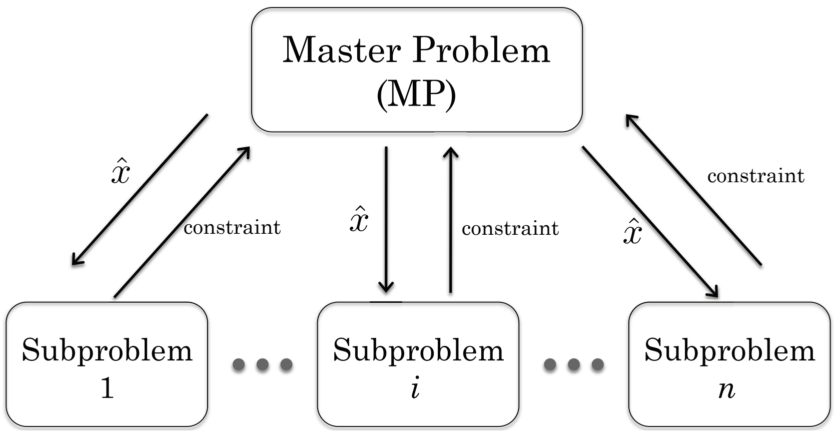

Figure 6.1: Iterative scheme of Benders decomposition.

Since p and r are the extreme point and ray of the feasible region, respectively, we may enumerate and add all of them to the MP. Then, the true value can be obtained in one step. However, considering the number of these constraints, it is practical to employ a "delayed" constraints generation strategy. That is, only a constraint which can reject the current solution of the MP is added in order to introduce the smallest number of constraints and save computational time.

6.3 The Tightest Cut Generation Strategy for Speeding up Benders Decomposition

For (6.2), given a set of cuts generated by SP, denoted by P and $p_1, p_2 \in P$, the constraint $p_1^T(d - Bx)$ is said to dominate $p_2^T(d - Bx)$, if $p_1^T(d - Bx) < p_2^T(d - Bx)$ for $\forall\ x$. The Pareto-optimal method always chooses the cut that no other cuts can dominate. This method is for the case where the optimality cut is readily available and the SP has multiple solutions. However, each iteration step in Benders decomposition gets a feasibility cut instead of the optimality cut most of the times, which is the major reason for slow convergence rate. This section proposes a faster method that finds the tightest feasibility cut for some regions in each iteration step.

Feasibility cut is defined in the subspace determined by the MP, and it is used to eliminate infeasible solutions in the MP. By adding these constraints continually, the MP can gradually characterize the true boundaries corresponding to the feasible region of the original problem. However, each feasibility cut generated from the extreme ray can only dominate a small area. Hence, if one can select those tight cuts, which construct the boundary related to the final integer solution, the number of iteration steps can be reduced dramatically. By comparing the two constraints of MP in Fig. 6.2, we can see that the line connecting the infeasible and feasible points must cross a true boundary, a hyper-plane. Therefore, the tightest constraint can be obtained by computing the distance between the feasible point and the intersection point. Only the hyperplane with the minimum distance to feasible points is added to the MP.

It is also worthwhile to note that the distance is not a necessary condition to determine the tightest constraint because different feasible points return various cutting planes according to their distances as shown in the feasible point 2 of Fig. 6.2. Hence, the proposed approach samples several feasible points and renders the cutting plane as close as possible to these points.

6.3.1 Determination of the Tightest Cutting Plane for One Feasible Point

Explicit computation and comparison of distances require solving a nonlinear programming problem, which is unacceptable in large-scale MILPs. However, once the feasible and infeasible points are fixed, all possible cuts will be crossed by the line linking these two points, and the minimum distance ratio λ is relatively easy to

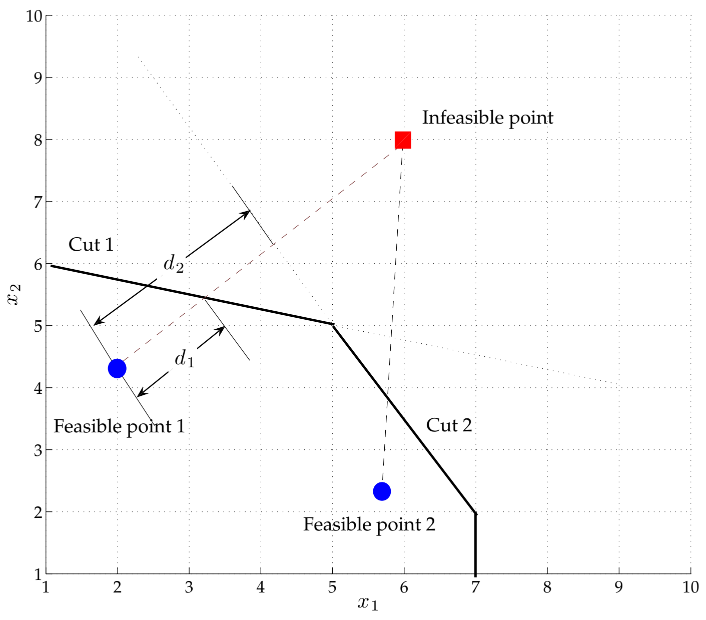

Figure 6.2: The line linking feasible and infeasible points

compute by the following optimization formulation:

$$\lambda = \frac{\text{Distance from the feasible point to the intersection at a cut}}{\text{Distance from the feasible point to the infeasible point}}$$

Problem 1:

$$\min_{r,\lambda} \ \lambda \tag{6.5}$$

$$s.t. \quad r^T(d - Bx_f - B\bar{x}\lambda) = 0 \tag{6.6}$$

$$r^T D \geqslant 0 \tag{6.7}$$

$$r_{(j)} \leqslant M \tag{6.8}$$

$$r, \lambda \geqslant 0 \tag{6.9}$$

where x_f and x_i represent the feasible and infeasible points, respectively. M is a big enough number. $\bar{x} = x_i - x_f$ is the vector from the feasible point to the infeasible point, and $r_{(j)}$ is the jth component of the vector r.

Let us now consider the constraint (6.6). Any positive λ satisfying this equality represents the distance ratio given by r. The intersection point can be expressed as $x_t = x_f + \lambda\bar{x}$, and plugging it to the feasibility cut equation yields this constraint.

The constraints (6.7) and (6.8) imply that r should be a ray of the dual subproblem. Since r is a vector, we specify the upper bound of its element as a reasonably large number M to avoid numerical artifacts and infinite magnitude. The solution of Problem 1 corresponds to the minimum distance ratio λ, thereby constructing the tightest boundary of the region located by the specified feasible point.

The above formulation is a bilinear optimization problem, which conventional LP solvers cannot solve. However, λ can be written as

$$\lambda = \frac{r^T(d - Bx_f)}{r^T B\bar{x}} \tag{6.10}$$

Eq. (6.10) yields fractional programming, which is also difficult to solve. Hence, an iterative scheme called Dinkelbach's method [30] is employed to get the optimal value of λ. Assume that we have an initial guess $0 < \lambda_0 < 1$, then consider the following problem:

$$\begin{aligned} L = \min_{r} \quad & r^T(d - Bx_f) - \lambda_0 r^T B\bar{x} && (6.11)\\ s.t. \quad & r^T D \geqslant 0 && (6.12)\\ & r_{(j)} \leqslant M && (6.13)\\ & r \geqslant 0 && (6.14) \end{aligned}$$

In order to discuss some properties of the above formulation, the following lemma is introduced:

Lemma 19. *Suppose λ_0 corresponds to the distance ratio of the ray that leaves out x_i from the feasible region. Let r_1 be the solution to (6.11) and assume the objective function is negative. Then,*

1. *$r_1^T(d - Bx_i) < 0$ always holds.*
2. *r_1 has a smaller distance ratio than λ_0.*

Proof. Since no constraint derived from the ray can cut out x_f, it follows that $r_1^T(d - Bx_f) \geqslant 0$. $\lambda_0 < 1$ also holds because it corresponds to the ray cutting out x_i.

From the objective function, we have

$$\begin{aligned} & r_1^T(d - Bx_f) - \lambda_0 r_1^T B\bar{x} < 0 \\ \rightarrow & (1 - \lambda_0) r_1^T(d - Bx_f) + \lambda_0 r_1^T(d - Bx_i) < 0 \\ \rightarrow & r_1^T(d - Bx_i) < -\frac{(1 - \lambda_0) r^T(d - Bx_f)}{\lambda_0} \leqslant 0 \end{aligned} \tag{6.15}$$

Moreover, there is

$$r_1^T B\bar{x} = r_1^T(d - Bx_f) - r_1^T(d - Bx_i) > 0 \tag{6.16}$$

If the objective (6.11) is negative, the following inequality also holds:

$$\lambda_1 = \frac{r_1^T(d - Bx_f)}{r_1^T B\bar{x}} < \lambda_0 \tag{6.17}$$

This means that the objective value corresponding to r_1 of Problem 1 is smaller than λ_0. $\square$

In order to get the tightest cut, the iteration on λ is performed until (6.11) becomes zero, meaning there is no superior cut. In addition, it is also guaranteed that λ decreases monotonically and converges [30].

Remark 1: M only affects the magnitude of ray, not the direction. Theoretically, the performance of the cut is not influenced by its value, but a large value of M can allow the solver to handle more digits.

Remark 2: The initial guess λ_0 can be obtained by calculating the λ of the extreme ray provided by CPLEX.

Remark 3: During this procedure, rays $r_1,\ r_2,\ \ldots,\ r_n$ and their corresponding cuts are generated successively. Though only the final constraint is the tightest one for a given feasible point, other constraints are stored in the pool for generating feasible points in the next step.

Remark 4: Once the optimal solution is found, the resulting cut is the tightest one. However, If the size of (6.11) is large and takes much time to solve, one can set a finite number of iterations for λ and proceed with an improved λ value to save the computing time. In that situation, we only get a tighter cut for this feasible point. For simplicity, we assume that the optimal λ is always obtained in this paper. Actually, significant improvement is achieved just after one iteration.

6.3.2 Selection of the Feasible Point

Sequential search

Similar with the approach in [105], the proposed framework can generate several tightest cuts $C^1,\ C^2,\ \ldots,\ C^K$ and their rays $R^1,\ R^2,\ \ldots,\ R^K$ by selecting a series of feasible points $x_f^1,\ x_f^2,\ \ldots,\ x_f^K$ in each iteration. Since the selection of cuts depends on the location of feasible points, a systematic procedure is necessary for specifying feasible points.

We propose a scheme to search feasible points sequentially. After the initialization of the first feasible point x_f^1, its successor point is located in a different region where the previous selected feasibility cuts cannot bound tightly. For example, let us consider searching for the feasible point x_f^k. It is noted that Dinkelbach's method can generate many cutting planes, denoted by $L_1,\ L_2,\ \ldots,\ L_{n-1},\ L_n$, which correspond to the rays $r_1,\ r_2,\ \ldots,\ r_{n-1},\ r_n$ for the previous feasible point x_f^{k-1}. Though other constraints are not tighter than L_n according to the x_f^{k-1}, they are still stored in the constraints pool because there may be a cutting plane in the pool, say L_1, which is tighter than L_n for other feasible points. Hence, in order to explore properties of those planes further, we search for a new feasible point x_f for which L_1 has a smaller value of λ than L_n.

Given

$$\lambda_1 = \frac{r_1^T(d - Bx_f^k)}{r_1^T B\bar{x}} \tag{6.18}$$

$$\lambda_n = \frac{r_n^T(d - Bx_f^k)}{r_n^T B\bar{x}}, \tag{6.19}$$

and

$$\lambda_1 > \lambda_n \iff 1 - \lambda_1 < 1 - \lambda_n, \tag{6.20}$$

consider the following inequalities:

$$\frac{1-\lambda_1}{1-\lambda_n} \leqslant \alpha \tag{6.21}$$

$$\rightarrow \frac{r_1^T(Bx_i - d)}{r_1^T B\bar{x}} \frac{r_n^T B\bar{x}}{r_n^T(Bx_i - d)} \leqslant \alpha \tag{6.22}$$

$$\rightarrow r_n^T B\bar{x} r_1^T(Bx_i - d) \leqslant \alpha r_1^T B\bar{x} r_n^T(Bx_i - d) \tag{6.23}$$

$$\begin{aligned}\rightarrow &\{r_n^T B(r_1^T(Bx_i - d)) - \alpha r_1^T B(r_n^T(Bx_i - d))\}x_i \leqslant \\ &\{r_n^T B(r_1^T(Bx_i - d)) - \alpha r_1^T B(r_n^T(Bx_i - d))\}x_f^k\end{aligned} \tag{6.24}$$

where α is a user-specified threshold.

The inequality (6.24) is a linear constraint in x_f, which is easy to solve using CPLEX. For any feasible point satisfying this constraint, the plane L_1 is tighter than L_n according to the resultant x_f. In order to check if L_1 can cut the infeasible region, it is only necessary to find if there is a x_f for which L_1 is tighter than previous constraints $C^1,\ C^2,\ \ldots,\ C^{k-1}$. Hence, the identified point is set as the k^{th} feasible point.

Searching in the equipotential hyperplane

Though feasible points can be found in the whole feasible region, it is more efficient to focus on some special region than others. For instance, the area with larger objective value of MP is more important than the smaller one because the solver needs to check points to solve for the maximum solution to MP. Sampling feasible points and adding more cuts in that region to characterize the tight boundary can prevent the solution of MP from falling into infeasible regions of the original problem, thereby decreasing the total number of iterations.

Suppose the current MP generates the objective value V. A new set of constraints can be appended to identify feasible points on an equipotential hyperplane, where every point has the same objective value V for MP. Let us first consider the formulation of MP:

$$\begin{aligned}
\max_{x,z}\ & c^T x + z && (6.25)\\
s.t.\quad & Ax \leq b \\
& p_h^T(d - Bx) \geqslant z \quad \{h = 1, 2, 3, \ldots\} && (6.26)\\
& q_l^T(d - Bx) \geqslant 0 \quad \{l = 1, 2, 3, \ldots\} && (6.27)\\
& x \in Z^+,\ z \geq 0
\end{aligned}$$

where (6.26) and (6.27) are the optimality and feasibility cuts added in the previous iterations, respectively. For the purpose of characterizing feasible points on the equipotential hyperplane in the x space, the following set of constraints, which determine the value of z, is appended:

$$p_h^T(d - Bx_f) \geqslant z \quad \{h = 1, 2, 3, \ldots\} \tag{6.28}$$

$$c^T x_f + z = V \tag{6.29}$$

$$Ax_f \leqslant b \tag{6.30}$$

$$Bx_f + Dy \leqslant d \tag{6.31}$$

$$z \geqslant 0,\ y \geqslant 0 \tag{6.32}$$

The optimality cuts of (6.28) are related to the value of z, and Eq. (6.29) guarantees that the objective value of MP for x_f is V. Eqs. (6.30) and (6.31) confine x_f to be a feasible point of the original problem.

Formulation for selecting feasible points

In all, to search for the k^{th} feasible point, given a new plane L from the constraints pool, the following formulation is solved:

Problem 2:

$$Ax_f \leqslant b \tag{6.33}$$

$$Bx_f + Dy \leqslant d \tag{6.34}$$

$$p_h^T(d - Bx_f) \geqslant z \quad \{h = 1, 2, 3, \ldots\} \tag{6.35}$$

$$c^T x_f + z = V \tag{6.36}$$

$$\begin{aligned} &\{R^{i^T} B(r^T(Bx_i - d)) - \alpha r^T B(R^{i^T}(Bx_i - d))\}x_i \leqslant \\ &\{R^{i^T} B(r^T(Bx_i - d)) - \alpha r^T B(R^{i^T}(Bx_i - d))\}x_f \quad i \in \{1, 2, \ldots, k-1\} \end{aligned} \tag{6.37}$$

$$x_f \geqslant 0, \ y \geqslant 0, \ z \geqslant 0 \tag{6.38}$$

where r is the ray related to the cutting plane L chosen from the constraints pool.

Though new constraints are appended to the original LP relaxed problem, computational burden will not increase significantly considering there are only a few optimality cuts in this kind of problems. Moreover, Problem 2 is solved much faster without the objective function, and the solver will yield x_f, y and z that satisfy the constraints of Problem 2.

If there is a feasible solution to Problem 2, it indicates at least one feasible point can be found for which L is tighter than any other constraints $C^1, \ C^2, \ \ldots, C^{k-1}$. In other words, it actually locates the region where $C^1, \ C^2, \ \ldots, C^{k-1}$ cannot characterize the true boundary. Given this new feasible point, one can apply the procedure introduced in the previous section to yield a new cutting plane. If there is no feasible solution, L can be removed from the constraints pool and the next cutting plane is checked until the entire constraints in the memory are checked.

6.3.3 Constraints Selection

Though the above procedure can yield the tightest constraint for each feasible point and a systematic method to generate feasible points is provided, two issues still remain: how many constraints and which constraints should be appended to MP. Whereas conventional Benders decomposition generates only one cutting plane for each sub-block in each iteration, the bundle cutting plane generation scheme [105] yields multiple constraints in each iteration step to reduce the total number of iterations. We borrow this idea that adding more than one constraints to MP at a time to improve the performance. Though directly incorporating all the generated constraints looks attractive, it actually increases the number of constraints dramatically and slows down the computational time of each step. Therefore, it is necessary to develop a method that selects suitable constraints among all the tight cutting planes generated in the last section.

Suppose there are K available feasible points. The above procedure can be applied to each point, and one can obtain at most n different cutting planes. For each plane i and feasible point j, there is a λ_{ij}, defined as

$$\lambda_{ij} = \frac{\text{Distance from the feasible point } j \text{ to the intersection at a cut } i}{\text{Distance from the feasible point } j \text{ to the infeasible point}}$$

For simplicity, we can only calculate the following value to rank these cuts in terms of the tightness over all the feasible points:

$$\lambda^i = \sum_{j=1}^{n} \frac{1-\lambda_{ij}}{1-\lambda_{jj}} \tag{6.39}$$

The reason here we employ $1-\lambda$, not λ directly, is that the λ_{jj} becomes 0 if a vertex of the subspace is sampled as a feasible point. Though (6.39) provides a measure to assess the tightness of each constraint over the sampled feasible points, the ranking based on λ^i can change significantly depending on the locations of sampled feasible points.

In order to choose a manageable number of meaningful cuts, the following procedure is suggested with a new metric, ϱ:

Step 1: Compute (6.39) for cuts $C^1, C^2, \ldots,$ and C^K. Choose the maximum one as the reference constraint C^b and append it to MP

Step 2: Compare C^b with the other cutting planes based on the metric ϱ, called "maximum difference." Then the cuts with large enough differences, greater than a user-defined threshold γ, are selected.

In Step 2, given a constraint C_c, a new metric, the maximum difference between C_b and C_c is defined as:

$$\varrho(x_p) = \frac{d_{pq}}{d_{px_i}} = \frac{\text{Distance from } x_p \text{ to } x_q}{\text{Distance from } x_i \text{ to } x_p} \tag{6.40}$$

$$\varrho_{\max} = \max_{x_p} \varrho \tag{6.41}$$

where x_p is the point on the plane C^c and x_q is the crossing point of vector $\overline{x_p x_i}$ and the plane C^b. If the ϱ_{max} is larger than γ, there is a region where C^c is much tighter than C^b, see Fig. 6.3. Hence, the cutting plane C^c should also be appended to MP.

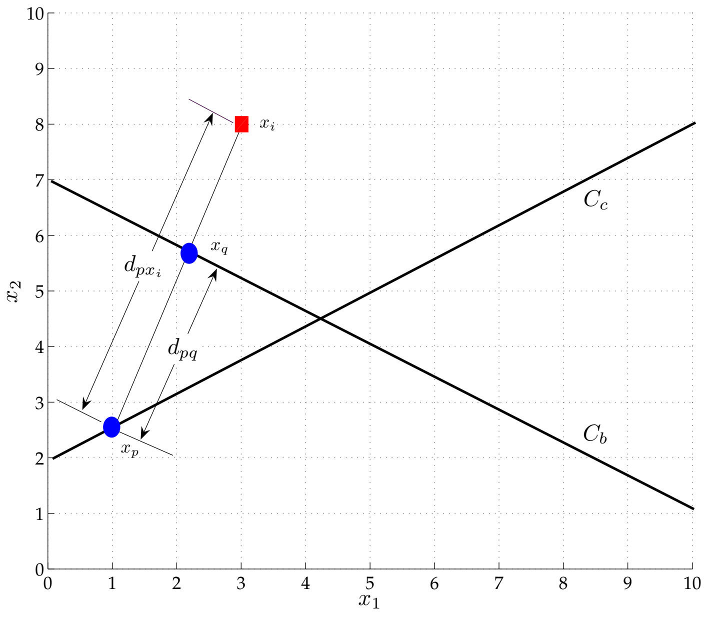

Figure 6.3: Comparison of two constraints

The following formulation can be solved to compute the maximum value of ϱ:

Problem 3:

$$\max_{x_p, z, \varrho} \quad \varrho \tag{6.42}$$

$$s.t. \quad z + c^T x_p = V \tag{6.43}$$

$$Ax_p \leqslant b \tag{6.44}$$

$$p_h^T(d - Bx_p) \geqslant z \quad \{h = 1, 2, 3, \ldots\} \tag{6.45}$$

$$q_l^T(d - Bx_p) \geqslant 0 \quad \{l = 1, 2, 3, \ldots\} \tag{6.46}$$

$$r_c^T(d - Bx_p) = 0 \tag{6.47}$$

$$\varrho = \frac{r_b^T(d - Bx_p)}{r_b^T B(x_i - x_p)} \tag{6.48}$$

$$x_p \geqslant 0, z \geqslant 0 \tag{6.49}$$

where V is the current value of objective function; r_b and r_c are the rays related to cut C^b and C^c, respectively. Same with the last section, the new point x_p is also restricted on the equipotential hyperplane. Constraints (6.44) to (6.46) characterize the current feasible region of MP and (6.47) confines the point on the cutting plane C^c. Eq. (6.48) can be derived from the definition of ϱ.

$$\varrho(x_p) \quad = \quad \frac{d_{pq}}{d_{px_i}} = \frac{|x_q - x_p|}{|x_i - x_p|} = \frac{r_b^T B(x_q - x_p)}{r_b^T B(x_i - x_p)} = \frac{r_b^T (d - Bx_p)}{r_b^T B(x_i - x_p)} \tag{6.50}$$

Eq. (6.40) is also a fractional programming problem, which can be solved in the same manner proposed in Section 6.3.1. Similar with Problem 1, optimal solution is not required. Once the ϱ becomes larger than a threshold, the iteration can be terminated and the constraint is added to MP.

Alternatively, the Problem 3 can also be solved through non-recursive manner. Note that the term $r_b^T Bx_p$ exists both in the numerator and denominator, the new non-fractional LP formulation can be constructed depending on parameters $r_b^T d$ and $r_b^T Bx_i$. Due to constraints (6.41) $\sim$ (6.47), the optimal solution may be searched and compared over the disjunctive set.

6.3.4 Algorithm

The whole procedure can be summarized as follows:

Step 1: Solve MP to get $\hat{x}$ and $\hat{z}$.

Step 2: Substitute $\hat{x}$ into each dual sub-problem and solve each block.

Step 3: For the sub-block which has a bounded solution, just obtain the optimality cut. For the sub-block which has an unbounded solution, obtain the extreme ray from LP solver (e.g., CPLEX) and compute λ_0.

Step 4: If there is $z_i(\hat{x}) > \hat{z}_i$ for each block i, the final solution is obtained and the algorithm is terminated; otherwise, go to Step 5.

Step 5: For the unbounded dual SP, $\hat{x}$ is the infeasible point. Let $j = 1$, initialize the feasible point x_f^1 and set the total number of feasible points as K.

Step 6: Solve the formulation **Problem 1** with x_f^j to get the tightest cut C^j. Also put extra cutting planes generated from Dinklebach's method into the constraint pool. If $j = K$, go to Step 8.

Step 7: Set $j = j + 1$. Plug the cut from constraint pool into **Problem 2** until a new feasible point is found. Meanwhile, any cuts rendering **Problem 2**

infeasible can be removed from the pool. If the pool is empty and no feasible point is found, re-initialize a new feasible point x_f^j. Then go back to Step 6.

Step 8: Solve **Problem 3**, select several constraints and add them to MP. Go back to Step 1.

6.3.5 Computational Burden

Since this work primarily focuses on reducing computational time of Benders decomposition, this section discusses the computational burden of each step in detail.

In Step 1, the MP is solved by commercial software such as CPLEX. This is the most time consuming part of this algorithm because the integer variables are usually involved in this problem. Compared to the original whole-scale MILP, the number of both variables and constraints of MP is reduced dramatically. Nevertheless, it may take several minutes to hours to solve even a small-scale MILP. The endeavor of this paper is reducing the number of iterations in this step.

In Step 2, each sub-problem is solved directly. Usually, only the LP model should be considered in this part. Thus, the amount of computational time is not comparable to Step 1. Since no inexact cut is considered in this method, the Simplex method is used to get the exact optimal solution for each subproblem.

In Step 6, the bilinear optimization is solved sequentially. Note that the number of variables and constraints of this bilinear problem is similar to that of the sub-problem; therefore, the computational time for each iteration is relatively small.

In Step 7, although more than one constraint are added to the original LP relaxed problem, the solver only needs to return the feasible solution, which can be found only in a few simplex iterations.

In Step 8, it is worthwhile to note that most of constraints come from MP, which is far smaller than the original LP. Hence, the optimal ϱ can be obtained very quickly. In order to compare other cutting planes in the pool with the reference cut, Problem 3 should be solved $K-1$ times. Therefore, it is necessary to limit the number of K to balance the efficiency and representativeness of the feasible points.

In summary, the algorithm aims to reduce the number of Benders iteration steps at the expense of solving three extra LP based problems. With proper tuned parameters, the total computational time can be reduced dramatically.

6.4 Case Study

In this chapter, a scheduling problem for multi-product, multipurpose batch plants is considered. The original mathematical formulation is proposed in [51]. The process is represented by a state task network shown in Fig. 6.4. A small modification is introduced to improve the efficiency. The original formulation considers the task and unit separately, thereby solution space involves 3-D integer variables. However, since each task is not performed at all the units, several variables are 0. In order to make use of this prior knowledge and reduce the complexity of the model, the task and unit are combined and only the 2-D variables are considered. For the details of the problem, see [51]. The indices are defined in Table 6.1, and the parameters and variables are introduced in Tables 6.2 and 6.3.

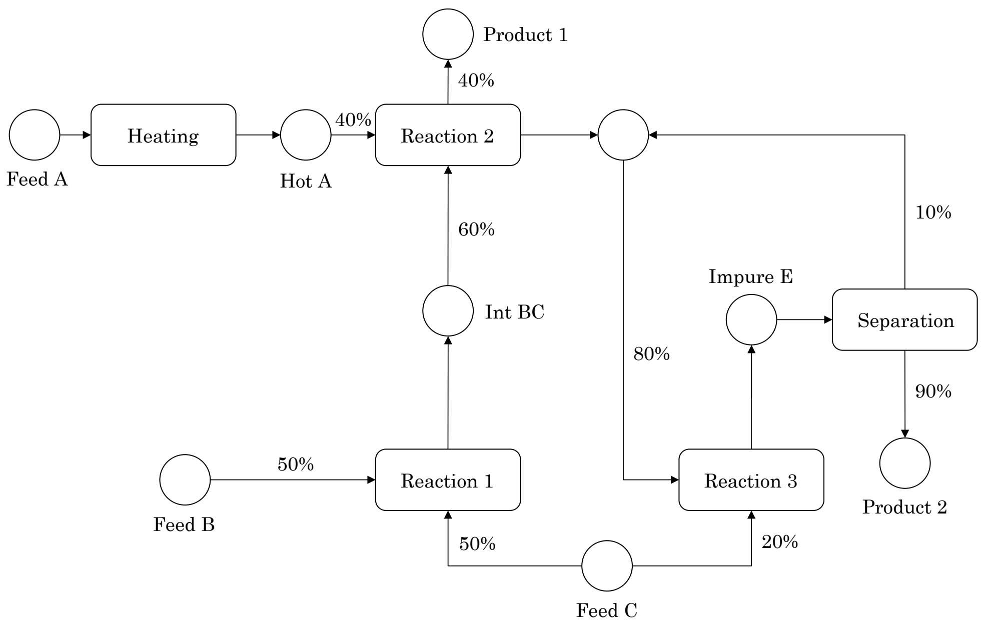

Figure 6.4: State task network of the case study

Table 6.1: Indices

I	Tasks
J	Units
N	Event points
S	States

The data is also from [51] and shown in Tables 6.4 and 6.5.

Table 6.2: Parameters

$Price(s)$	Price of product produced by unit s
$\rho^p(s,i)$	Proportion of state s produced by task i
$\rho^c(s,i)$	Proportion of state s consumed from task i
$st^{max}(s)$	Maximum storage capacity for state s
$V^{min}(i)$	Minimum capacity of the unit processing task i
$V^{max}(i)$	Maximum capacity of unit processing task i
H	Time horizon

Table 6.3: Varables

$wv(i,n)$	Binary variables that assign the task i at time point n
$d(s,n)$	Amount of state s being delivered at time point n
$b(i,n)$	Amount of materials undertaking task i at time point n
$st(s,n)$	Amount of state s at time point n
$Tf(i,n)$	Finishing time of task i at time point n
$Ts(i,n)$	Starting time of task i at time point n

Table 6.4: Data 1

Task	Capacity	Processing time
Heating	100	1.0
Reaction 1 in reactor 1	50	2.0
Reaction 2 in reactor 1	50	2.0
Reaction 3 in reactor 1	50	1.0
Reaction 1 in reactor 2	80	2.0
Reaction 2 in reactor 2	80	2.0
Reaction 3 in reactor 2	80	1.0
Separation	200	1.5

Table 6.5: Data 2

State	Storage capacity	Initial amount	Price
Feed A	Unlimited	Unlimited	0
Feed B	Unlimited	Unlimited	0
Feed C	Unlimited	Unlimited	0
Hot A	100	0	0
Int AB	200	0	0
Int BC	150	0	0
Impure E	200	0	0
Product 1	Unlimited	0	10.0
Product 2	Unlimited	0	10.0

In order to maximize the profit, the following MILP formulation is solved.

$$\max \sum_{s,n} \text{Price}(s) d(s,n) \tag{6.51}$$

$$s.t. \sum_{i} wv(i,n) \leq 1 \quad \forall n \tag{6.52}$$

$$st(s,n) = st(s,n-1) - d(s,n) \tag{6.53}$$

$$+ \sum_{i} \rho^{p}(s,i) b(i,n-1) - \sum_{i} \rho^{c}(s,i) b(i,n)$$

$$1 < \forall n \in N, \forall s \in S$$

$$st(s,n) \leq st^{max}(s) \quad \forall n \in N, \forall s \in S \tag{6.54}$$

$$V^{min}(i) wv(i,n) \leq b(i,n) \leq V^{max}(i) wv(i,n) \tag{6.55}$$

$$\forall i \in I, \forall n \in N$$

$$Tf(i,n) = Ts(i,n) + \alpha(i) wv(i,n) + \beta(i) b(i,n) \tag{6.56}$$

$$\forall i \in I, \forall n \in N$$

$$Ts(i,n+1) \geq Tf(i,n) - H(1 - wv(i,n)) \tag{6.57}$$

$$\forall i \in I, \forall n \in N$$

$$Ts(i,n+1) \geq Tf(i',n) - H(1 - wv(i',n)) \tag{6.58}$$

$$\forall i, i' \in I, \forall n \in N$$

$$Ts(i,n+1) \geq Ts(i,n) \quad \forall i \in I, \forall n \in N \tag{6.59}$$

$$Tf(i,n+1) \geq Tf(i,n) \quad \forall i \in I, \forall n \in N \tag{6.60}$$

$$Ts(i,n+1) \leq H \quad \forall i \in I, \forall n \in N \tag{6.61}$$

$$Tf(i,n+1) \leq H \quad \forall i \in I, \forall n \in N \tag{6.62}$$

The parameters are

$$\alpha_i = (2/3)\bar{T}(i) \tag{6.63}$$

$$\beta_i = (2/3)\bar{T}(i)/(V^{max}(i) - V^{min}(i)) \tag{6.64}$$

where the $\bar{T}$ is the mean processing time of task i. $wv(i, n)$ is equal to 1, if the task i is assigned to the time point n.

6.4.1 Implementation Issue

In the above formulation, the integer variables $wv(i, n)$ are involved in MP and the other continuous variables can be assigned to a single SP. However, having one SP is not efficient in using Benders decomposition in general. Here we employ the idea presented in [104] that the $wv(i, n)$ and $b(i, n)$ are combined into the same master group. Then the variables $d(s, n)$ and $st(s, n)$ are in SP1 and $Tf(i, n)$ and $Ts(i, n)$ are in SP2. In this case, the original problem is split into three blocks each of which is much easier to solve.

The conventional Benders decomposition is applied first. We found that most of the cuts were feasibility cuts, that is, the solver cannot find the feasible point for the original problem in the MP subspace. As a result, the Pareto-optimal cut method is not applicable in this case. On the other hand, the algorithm proposed in this chapter is suitable to handle feasibility cuts.

The implementation of both common Benders decomposition and the proposed method employs the heuristic strategy proposed in [100]. The procedure can be divided into two stages. In the first stage, the integer constraints are relaxed and a LP model is solved by Benders decomposition. Since this relaxed LP formulation is not very large to the commercial software, it can be solved quickly and some optimality cuts are added to the MP. Once the MP yields the optimal LP relax solution, the algorithm will switch to the second stage, i.e., the integer constraints are considered again.

The initial feasible points for the proposed method are obtained randomly by solving the relaxed LP formulation of the original problem with equipotential constraints and random objective function. Note that Problem 2 may not have a feasible solution despite enumerating the whole cutting pool. In this case, a random feasible point should be used.

Both the conventional Benders decomposition and proposed approach are tested in the same environment. The model is described by AMPL and solved by CPLEX (ver. 11.0.1). The computational platform is the IBM X-series 3550 with eight Intel Xenon processors running at 3.1GHz with 32 GB of RAM. The operating system is Windows Server 2003 R2 Enterprise x64 Edition with Service Pack 2.

The number of time points n and time horizon H are changed to test these algorithms in different cases. Since the 3-D variables are replaced by 2-D variables, the quantity of constraints and variables are reduced considerably. Thus, only the $n \geqslant 10$ cases are considered here because solving small scale examples using CPLEX is trivial and cannot show the difference between the proposed method and classical Benders decomposition. Usually, for the same n, the smaller H requires more computational time. The number of constraints and variables are also illustrated in Table 6.6. Although the problem may not be a large-scale LP, we note that the IP part makes the problem structure-dependent and more complex than solving LP only.

6.4.2 Discussions on Results

The main results are shown in Table 6.6. The CPU time 1 is for the classical Benders decomposition and CPU time 2 represents the computation time of the proposed method. Moreover, given that some of parameters in the proposed algorithm should be well-tuned, we also show their effects on computation time by varying those parameters one by one.

There are two important criteria for comparison. The first is CPU time and the other is the number of iterations. Both of them can show the efficiency of an algorithm. The number of iterations directly reflects the effectiveness of the selected cuts. However, the computation time within each iteration also varies depending on the selected cuts. Sometimes, a tighter cut may require longer computation time for MP. Therefore, the amount of CPU time is also significant to test the proposed approach. As shown in the Table 6.6, the proposed method outperformed Benders decomposition for all cases in these two tests. This is because the classical method just employs the extreme ray given by CPLEX, but the proposed method selects the ray for the tightest constraint in terms of the specified point.

In Table 6.7, we change K, the number of feasible points in each iteration. We select ten feasible points instead of five. For some cases, the performance is not improved remarkably compared with the case of $K = 5$. Note that the number

Table 6.6: Comparison of Benders decomposition with proposed method ($K = 5$,$\alpha = 0.1$,$\gamma = 0.4$)

n	H	Constraints	# of var.	Time 1	# of iter. 1	Time 2	# of iter. 2	Time saving (%)
10	20	874	400	321.6s	169	248.0s	127	22.9
11	23	966	440	627.4s	95	539.7s	74	14.0
11	22.5	966	440	1354.7s	162	1080.5s	131	20.2
11	22	966	440	2104.0s	217	1354.0s	141	35.6
12	24	1058	480	570.6s	30	214.0s	11	62.5
12	26	1058	480	60.8s	4	30.4s	2	50.0
13	29	1150	520	2826.0s	31	2413.0s	31	14.6
13	28	1150	520	3252.0s	37	1262.0s	20	61.2

Table 6.7: Comparison of Benders decomposition with proposed method ($K = 10$)

n	H	# of iter.	Time	Time saving (%)	Final # of constraints for MP
10	20	118	314.0s	2.4	288
11	23	79	581.7s	7.4	184
11	22.5	154	1515.0s		292
11	22	113	1124.0s	46.6	322
12	24	7	163.3s	71.4	69
12	26	2	27.6s	54.6	50
13	29	32	2215.2s	21.6	106
13	28	28	2639.5s	18.8	101

Table 6.8: Comparison of Benders decomposition with proposed method ($\alpha = 0.5$)

n	H	# of iter.	Time	Time saving (%)	Final # of constraints for MP
10	20	130	303.4s	5.7	265
11	23	89	614.2s	2.1	180
11	22.5	147	1266.7s	6.4	231
11	22	118	1027.0s	51.2	259
12	24	15	277.5s	51.4	77
12	26	2	30.3s	50.0	50
13	29	34	2639.1s	6.6	115
13	28	20	1121.5s	65.5	79

Table 6.9: Comparison of Benders decomposition with proposed method ($\gamma = 0.2$)

n	H	# of iter.	Time	Time saving (%)	Final # of constraints for MP
10	20	72	132.4s	58.8	238
11	23	76	590.5s	5.9	209
11	22.5	138	1275.3s	5.8	249
11	22	127	1159.0s	44.9	326
12	24	26	540.0s	5.4	97
12	26	2	27.7s	54.6	50
13	29	36	3391.0s		120
13	28	> 40	> 4000s		> 130

Table 6.10: Comparison of Benders decomposition with proposed method (Different initial feasible points)

n	H	# of iter.	Time	Time saving (%)	Final # of constraints for MP
10	20	141	298.0s	7.3	263
11	23	77	570.1	9.1	194
11	22.5	157	1302.5s	3.8	250
11	22	174	1655.0s	21.3	354
12	24	11	242.1s	57.6	72
12	26	2	28.1	53.7	50
13	29	30	2549.0s	9.8	112
13	28	> 40	> 4000s		> 130

of feasible points should be selected considering the trade-off between reduction in iteration numbers and total time of solving Problems 1–3, which is case-specific.

Then, we try two different values (0.1 and 0.5) for α in Problem 2 and the result is shown in Table 8. Note that α is the threshold for searching new feasible points. Increasing this value will find more feasible points. However, for the case of $\alpha = 0.5$, many new feasible points yield weak differences in λ, and these points do not improve computational efficiency compared to the case of $\alpha = 0.1$. In the comparison, α of 0.5 showed similar performance for $n = 13$, but inferior performance for the smallest-scale case, $n = 10$. A further investigation on adaptive way of adjusting α based on differences in λ would be beneficial in the future.

Next, the parameter of γ is changed. This value directly affects the number of constraints appended to MP. Here we decrease the value from 0.4 to 0.2 and its result is shown in Table 6.9. As γ is reduced, more constraints will be added to MP. Even the number of iterations is reduced, the total computation time may not be shortened because the computing time for MP is increased due to a large number of constraints. It indicates that excessive number of cuts is not a good choice.

Finally, random initial feasible points are also changed to see the sensitivity of the proposed approach. Its result is shown in Table 6.10. As with most optimization algorithms, the initial value plays an important role in total computing time. Inappropriate selection may slow down the proposed method. How to locate these initial feasible point is still an open question and worth further investigation.

6.5 Summary

In this chapter, a new computational strategy is provided to accelerate Benders decomposition, especially when only feasibility cuts are available. Since the line linking the infeasible and feasible points is supposed to cross the boundary of feasible region, the distance between the infeasible point and the intersection is employed to determine the tightest cutting plane. To avoid solving nonlinear programming, a bilinear optimization scheme is proposed and addressed efficiently. A classical scheduling problem is studied for comparison, and the proposed approach shows significant improvement in saving computational time and reducing the total number of iterations. This approach still has room for further improvement. A systematic guideline for choosing feasible points may improve performance of the proposed approach.

Chapter 7

Conclusion and Future Work

7.1 Major Contributions

Motivated by the great demands of computational abilities in high level process control activities, this thesis is concerned with developing computationally efficient framework to deal with the advanced process control, plant-wide dynamic optimization and large-scale mixed integer linear programming (MILP). The current optimization based methods, such as MPC and D-RTO, highly rely on the traditional DAE model and require considerable computing power for the online calculations. In some cases, especially for the large-scale nonlinear system with strong uncertainties, these methodologies may be computationally impractical. The approximate dynamic programming (ADP), well-developed in the computer science and operation researches (OR), is one of the most promising frameworks to alleviate the needs of computations and the "curse of dimensionality". Although it has achieved some success in the Artificial Intelligence (AI) and OR communities, very few applications are reported in the process control. Thus, one of the contributions of this work is to tailor the conventional ADP approach to suit the characteristics of the process control. Another work is focused on improving the current decomposition method to facilitate solving of MILP.

In Chapter 3, to guarantee the stability of the proposed control technology, a control Lyapunov function (CLF) design method is developed. Most of previous work on CLF design focus on the polynomial system, which is rare in the process control. The other general design methods, devoted to enlarging its region of attraction (ROA) and building upon the convex optimization, resulting in a large-scale linear/quadratic programming or linear matrix inequality (LMI) formulation. The proposed optimization formulation is mainly concerned with the stabilization of the user specified region under the input and state constraints, thus particularly suitable for the process control practice. An innovative coordinate search method is proposed to seek the sub-optimal solution for the non-convex and non-differentiable optimization problem. Furthermore, this scheme is also extended to the design of robust control Lyapunov function (RCLF).

In Chapter 4, the proposed RCLF is combined with the MPC and ADP to guarantee the stability. The ADP controller is constructed based on the historical operational data and the polynomial approximator is applied to approach the value function of the existing control policy. By means of the Bellman operator, the better control actions can be inferred through the value function both in the experienced region and their neighbourhoods. In the meantime, for the purpose of avoiding too aggressive regulations, the feasible region of the ADP is strictly characterized and the other areas in the state space are controlled by the MPC. As a result, a Lyapunov based mixed control strategy is presented to handle the continuous nonlinear system with uncertainties.

In Chapter 5, we presented the ADP based RTO. Usually, the classical DP style methods suffer from continuous variables and high-dimensional state space in the process control. On the contrary, the suggested scheme makes full use of the operational data to transfer the continuous and large-scale system to a discrete Markov chain model, which facilitates the application of DP. Though some previous paper have recognized the importance of incorporating the "risk" in the MDP framework, this study first designs the "risk state" artificially and provides the rigorous mathematical derivation to lead a tradeoff between the profits and risks, particularly useful to the process control.

In Chapter 6, we review the current decomposition methods for the MILP and point out that the key factor affecting the computational time is the quality and quantity of the cutting plane. In view of this, we proposed a systematic scheme to select the tightest constraint based on the points in the feasible region. Clearly, so called tightest constraints vary from the different feasible points. Hence, further investigations provide a new criterion to choose a number of tightest cutting planes, generated from various feasible points, and speed up the algorithm. The comparison results show that the new method overrides the conventional Benders decomposition and the sensitivity of parameters to the computational time is also checked.

7.2 Future Work

The work of this thesis explored the development and application of efficient optimization techniques in the process control. A number of challenges still remain and deserve further investigation in the future. Some possible directions are listed as follows:

- *Online learning scheme:* One of the merits of the ADP is its online learning scheme. The coefficient of the approximator can be updated according to the new arrival data. Theoretically, based on this scheme, the derived control policy can adjust to the change of the environment timely. However, without the variation of the environment, our experiments show that the performance may be degraded if the online learning takes the place of the batch learning. This phenomenon is not surprising because the batch learning employs the least square which approximates each state equally, whereas the online learning weights the current state more. As a result, the control law is always in favor of some particular states and ineffective in other states. Moreover, the convergence property of the online learning is still an open research problem. The unexpected fluctuation of the cost and production is not acceptable in the process industry. To enjoy the benefits of the online learning, it is necessary to develop more reliable weighting procedure and figure out the strict condition of the convergent learning.

- *Partially observed Markov decision process (POMDP).* Any ADP approaches suggested in this thesis assume that all states can be observed immediately. Unfortunately, this premise does not hold in most of practical cases. For instance, to know the concentration, the sample of the reactants should be sent to the analyzer and it usually takes a long time to get result. Moreover, some of states, are not even observable. Thus, the ADP for POMDP directs a way for the future research. Actually, in the process control research, people have already noticed this issue and several soft sensor methods are proposed including Kalman filter (KF), extended Kalman filter (EKF), particle filter, unscented Kalman filter (UKF). Once the unknown state is estimated, the problem can be solved as before. Another way to solve the POMDP stems from machine learning. Given the calibration data, the distribution of the unknown state can be derived. The Bellman optimal equation thus needs to modify the probability of the successive state by taking this distribution into account. It renders the computation more complicated even for a small size problem.

- *Further application of the proposed coordinate search method.* In chapter 4, a CLF/RCLF design framework is proposed to handle the non-convex, non-differentiable optimization. In essence, this method can be extended to the general medium sized minimax problem. A potential application is to solve the Hamilton-Jacobin-Bellman (HJB) equation approximately. Considering the HJB equation:

$$\frac{\partial J^*(x,t)}{\partial t} + \min_u (r(x,u) + \frac{J^*(x,t)}{\partial t} f(x,u)) = 0 \tag{7.1}$$

  subject to the system dynamics $\dot{x} = f(x,u)$ and boundary condition $J(x,T) = D(x)$. Since this partial differential equation (PDE) does not have the smooth solution in general, a well-designed approximation is acceptable for the optimal control. At beginning, the $J(x,t)$ can be parameterized by the linear approximation, for example, $J = [x,t]\omega[x,t]^T$. Then we try to minimize the Bellman residual:

$$\min_\omega \max_{x,t} \left| \frac{\partial J^*(x,t,\omega)}{\partial t} + \min_u (r(x,u) + \frac{J^*(x,t,\omega)}{\partial t} f(x,u)) \right| \tag{7.2}$$

$$\text{s.t.} \quad J(x,T,\omega) = D(x) \tag{7.3}$$

$$\dot{x} = f(x,t) \tag{7.4}$$

  However, this minimax formulation is very challenging to our coordinate search approach. Hence, further improvement should be considered to handle such problem.

- *Decomposition based MPC and ADP.* The effectiveness of the model based ADP highly depends on the structure of the model. Hence, exploring the individual structure of each model will be beneficial to the ADP. Note that the decomposition method just takes advantage of the sparse and factorable nature of the model, borrowing this idea into the ADP is also an attractive choice. The similar work has been done in the adaptive aggregation [11], in which the highly related states are clustered together. In addition, the plant-wide MPC can also make use of this decomposition method to simplify its computations [24, 23].

Bibliography

[1] T. Alamo, D.R. Ramirez, and E.F. Camacho D. Munoz de la Pena. Min-max MPC using a tractable QP problem. *Automatica*, 43:693–700, 2007.

[2] N. B. Almutairi and M. Zribi. Sliding mode control of coupled tanks. *MECHATRONICS*, 16:427–441, 2006.

[3] Z. Artstein. Stabilization with relaxed controls. *Nonlinear Analysis*, 7:1163–1173, 1983.

[4] J. Y. Audibert, R. Munos, and C. Szepesvari. Exploration-exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 19:1876–1902, 2009.

[5] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[6] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. Technical report, U. Mass. Amherst, 1993.

[7] R. E. Bellman. *Dynamic programming.* Princeton University Press, 1957.

[8] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[9] J. F. Benders. Partitioning methods for solving mixed variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[10] D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2005.

[11] D. P. Bertsekas and D. A. Castanon. Adaptive aggregation methods for inifite horizon dynamic programming. *IEEE Transaction on Automatic Control*, 34:589–598, 1989.

[12] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[13] L. T. Biegler, A. M. Cervantes, and A. Wachter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57:575–593, 2001.

[14] L.T. Biegler and V.M. Zavala. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers and Chemical Engineering*, 33:575–582, 2009.

[15] S. Binato, M. V. F. Pereira, and S. Granville. A new Benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems*, 16(2):235–240, 2001.

[16] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33:989–1007, 1985.

[17] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, 2000.

[18] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory studies in applied mathematics.* SIAM, 1994.

[19] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:1–3, 33–57, 1996.

[20] H. Chang, M. Fu, J. Hu, and S. Marcus. *Simulation-Based Algorithm for Markov Decision Processes.* Springer, 2007.

[21] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[22] Y. Chen, T. Edgar, and V. Manousiouthakis. On infinite-time nonlinear quadratic optimal control. *Syetems and Control Letters*, 51:259–268, 2004.

[23] R. Y. Cheng, J. F. Forbes, and W.S. Yip. Dantzigcwolfe decomposition and plant-wide MPC coordination. *Computers and Chemical Engineering*, 32:1507–1522, 2008.

[24] Y. R. Cheng. *Decomposition and Coordination of Large-scale Operations Optimization*. PhD thesis, University of Alberta, 2007.

[25] G. Cornuejols and C. Lemarechal. A convex-anlysis perspective on disjunctive cuts. *Math. Program., Ser. A*, 106:567–586, 2006.

[26] G. Cote and M. A. Laughton. Large-scale mixed integer programming: Benders-type heuristics. *European Journal of Operation Research*, 16:327–333, 1984.

[27] P.D. Couchman, M. Cannon, and B. Kouvaritakis. Stochastic MPC with inequality stability constraints. *Automatica*, 42:2169–2174, 2006.

[28] C. R. Cutler and R. T. Perry. Real time optimization with multivariable control is required to maximize profits. *Computers and Chemical Engineering*, 7:663–667, 1983.

[29] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[30] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13:492–498, 1967.

[31] J. J. Downs and E. F. Vogel. A plant-wide industrial process control problem. *Computers and Chemical Engineering*, 17:245–255, 1993.

[32] P. M. Duvall and J. B. Riggs. On-line optimization of the tennessee eastman challenge problem. *Journal of Process Control*, 10:19–33, 2000.

[33] N. H. El-Farra and P. D. Christofides. Bounded robust control of constrained multivariable nonlinear processes. *Chemical Engineering Science*, 58:3025–3047, 2003.

[34] M. Fischetti and D. Salvagnin. A note on the selection of Benders cuts. *Math. Program., Ser. B*, 124:175–182, 2010.

[35] M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of Benders cuts. *Math. Program., Ser. B*, 124:175–182, 2010.

[36] P. Frazier. *Knowledge-Gradient Methods for Statistical Learning*. PhD thesis, Princeton University, 2009.

[37] R. A. Freeman and P. V. Kokotovic. Design of softer robust nonlinear control laws. *Automatica*, 29:1425–1437, 1993.

[38] R. A. Freeman and P. V. Kokotovic. *Robust Nonlinear Control Design State-Space and Lyapunov Techniques*. Birkhauser, 1996.

[39] M. Galati. *Decomposition methods for integer linear programming*. PhD thesis, Lehigh University, 2010.

[40] P. Geibel and F. Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

[41] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10:237, 1972.

[42] P. E. Gill, W. Murray, and M. A. Saunders. User's guide for snopt: a fortran package for large-scale nonlinear programming. Technical report, University of California, 1998.

[43] M. Golshan, R. B. Boozarjomehry, and M. R. Pishvaie. A new approach to real time optimization of the Tennessee Eastman challenge problem. *Chemical Engineering Journal*, 112:33–44, 2005.

[44] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel. When nonlinear model predictive control is nonrobust. *Automatica*, 40:1729–1738, 2004.

[45] J. K. Gruber, D. R. Ramirez, T. Alamo, and E. F. Camacho. Min-max MPC based on an upper bound of the worst case cost with guaranteed stability. Application to a pilot plant. *Journal of Process Control*, 21:194–204, 2011.

[46] R. Guder, M. Dellnitz, and E. Kreuzer. An adaptive method for the approximation of the generalized cell mapping. *Chaos, Solitons and Fractals*, 8:525–534, 1997.

[47] V. Havlena and J. Lu. A distributed automation framework for plant-wide control, optimisation. In *16 IFAC World Congress*, 2005.

[48] C. S. Hsu. *Cell-to-cell mapping: A method for global analysis for nonlinear systems*. Springer-Verlag, 1987.

[49] G. S. Huang and A. L. Dexte. Realization of robust nonlinear model predictive control by offline optimisation. *Journal of Process Control*, 18:431–438, 2008.

[50] K. S. Hwang, S. W. Tan, and M. C. Tsai. Reinforcement learning to adaptive control of nonlinear systems. *IEEE Transactions On Systems, Man, And CyberneticsPart B: Cybernetics*, 33:514–521, 2003.

[51] M. G. Ierapetritou and C. A. Floudas. Effective continuous-time formulation for short-term scheduling: 1. Multipurpose batch processes. *Industrial and Engineering Chemistry Research*, 37(11):4341–4359, 1998.

[52] A. Jadbabaie and J. Hauser. Control of a thrust-vectored flying wing: a receding horizon-LPV approach. *International Journal of Robust and Nonlinear control*, 12:869–896, 2002.

[53] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding-horizon control of nonlinear systems. *IEEE Transaction on Automatic Control*, 46:776–783, 2001.

[54] T. Jockenhovel, L. T. Biegler, and A. Wachter. Dynamic optimization of the Tennessee Eastman process using the OptControlCentre. *Computers and Chemical Engineering*, 27:1513–1531, 2003.

[55] Tor A. Johansen. Computation of Lyapunov functions for smooth nonlinear systems using convex optimization. *Automatica*, 36:1617–1626, 2000.

[56] J. V. Kadam, M. Schlegel, B. Srinivasan, D. Bonvin, and W. Marquardt. Dynamic optimization in the presence of uncertainty: From off-line nominal solution to measurement-based implementation. *Journal of Process Control*, 17:389–398, 2007.

[57] P. Kall and S. W. Wallace. *Stochastic Programming.* John Wiley & Sons, 1994.

[58] S. S. Keerthi and E. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximation. *Journal of Optimization Theory and Applications*, 57:265–293, 1988.

[59] H. K. Khalil. *Nonlinear Systems.* Macmillan, New York, 1992.

[60] K. C. Kiwiel. A direct method of linearization for continuous minimax problems. *Journal of Optimization Theory and Applications*, 55:271–287, 1987.

[61] T. Kohonen. *Self-organizing Maps*. Springer, 2001.

[62] P.V. Kokotovic. The joy of feedback: nonlinear and adaptive. *Control Systems Magazine*, 12:7–17, 1992.

[63] M. V. Kothare, V. Balakrishnans, and M. Moraris. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379, 1996.

[64] M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic. *Nonlinear and Adaptive Control Design*. John Wiley & Sons, New York, 1995.

[65] W. Marquardt L. Wurth, R. Hannemann. Neighboring-extremal updates for nonlinear model-predictive control and dynamic real-time optimization. *Journal of Process Control*, 19:1277–1288, 2009.

[66] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[67] W. Langsona, I. Chryssochoosb, S. V. Rakovi, and D. Q. Mayneb. Robust model predictive control using tubes. *Automatica*, 40:125–133, 2004.

[68] M. Lazar and W. P. M. H. Heemels. Predictive control of hybrid systems: Input-to-state stability results for sub-optimal solutions. *Automatica*, 45:180–185, 2009.

[69] J. H. Lee and J. M. Lee. Approximate dynamic programming based approach to process control and scheduling. *Computers and Chemical Engineering*, 30:1603–1618, 2006.

[70] J. H. Lee and Z. Yu. Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33:763–781, 1997.

[71] J. M. Lee, N. S. Kaisare, and J. H. Lee. Choice of approximator and design of penalty function for an approximate dynamic programming based control approach. *Journal of Process Control*, 16:135–156, 2006.

[72] J. M. Lee and J. H. Lee. An approximate dynamic programming based approach to dual adaptive control. *Journal of Process Control*, 19:859–864, 2009.

[73] L.Grne. *Asymptotic Behavior of Dynamical and Control Systems under Perturbation and Discretization*. SpringerCVerlag, 2002.

[74] P. Li, M. Wendt, and G. Wozny. Robust model predictive control under chance constraints. *Computers and Chemical Engineering*, 24:829–834, 2000.

[75] D. Limon, T. Alamo, F. Salas, and E. F. Camacho. Input to state stability of min-max MPC controllers for nonlinear systems with bounded uncertainties. *Automatica*, 42:797–803, 2006.

[76] Y. Lin and E. D. Sontag. A universal formula for stabilization with bounded controls. *Systems and Control Letters*, 19:393–397, 1991.

[77] T. L. Magnanti and R. T. Wong. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. *Operations Research*, 29:464–484, 1981.

[78] L. Magni and R. Sepulchre. Stability margins of nonlinear receding horizon control via inverse optimality. *Systems and Control Letters*, 32:241–245, 1997.

[79] T. E. Marlin and A. N. Hrymak. Real-time operations optimization of continuous processes. In *Chemical Process Control-V Conference*, 1996.

[80] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[81] D. Megias, J. Serrano, and C. de Parda. Min-max constrained quasi-infinite horizon model predictive control using linear programming. *Journal of Process Control*, 12:495–505, 2002.

[82] M. Mercangoz and F. J. Doyle III. Real-time optimization of the pulp mill benchmark problem. *Computers and Chemical Engineering*, MeD:08:789–804, 2008.

[83] P. Mhaskar, N. H. El-Farra, and P. D. Christofides. Predictive control of switched nonlinear systems with scheduled mode transitions. *IEEE Transactions On Automatic Control*, 50:1670–1680, 2005.

[84] P. Mhaskar, N. H. El-Farra, and P. D. Christofides. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Systems and Control Letters*, 55:650–659, 2006.

[85] I. Miletic and T. E. Marlin. On-line statistical results analysis in real-time operations optimization. *Industrial and Engineering Chemistry Research*, 37:3670–3684, 1998.

[86] B. J. Nelson. *Multivariate state space mapping models for process and quality improvement*. PhD thesis, Arizona State University, 2000.

[87] G. D. Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time-varying systems. *IEEE Transactions On Automatic Control*, 43:1030–1036, 1998.

[88] H. Nosair, Y. Yang, and J. M. Lee. Min-max control using parametric approximate dynamic programming. *Computers and Chemical Engineering*, 18:190–197, 2010.

[89] C. Oguz and M. A. Gallivan. Optimization of a thin film process using a dynamic model extracted from molecular simulations. *Automatica*, 44:1958–1969, 2008.

[90] G. R. Wilson P. Wu, J. C. Hartman. A demand-shifting feasibility algorithm for benders decomposition. *European Journal of Operation Research*, 148:570–583, 2003.

[91] A. Papachristodoulou and S. Prajna. *Positive Polynomials in Control*. Springer, 2005.

[92] M. Pereira and L.Pinto. Multistage stochastic optimization applied to energy planning. *Mathematical Programming*, 52:359–375, 1991.

[93] C. A. Poojari and J. E. Beasley. Improving Benders decomposition using a genetic algorithm. *European Journal of Operation Research*, 199:89–97, 2009.

[94] N. K. Poulsen, B. Kouvaritakis, and M.Cannon. Constrained predictive control and its application to a coupled-tanks apparatus. *International Journal of Control*, 74:552–564, 2001.

[95] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23 rd International Conference on Machine Learning,*, 2006.

[96] W. B. Powell. *Approximate Dynamic Programming.* Wiley, 2007.

[97] S. Prajna, P. A. Parrilo, and A. Rantzer. Nonlinear control synthesis by convex optimization. *IEEE Transaction on Automatic Control*, 49:310–314, 2004.

[98] M. L. Puterman. *Markov Decision Processes.* Wiley, 1994.

[99] A. Rantzer. A dual to Lyapunov's stability theorem. *Systems and Control Letters*, 42:161–168, 2001.

[100] W. Rei, J. F. Cordeau, M. Gendreau, and P.Soriano. Accelerating benders decomposition by local branching. *INFORMS Journal on Computing*, 21:333–345, 2009.

[101] N. L. Ricker and J. H. Lee. Nonlinear model-predictive control of the Tennessee-Eastman challenge process. *Computers and Chemical Engineering*, 19:961–981, 1995.

[102] G. Rummery and M. Niranjan. On-line Q-learning using Connectionist systems. Technical report, University of Cambridge, 1994.

[103] B. Rustem, S. Zakovic, and P. Parpas. An interior point algorithm for continuous minimax: implementation and computation. *Optimization Methods& Software*, 23:911–928, 2008.

[104] G. K. D. Saharidis and M. G. Ierapetritou. Improving Benders decomposition using maximum feasible subsystem (MFS) cut generation strategy. *Computers and Chemical Engineering*, 34:1237–1245, 2010.

[105] G. K. D. Saharidis, M. Minoux, and M. G. Ierapetritou. Accelerating Benders method using covering cut bundle generation. *International Transaction in Operational Research*, 17(2):221–237, 2010.

[106] N. V. Sahinidis and I. E. Grossmann. Convergence properties of generalized Benders decomposition. *Computers and Chemical Engineering*, 15:481–491, 1991.

[107] A. T. Schwarm and M. Nikolaou. Chance-constrained model predictive control. *AICHE Journal*, 45:1743–1752, 1999.

[108] D. E. Seborg, T. F. Edagar, and D. A. Mellichamp. *Process Dynamics and Control*. Wiley, 2003.

[109] J. Si, A. G. Barto, and W. B. Powell. *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE, 2004.

[110] A. Singh, J. F. Forbes, P. J. Vermeer, and S. S. Woo. Model-based real-time optimization of automotive gasoline blending operations. *Journal of Process Control*, 10:43–58, 2000.

[111] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[112] R. V. Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17:638–663, 1969.

[113] E. D. Sontag. A ‘universal’ construction of Artein’s theorem on nonlinear stability. *Systems and Control Letters*, 13:117–123, 1989.

[114] R. S. Sutton. Integrating architectures for learning, planning, and reacting based on approximating dynamic programming. In *7th International Conference on Machine Learning*, 1990.

[115] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. The MIT Press, 1998.

[116] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.

[117] M. Sznaier, R. Suarez, and J. Cloutier. Suboptimal control of constrained nonlinear system via receding horizon constrained control Lyapunov functions. *International Journal of Robust and Nonlinear Control*, 13:247–259, 2003.

[118] W. Tan. *Nonlinear Control Analysis and Synthesis using Sum-of-Squares Programming.* PhD thesis, University of California, Berkeley, 2006.

[119] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.

[120] U. Topcu, A. Parkard, and P. Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44:2669–2675, 2008.

[121] T. Tosukhowong, J. M. Lee, J. H. Lee, and J. Lu. An introduction to a dynamic plant-wide optimization stradegy for an integrated plant. *Computers and Chemical Engineering*, 29:199–208, 2004.

[122] A. Toumi, M. Diehl, S. Engell, H. G. Bock, and J. P. Schloder. Finite horizon optimizing control of advanced SMB chromatographic processes. In *16th IFAC World Congress*, 2005.

[123] J. W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, Reading, MA., 1977.

[124] A. Wachter. *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering.* PhD thesis, Carnegie Mellon University, 2002.

[125] Z. Y. Wan and M. V. Kothare. An efficient off-line formulation of robust model predictive control using linear matrix inequalities. *Automatica*, 39:837–846, 2003.

[126] Z.Y. Wan and M. V. Kothare. Efficient robust constrained model predictive control with a time varying terminal constraint set. *Systems and Control Letters*, 48:375–383, 2003.

[127] C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, Cambridge University, 1989.

[128] S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917, 2006.

[129] Y. Yang and J. M. Lee. Design of a control lyapunov function for stabilizing specified states. In *Proc. of the 8th International Symposium on Dynamics and Control of Process Systems (DYCOPS)*, 2010.

[130] Y. Yang and J. M. Lee. Probabilistic modeling and dynamic optimization for performance improvement and risk management of plant-wide operation. *Computers and Chemical Engineering*, 34:567–579, 2010.

[131] G. Zakeri, A.B. Philpott, and D. M. Ryan. Inexact cuts in Benders decomposition. *SIAM Journal of Optimization*, 10:643–657, 1998.

[132] A. C. Zanin, M. Tvrzská de Gouvêa, and D. Odloak. Industrial implementation of a real-time optimization strategy for maximizing production of LPG in a FCC unit. *Computers and Chemical Engineering*, 15:525–531, 2000.

[133] Y. Zhang, D. Monder, and J. F. Forbes. Real-time optimization under parametric uncertainty: a probability constrained approach. *Journal of Process Control*, 12:373–389, 2002.

[134] J. H. Zhong, D. Z. Cheng, and X. M. Hu. Constructive stabilization for quadratic input nonlinear systems. *Automatica*, 44:1996–2005, 2008.