

Distributed model predictive control of integrated process systems using an
adaptive community detection approach

by

Amirmohammad Ebrahimi

B.S., Sharif University of Technology, 2017

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Tim Taylor Department of Chemical Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Davood B. Pourkargar
Assistant Professor
Tim Taylor Dept. of Chemical Engineering

Copyright

© Amirmohammad Ebrahimi 2024.

Abstract

Integrated processes offer significant advantages in chemical manufacturing, but their complexity poses challenges for control and operation. Model predictive control (MPC) shows promise in addressing these control challenges, but its practical implementation is hindered by the need for real-time optimization. Distributed Model Predictive Control (DMPC) is a viable alternative to MPC. It divides the complex process into smaller, interconnected communities and assigns a control agent to each. These agents collaborate to optimize their respective communities and share information, leading to a more manageable and efficient control strategy. This master's report thoroughly examines the relevant literature on integrated processes, distributed control and estimation, network theory, and community detection. It highlights the superiority of distributed control and estimation over centralized control, particularly when applied to unweighted networks. Furthermore, the report extends the existing research by exploring weighted graphs, generalizing state-space models to non-control affine models, and proposing novel improvements to community detection approaches. These contributions address gaps in the current literature and enhance the applicability of DMPC in real-world scenarios.

Table of Contents

List of Figures	vi
List of Tables	viii
Acknowledgements	ix
Dedication	x
1 Introduction	1
2 Literature review	4
2.1 Integrated and intensified process	4
2.2 Control and Distributed Control	5
2.2.1 From Decentralized control to proportional-integral-derivative (PID)	5
2.2.2 Centralized Control	6
2.2.3 Distributed Control	7
2.3 Preliminaries	7
2.3.1 Notation	7
2.3.2 Process Model Formulation	8
2.4 Lyapunov Base Control	9
2.5 Continuous MPC Formulation	10
2.6 DMPC	13
2.6.1 Non-cooperative DMPC	13
2.6.2 Cooperative DMPC	14
2.7 Community detection	15

2.7.1	Modularity based methods	16
2.7.2	Spectral clustering methods	22
2.8	Comparing Community Detection Methods: An Integrated Process Control Perspective	26
3	Spectral community detection approach	28
3.1	Weighted Adjacency Matrix	28
3.2	Community Detection Algorithm	31
3.3	Comparative Study: Enhanced Spectral Community Detection Approach vs. Traditional Methods	36
3.4	Identifying the Optimal Number of Communities: Elbow method	38
4	Case Study: An Integrated Process System	39
4.1	Process Description	39
4.2	Distributed Control Structure	41
5	Simulation Result	46
6	Conclusion	52
	Bibliography	53
A	Process Model Description	66

List of Figures

2.1	A schematic of the predictive control at time t	12
3.1	Small network section representing communication in the entire system. . . .	29
3.2	Community detection index results for 20 different random graphs.	37
4.1	Integrated benzene alkylation process.	40
4.2	Multi-agent distributed control architectures for (a) $R = 1$ and (b) $R = 0.001$	44
5.1	Selected temporal profiles of the state variables under DMPC based on the decomposition obtained weighted graph representation (for $R = 1$) and unweighted graph representation: (a) holdup in the first CSTR, (b) temperature in fourth CSTR, (c) temperature in the separator, and (d) concentration of desired product in the third CSTR.	47
5.2	Temporal profiles of selected manipulated inputs under DMPC based on the decomposition obtained from weighted graph representation (for $R = 1$) and unweighted graph representation: (a) effluent flow of separator, (b) outlet flow from first CSTR to second CSTR, (c) heat to the fourth CSTR, and (d) feed to first CSTR.	48
5.3	Temporal profiles of selected state variables under DMPC based on the decomposition obtained from weighted graph representation (for $R = 0.001$) and unweighted graph representation: (a) holdup in the third CSTR, (b) temperature in third CSTR, (c) concentration of desire product in the separator, and (d) concentration of desired product in the third CSTR.	49

- 5.4 Temporal profiles of selected manipulated inputs under DMPC based on the decomposition obtained from weighted graph representation (for $R = 0.001$) and unweighted graph representation: (a) effluent flow of separator, (b) feed to first CSTR, (c) heat to the separator, and (d) effluent flow of first CSTR. 50

List of Tables

3.1	Definition of edge-weights at a steady state operation point.	31
3.2	Φ value for different method.	38
4.1	Steady state values of the state variables.	42
4.2	Steady state values of the manipulated inputs.	43
4.3	Distributed control structures.	45
5.1	Closed-loop performance for $R = 1$	51
5.2	Closed-loop performance for $R = 0.001$	51
A.1	Variables and their definitions in the process flow diagram.	68
A.2	Process Parameters	70

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. Pourkargar, for his guidance and support throughout this research. I am also thankful to my colleagues for their valuable feedback and to my family for their unwavering support.

Dedication

This thesis is dedicated to my parents, whose endless love and encouragement have been my greatest source of strength.

Chapter 1

Introduction

The widespread adoption of integrated processes in chemical manufacturing is designed to lower expenses, optimize resource usage, and improve overall effectiveness, especially by reducing material waste and energy usage. Nonetheless, this integration poses significant operational and control obstacles.^{1;2} The complexities are further magnified due to the non-linear behaviors of the individual components and their interconnected integration and recycling streams. These factors contribute to the heightened complexity of the overall process network.³ Model predictive control (MPC) is a complex yet effective method for real-time decision-making, providing significant advantages over traditional control strategies.⁴ MPC structures the control problem as a dynamic optimization problem, requiring a real-time solution of underlying optimization while considering constraints on state variables and manipulated inputs.^{5;6} These constraints and the complexity of the optimization can lead to substantial computational requirements, making the implementation of centralized MPC computationally expensive. This computational burden can cause significant delays in highly integrated processes⁶⁻⁸, leading to decreased production efficiency and profit loss due to the lag in controller response.

Distributed model predictive control (DMPC) emerges as a promising alternative to centralized MPC, providing faster computation through parallelization while maintaining similar closed-loop performance. In contrast to centralized MPC, which relies on a single controller,

DMPC utilizes multiple controllers that collaborate to solve the dynamic optimization problem in a distributed manner. The initial step in implementing DMPC involves determining the distributed control architecture. This encompasses identifying the ideal number of distinct subsystems and appropriately distributing manipulated inputs, controlled outputs, state variables, and system constraints among them. Optimal decomposition is essential as it minimizes the necessary communication between different controllers. Ideally, when the control system is divided into P subsystems, each controller can operate independently with minimal external influence.⁹

The study of community detection, which aims to find optimal network decomposition, has garnered significant attention. Community detection involves identifying subsystems with higher internal connectivity compared to their connections with other subsystems¹⁰. The effectiveness of community detection algorithms is quantitatively assessed using a metric known as modularity. Modularity measures the proportion of edges (or their weights) within communities compared to what would be expected in a randomized graph with the same degree distribution, essentially evaluating how well the identified communities are defined. Maximizing modularity involves reducing connections between different subsystems while enhancing connections within each subsystem⁹.

A recursive solution to the underlying dynamic optimization problem for MPC requires frequent access to all system state variables. Although prior research often assumes this condition, real-world situations frequently limit the ability to measure all state variables. Moving horizon estimation (MHE) has emerged as a promising solution for estimating values that cannot be directly measured. MHE leverages historical data and is structured as a dynamic optimization problem, similar to MPC^{11;12}. Likewise, solving MHE optimization problems, particularly for large-scale and highly integrated processes, also incurs substantial computational expenses. Delays in MHE calculation can negatively impact the performance of closed-loop control systems because MPC relies on comprehensive knowledge of all states to make effective control decisions. A strategy akin to the DMPC method, known as distributed moving horizon estimation (DMHE), is utilized to tackle the computational challenges of centralized estimation. This approach involves multiple estimation agents that

collaborate and share information among themselves¹³⁻¹⁵.

So far, the effectiveness of DMPC based on the unweighted networks over centralized MPC has been proven¹⁵⁻¹⁷. In this thesis, a comprehensive literature review on the concept of integrated processes, distributed control and estimation, community detection, and process networks has been conducted. Then, in the results section, we analyze the effect of interactions among network components on the optimal decomposition of control and estimation problems at various operational points. Specifically, we investigate the development of multi-agent distributed architectures for an integrated estimation and control framework and assess the computational efficiency of the adaptive network decomposition approach. We employ an energy-based spectral community detection approach¹⁸, which is computationally efficient and more reliable than traditional methods¹⁹. We apply this decomposition method to develop distributed architectures for multi-agent DMHE and DMPC of a benchmark benzene alkylation process. Closed-loop performance and computational requirements are evaluated under various operational conditions. Simulation outcomes demonstrate the effectiveness of the proposed adaptive community detection-based distributed estimation and control approach in regulating key benzene alkylation process components. The simulation studies also show how optimal decomposition varies with different operating conditions.

The structure of the thesis is outlined as follows: Chapter 2 offers a review of the literature; Chapter 3 addresses the formulation of MPC, MHE, DMPC, and DMHE; Chapter 4 explores the construction of weighted control and estimation networks; Chapter 5 introduces an enhanced approach to community detection; and Chapter 6 presents the study’s findings.

Chapter 2

Literature review

2.1 Integrated and intensified process

Chemical engineering’s integrated processes represent a sophisticated approach to manufacturing, merging diverse operational units and production stages into cohesive systems. This methodology evolved to boost efficiency, reduce waste, reinforce sustainability, and enhance overall system performance. Typically integrated processes feature interconnected elements, such as multi-stage reaction and separation units, heat exchange networks, varied reactor designs, and material recycling pathways. The concept gained traction during the chemical industry’s maturation in the latter 20th century, as manufacturers sought economies of scale through larger plant designs, particularly for bulk chemicals and petrochemicals. The 1970s marked a turning point, with energy scarcity, rising costs, and stricter environmental mandates catalyzing more aggressive integration strategies. Contemporary integrated processes emphasize material recycling to conserve raw materials and reduce emissions while prioritizing heat recovery for improved energy efficiency. These advanced integration techniques are now standard practice in chemical and petrochemical facilities globally, signifying a paradigm shift towards more sustainable and economically viable production methods.^{1,3} Process intensification offers an alternative approach to enhancing process efficiency. This method prioritizes the optimization of driving forces, aiming to ensure that intrinsic reaction rates,

rather than transport or transfer phenomena, dictate process dynamics. The key strategy involves operating at reduced spatial scales compared to traditional processes. This approach is predicated on the assumption that scaling up production can be achieved by increasing the number of parallel-operating intensified units rather than enlarging individual units. By focusing on these smaller-scale, high-efficiency modules, process intensification seeks to unlock performance levels that surpass those attainable through conventional methods, potentially revolutionizing chemical engineering practices.^{20;21} Based on the aforementioned explanation, the following definition has been provided in¹. Process intensification encompasses any advancement in chemical engineering that results in markedly more compact, environmentally friendly, safer, and energy-efficient technologies. This concept also extends to innovations that consolidate multiple operations into fewer devices or even a single apparatus.

2.2 Control and Distributed Control

2.2.1 From Decentralized control to proportional-integral-derivative (PID)

Chemical process operations continually grapple with the imperatives of safety, environmental sustainability, and profitability, leading to an extensive reliance on automated control systems. Traditional process control mechanisms, exemplified by proportional-integral-derivative (PID) control, operate by measuring a single process output variable—such as temperature, pressure, level, or product concentration—to determine the necessary control action for implementation via a control actuator. This approach aims to maintain the output variable at a predetermined set point. PID controllers have a proven track record of success in chemical process control and are expected to remain integral to process industries.²²

However, decentralized control systems, including those based on multiple single-loop PID controllers, face limitations. They fail to account for interactions between plant components (subsystems) and control loops, potentially constraining the achievable closed-loop performance. Recognizing these challenges, significant research efforts have been directed toward

developing tools for optimal input/output pair selection and PID controller tuning. These advancements aim to enhance control system efficacy, addressing the complexities of modern chemical processes and the intricate interplay between various system components.²³⁻²⁵ Seesaw algorithms enhance PID controllers with adaptive capabilities by leveraging real-time error and its derivative, resulting in Seesaw APID controllers that outperform conventional PIDs in terms of response time, stability, and error reduction while maintaining operational simplicity.²⁶ A novel approach to PID high-type controller tuning, utilizing Linear Quadratic Regulator (LQR) and pole-placement techniques while extending the Linear Extended State Observer (LESO) design to address modeling errors and uncertainties. The proposed PID high-type controller demonstrates superior performance over traditional PD and PID type-i controllers, offering improved high-order steady-state error elimination, high-precision tracking, enhanced disturbance suppression, and better dynamic response characteristics such as reduced rise and settling times.²⁷

2.2.2 Centralized Control

The limitations of decentralized control loops in directly addressing multivariable interactions have been a significant driving force behind the development of model-based centralized control architectures. Despite the numerous powerful tools available for implementing, quantifying, and optimizing decentralized control loop interactions, this shortcoming spurred the creation of various centralized control methods. These include linear pole placement, linear optimal control, and linear model predictive control (MPC), all aiming to better handle complex system interactions. This centralized approach employs a single controller to calculate control actions for the entire system. Initially developed for linear systems, it was later expanded to accommodate nonlinear systems as well. This method allows for comprehensive management of system-wide interactions and dynamics through a unified control strategy.^{26;28-32}

2.2.3 Distributed Control

Using integrated processes with complex dynamic behavior often results in substantial time delays when implementing real-time, centralized control solutions. This can lead to resource waste and a reduction in the process's overall efficiency.^{7;8;33} Distributed control offers an alternative to centralized control systems, employing multiple controllers that perform individual calculations and communicate results to achieve closed-loop objectives. A powerful approach within this framework is distributed model predictive control (DMPC), which leverages a distributed architecture to address two key issues: (1) solving the computation time problem associated with centralized Model Predictive Control (MPC), and (2) effectively managing challenges posed by integrated processes with complex, dynamic behavior. These processes often lead to substantial time delays in real-time, centralized control implementations, potentially resulting in resource waste and reduced overall process efficiency. By utilizing a distributed approach, DMPC aims to improve system responsiveness and efficiency, mitigating the drawbacks of centralized control in complex, dynamic systems.³⁴ A series of review papers explain distributed MPC.³⁵⁻³⁸ In the rest of this chapter, we will further explore distributed model predictive control (DMPC) and examine its key properties.

2.3 Preliminaries

2.3.1 Notation

We denote the Euclidean norm by $\|\cdot\|$ and the squared weighted Euclidean norm by $\|\cdot\|_Q^2$, where $\|x\|_Q^2 = x^T Q x$ for $x \in \mathbb{R}^n$. A class $\mathcal{K}$ function $\alpha : [0, a) \rightarrow [0, \infty)$ is continuous, strictly increasing, and satisfies $\alpha(0) = 0$, while a class $\mathcal{KL}$ function $\beta(r, s)$ is class $\mathcal{K}$ in r for fixed s and decreasing in s with $\lim_{s \rightarrow \infty} \beta(r, s) = 0$ for fixed r . We define $\Omega_r := \{x \in \mathbb{R}^n : V(x) \leq r\}$ for a positive definite, continuously differentiable V with $V(0) = 0$, and use $\setminus$ for set subtraction: $A \setminus B := \{x \in A : x \notin B\}$. $\text{diag}(v)$ is a diagonal matrix with elements of vector v , $\oplus$ denotes the Minkowski sum, t_0 is the initial time, $\{t_k \geq 0\}$ represents synchronous time instants with $t_k = t_0 + k\delta$ and $t_{k+i} = t_k + i\delta$ (δ fixed, i integer), and $\{t_a \geq 0\}$ denotes

asynchronous time instants with variable intervals.

2.3.2 Process Model Formulation

Let us examine a set of ordinary differential equations (ODEs) that encapsulate the nonlinear dynamic behavior of an integrated process system in a general state space form:

$$\begin{aligned}\dot{x}(t) &= f(x(t)) + g(x(t), u(t)), \\ y(t) &= h(x(t)).\end{aligned}\tag{2.1}$$

Remark 1. *While functions f and g are presented separately to elucidate the derivation process, the model maintains its general nonlinear, non-control affine nature.*

In this formulation:

- $x(t) = [x_1(t), \dots, x_n(t)]^T \in \mathbb{R}^n$ denotes the state vector,
- $u(t) = [u_1(t), \dots, u_m(t)]^T \in \mathbb{R}^m$ represents the manipulated input vector,
- $y(t) = [y_1(t), \dots, y_l(t)]^T \in \mathbb{R}^l$ signifies the output vector.

The variable t indicates time, while $f, g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $h : \mathbb{R}^n \rightarrow \mathbb{R}^l$ are smooth nonlinear vector functions characterizing the system dynamics and output, respectively. Consider the system described by Equation 2.1, where the manipulated inputs, outputs, and state variables can be decomposed into p interconnected subsystems. Each subsystem is characterized by the following submodel:

$$\begin{aligned}\dot{x}_i(t) &= f_i(x(t)) + g_i(x(t), u_i(t)), \quad i = 1, 2, \dots, p, \\ y_i(t) &= h_i(x(t)).\end{aligned}\tag{2.2}$$

The composite vectors for the entire system are defined as:

$$\begin{aligned} x &= [x_{(1)}^T \quad x_{(2)}^T \quad \cdots \quad x_p^T]^T, \\ u &= [u_{(1)}^T \quad u_{(2)}^T \quad \cdots \quad u_p^T]^T, \\ y &= [y_{(1)}^T \quad y_{(2)}^T \quad \cdots \quad y_p^T]^T, \end{aligned} \tag{2.3}$$

with corresponding functions:

$$\begin{aligned} f &= [f_{(1)}^T \quad f_{(2)}^T \quad \cdots \quad f_p^T]^T, \\ g &= [g_{(1)}^T \quad g_{(2)}^T \quad \cdots \quad g_p^T]^T, \\ h &= [h_{(1)}^T \quad h_{(2)}^T \quad \cdots \quad h_p^T]^T. \end{aligned} \tag{2.4}$$

Here, $f_{(i)} : \mathbb{R}^n \times \mathbb{R}^{m_i} \times \mathbb{R}^+ \rightarrow \mathbb{R}^{n_i}$ and $g_{(i)} : \mathbb{R}^n \times \mathbb{R}^{m_i} \times \mathbb{R}^+ \rightarrow \mathbb{R}^{l_i}$ represent the relevant components of f and g for the i^{th} subsystem model. The state variables, manipulated inputs, and output variables of each subsystem are defined as:

$$\begin{aligned} x_{(i)} &= [x_{(i),1}^T \quad x_{(i),2}^T \quad \cdots \quad x_{(i),n_i}^T]^T, \\ u_{(i)} &= [u_{(i),1}^T \quad u_{(i),2}^T \quad \cdots \quad u_{(i),m_i}^T]^T, \\ y_{(i)} &= [y_{(i),1}^T \quad y_{(i),2}^T \quad \cdots \quad y_{(i),l_i}^T]^T, \end{aligned} \tag{2.5}$$

where $n = \sum_{i=1}^p n_i$, $m = \sum_{i=1}^p m_i$, and $l = \sum_{i=1}^p l_i$.

2.4 Lyapunov Base Control

In DMPC architectures, Lyapunov-based control is crucial for determining closed-loop system stability regions.^{28;39;40} We postulate the existence of a locally Lipschitz control law $h(x) = [h_1(x), \dots, h_m(x)]^T$ with $u_i = h_i(x)$, $i = 1, \dots, m$, which:

1. Ensures asymptotic stability of the origin for the nominal closed-loop system (Eq. (2) with $u_i = h_i(x)$, $i = 1, \dots, m$, and $w = 0$)

2. Satisfies input constraints for all states x within a specified stability region

Applying converse Lyapunov theorems, this assumption implies the existence of class $\mathcal{K}$ functions $\alpha_i(\cdot)$, $i = 1, 2, 3$, and a continuously differentiable Lyapunov function $V(x)$ for the nominal closed-loop system, satisfying:

$$\begin{aligned} \alpha_1(|x|) &\leq V(x) \leq \alpha_2(|x|) \\ \frac{\partial V(x)}{\partial x} [f(x(t)) + g(x(t), u(t))] &\leq -\alpha_3(|x|) \\ h_i(x) &\in U_i, \quad i = 1, \dots, m \end{aligned} \tag{2.6}$$

for all $x \in \mathcal{O} \subseteq \mathbb{R}^n$, where $\mathcal{O}$ is an open neighborhood of the origin. We denote $\Omega \subseteq \mathcal{O}$ as the stability region of the nominal closed-loop system under the Lyapunov-based controller $h(x)$. Typically, Ω is a level set of the Lyapunov function $V(x)$: $[\Omega := x \in \mathbb{R}^n : V(x) \leq \gamma]$

While Lyapunov-based control provides stability guarantees control, determining the optimal function that satisfies Equation 2.6 remains challenging. For a thorough examination of the most effective functions, readers are directed to the comprehensive survey presented in the review paper by Zhou et al.⁴¹

2.5 Continuous MPC Formulation

Let us initially consider that all state variables of the system described in Equation 2.1 are measurable at each sampling instant. However, we will later relax this assumption to align the control process more closely with real-world scenarios. Under these conditions, we can formulate the MPC problem as a constrained dynamic optimization problem:

$$\min_u \int_{t_k}^{t_k + N_c T} (\tilde{y}^T P \tilde{y} + \tilde{u}^T W \tilde{u}) dt, \tag{2.7}$$

subject to the system constraints:

$$\begin{aligned}
\dot{x} &= f(x(t)) + g(x(t), u(t)), \\
y &= h(x(t)), \\
u^{\min} &\leq u \leq u^{\max}, \\
\xi(x, u, t) &\leq 0, \\
\phi(x, u, t) &= 0,
\end{aligned} \tag{2.8}$$

where:

- T is the sampling period,
- t_k represents the k^{th} sampling time instance,
- N_c is the number of sampling times within the control horizon,
- $\tilde{y}$ and $\tilde{u}$ represent the deviations from the steady state output and manipulated input values, respectively.

The steady state manipulated inputs and desired controlled outputs are denoted by:

$$\begin{aligned}
u^{\text{ref}}(t) &= [u_1^{\text{ref}}(t) \quad u_2^{\text{ref}}(t) \quad \cdots \quad u_m^{\text{ref}}(t)]^T \in \mathbb{R}^m, \\
y^{\text{ref}}(t) &= [y_1^{\text{ref}}(t) \quad y_2^{\text{ref}}(t) \quad \cdots \quad y_l^{\text{ref}}(t)]^T \in \mathbb{R}^l.
\end{aligned} \tag{2.9}$$

The objective function incorporates two positive definite matrices:

- $P \in \mathbb{R}^{l \times l}$: penalizes the impact of output regulation errors,
- $W \in \mathbb{R}^{m \times m}$: penalizes the impact of manipulated inputs.

The manipulated inputs are bounded by $u^{\min} \in \mathbb{R}^m$ and $u^{\max} \in \mathbb{R}^m$. The system's inequality and equality constraints are specified using the generic nonlinear vector functions ξ and ϕ , respectively.

At each sampling instant t_k , the constrained nonlinear dynamic optimization problem, formulated in Equation 2.8 is solved over the control horizon $[t_k, t_k + N_c T]$. This process involves the following steps:

1. The optimal control sequence is computed by solving the optimization problem.
2. The first element of this sequence, corresponding to the interval $[t_k, t_k + T]$, is applied to the system.
3. The control horizon is shifted forward by one sampling period.
4. The optimization problem is reformulated and solved for the new horizon $[t_k + T, t_k + (N_c + 1)T]$.
5. This new solution is then applied at the next sampling instant, t_{k+1} .

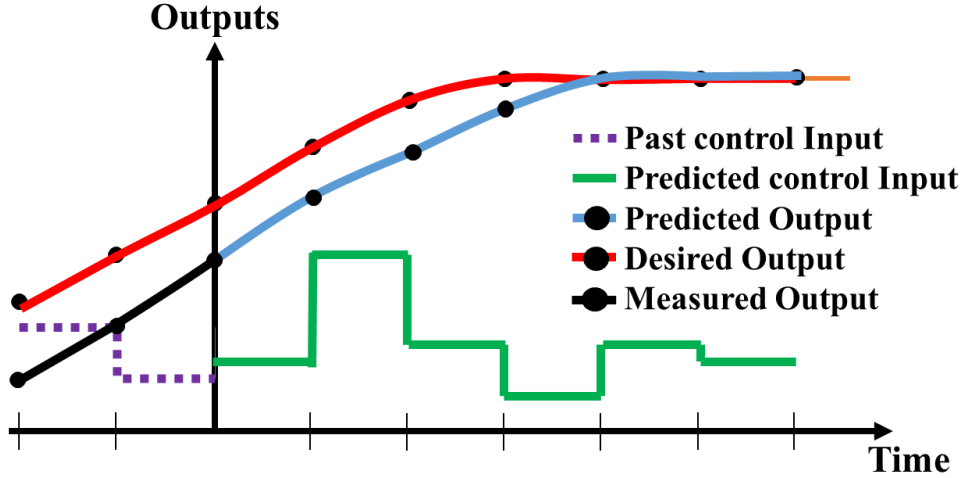


Figure 2.1: A schematic of the predictive control at time t

The receding horizon strategy ensures that the controller always uses the most recent measurements and predictions, enhancing its ability to handle uncertainties and nonlinearities in the system. This approach strikes a balance between computational feasibility and control performance, as it solves a finite-horizon problem while approximating the behavior of an infinite-horizon controller. Figure 2.1 illustrates this dynamic optimization process,

highlighting the relationship between the prediction horizon, control horizon, and the implemented control actions. The figure demonstrates how the optimization window shifts with each time step, always looking ahead to anticipate and optimize future system behavior.

2.6 DMPC

To balance computational efficiency and solution accuracy, a degree of inter-controller communication is essential, giving rise to the concept of DMPC. Some classic review papers have been written on DMPC architecture.^{42;43} The classification of DMPC architectures can be based on the cost function optimization approach and the communication topology among local controllers. When each local controller optimizes its own cost function independently, it is termed *non-cooperative DMPC*, and *cooperative DMPC* refers to an algorithm where all controllers optimize a global cost function.⁴³

2.6.1 Non-cooperative DMPC

Non-cooperative DMPC finds its primary application in systems composed of interconnected subsystems. These subsystems operate autonomously but are bound by shared constraints, creating a complex network of independent yet interrelated components. The concept of a DMPC sequential architecture was first introduced in the literature⁴⁴. This approach involves a step-by-step process where each local controller in the sequence performs its calculations and actions. Specifically, the i -th controller determines and implements its set of manipulated inputs for its corresponding subsystem. Only after this step is complete does the $(i + 1)$ -th controller begin its computations. Parallel evaluation is predominantly employed in non-cooperative algorithms. Researchers have developed a parallel DMPC approach for certain discrete-time linear systems.³⁴ This method was subsequently expanded to address dynamically decoupled subsystems, wherein the system's dynamic behavior is interconnected through shared cost functions and constraints.⁴⁵ A parallel DMPC algorithm for non-cooperative, dynamically coupled nonlinear systems has been developed. This approach

ensures stability through a key principle: each local controller commits to maintaining its state and input trajectories within a bounded deviation from the trajectories anticipated by neighboring subsystems.^{46–48}

2.6.2 Cooperative DMPC

In cooperative DMPC, Venkat and colleagues propose a fundamental principle that all local controllers share a common optimization framework. This unified approach underpins the control strategy’s cooperative nature.⁴⁹ This methodology employs an iterative process. Initially, each local controller optimizes the global cost function under the assumption that other controllers’ actions remain constant. Subsequently, the controllers exchange updated information. Using this new data, they then recalculate their respective sets of manipulated inputs. From an engineering perspective, cooperative control is widely used due to its practical advantages, particularly the fact that it does not require system decomposition. In other words, cooperative control strategies can be implemented without the need for decoupled or coupled models of the system. This cycle of optimization, information exchange, and recalculation continues until convergence is achieved.¹⁶ The optimization problem for the i^{th} subsystem is formulated as:

$$\min_{u_{(i)}} \int_{t_k}^{t_k + N_c T} \left(\tilde{y}(i)^T P(i) \tilde{y}(i) + \tilde{y}(i)^T W_{(i)} \tilde{y}(i) \right) dt \quad (2.10)$$

subject to the system constraints:

$$\begin{aligned} \dot{x} &= f(x, u, t) \\ y_{(i)} &= h_{(i)}(x, t) \\ u_{(i)}^{\min} &\leq u_{(i)} \leq u_{(i)}^{\max} \\ \xi_{(i)}(x, u, t) &\leq 0 \\ \phi_{(i)}(x, u, t) &= 0 \end{aligned} \quad (2.11)$$

where $P_{(i)}$ and $W_{(i)}$ are matrices designed to penalize the elements corresponding to the outputs and manipulated inputs of the i^{th} subsystem, respectively. At each sampling time t_k , the control agents perform the following steps:

1. Calculate the optimal projected trajectory of manipulated inputs for their respective subsystems by solving the optimization problem in Equation 2.10 over the prediction horizon $[t_k, t_k + N_c T]$.
2. Exchange calculated manipulated inputs with other control agents.
3. Recalculate manipulated inputs using the updated information received from other control agents.
4. Repeat steps 2 and 3 until the absolute value of the differences between two successive iterations falls below a predetermined threshold.

For the remainder of this report, we will use the term "DMPC" to refer specifically to cooperative DMPC for the sake of brevity and convenience. This simplification allows us to discuss the topic more efficiently without repeatedly using the full term. It's worth noting that the field of DMPC has garnered significant attention in both theoretical research and practical applications. As a result, numerous review papers have been published, covering various aspects of DMPC:^{4;50;51} A comprehensive review has been conducted on DMPC and its various topologies. This review not only summarizes the current state of DMPC research but also provides empirical evidence of its effectiveness compared to centralized MPC. The superiority of DMPC was demonstrated through a case study involving the closed-loop performance of a benzene alkylation process.^{16;52}

2.7 Community detection

The identification of optimal communities within process systems has been recognized as a critical factor directly influencing the closed-loop performance of controllers in DMPC. This significance has led to substantial research interest, resulting in numerous comprehensive

reviews that examine various community detection approaches.⁵³ Community detection is also super important in other fields, e.g., Power Systems⁵⁴, Biological Networks⁵⁵, Social Network Analysis⁵⁶, and Transportation Networks⁵⁷. Recently, researchers have conducted extensive investigations into identifying ideal community structures.^{19;58-61} The modularity measure is a well-known method for evaluating community detection in networks. It quantifies the quality of identified communities by comparing the actual number of connections within communities to the expected number in a random network with the same degree distribution. Essentially, it assesses how well the discovered communities capture the network's internal structure.^{62;63}

Spectral community detection is another prominent approach for identifying communities within networks. This method leverages the mathematical properties of matrices associated with the network, specifically their eigenvalues and eigenvectors.⁶⁴ Spectral clustering algorithms can be primarily divided into two categories: the Normalized Cut method and the RatioCut method. These approaches differ in their objective functions and normalization techniques. The Normalized Cut method, denoted as $\text{NCut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)}$, aims to minimize the cut size relative to the volume of each partition.⁶⁵ In contrast, the RatioCut method, expressed as $\text{RatioCut}(A, B) = \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|}$, seeks to minimize the cut size relative to the number of vertices in each partition. Both methods utilize eigenvectors of graph Laplacian matrices to approximate optimal solutions, but their distinct normalizations can lead to different clustering results, particularly in graphs with varying degrees of vertex connectivity.⁶⁶

2.7.1 Modularity based methods

These approaches predominantly emphasize the maximization of modularity, a mathematical metric quantifying the proportion of edges encompassed by a subsystem. The modularity Q is typically defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \quad (2.12)$$

where A_{ij} represents the weight of the edge between nodes i and j , k_i and k_j are the degrees of nodes i and j respectively, m is the total number of edges in the network, c_i and c_j are the communities of nodes i and j , and δ is the Kronecker delta function.

It is imperative to recognize, however, that despite their demonstrated effectiveness in identifying communities across diverse network types, modularity-based methods are not without limitations. These methods provide a wide spectrum of techniques for community detection, each exhibiting distinct advantages and constraints.

Louvain Method

This method executes rapidly and reveals communities in a hierarchical manner, providing access to multiple levels of clustering. This approach initializes with each node in a distinct community. Subsequently, for each node i and its neighbors j , the algorithm evaluates the change in modularity ΔQ that would result from merging nodes i and j . Node i is then merged with the neighbor that maximizes the modularity gain.⁶⁷ This process is represented by the equation:

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \quad (2.13)$$

The terms in the equation are defined as follows:

- $\sum_{in}$: The aggregate weight of all links within community C
- $\sum_{tot}$: The total weight of links connected to nodes in community C
- k_i : The sum of weights of all links connected to node i
- $k_{i,in}$: The total weight of links connecting node i to nodes within community C
- m : The sum of weights of all links in the entire network

Despite the widespread adoption of this approach, it faces significant limitations in efficiency. The method relies heavily on greedy search algorithms, which, while effective for

many applications, prove inadequate for real-time DMPC. The computational demands of greedy search, coupled with the iterative nature of the algorithm, render it unsuitable for systems requiring rapid, dynamic community detection and control adjustments. A novel method combines differential privacy and homomorphic encryption to protect data confidentiality while identifying clusters at varying resolutions. It employs the Louvain algorithm to optimize modularity, enabling the discovery of community structures in complex networks while maintaining robust privacy safeguards. This approach offers a secure means of performing cluster analysis on sensitive data across multiple scales of granularity.⁶⁸ In⁶⁹ improved version of the Louvain method is tailored for distributed memory systems, enabling efficient processing of enormous graphs with billions of edges. It introduces innovative graph mapping and data representation techniques, coupled with an optimized communication system, to achieve fine-grained parallelism on supercomputers. This approach tackles the scalability issues in community detection, allowing for rapid analysis of massive networks. The implementation demonstrates remarkable scalability, successfully parallelizing graphs containing up to 138 billion edges across thousands of nodes. To address scalability challenges in managing extensive and intricate systems, researchers have created a decentralized adaptation of the Lovian methodology.⁷⁰ The conventional Lovian technique has undergone a significant alteration. Instead of evaluating all possible communities, it now selects a random neighboring community as the potential destination for a node. This modification accelerates the community detection process while maintaining the quality of results.⁷¹

The Leiden method for community detection offers an innovative alternative to the traditional Louvain approach. In their research, Traag et al.⁷² demonstrated that the Leiden algorithm ensures well-connected communities, addressing a key limitation of the Louvain method. Their study also established that the Leiden algorithm not only outperforms the Louvain algorithm in terms of speed but also produces superior community partitions. Furthermore, the Leiden method provides explicit guarantees about the quality of the detected communities, a feature absent in the Louvain approach. The Leiden algorithm offers several key advantages over the Louvain method in community detection. Primarily, it guarantees well-connected communities, a feature the Louvain method lacks. Leiden incorporates a re-

finement phase that allows for the identification and reassignment of poorly connected nodes, leading to more cohesive communities. This refinement step also enables the algorithm to escape local optima more effectively, resulting in higher-quality partitions overall. Additionally, Leiden can split poorly connected communities during the process, which Louvain cannot do. This ability to break up communities contributes to its superior performance in detecting fine-grained community structures. Furthermore, Leiden is generally faster and more computationally efficient, especially for larger networks. These improvements collectively result in more accurate and reliable community detection, particularly in complex and large-scale networks where the Louvain method might struggle or produce suboptimal results.^{73–75}

Infomap Method

This approach is primarily based on information theory, aiming to minimize random walk length in networks while revealing the community structure in weighted and directed systems. The method utilizes the probability flow of random walks on a network as a proxy for information flows in real systems, decomposing the network into modules by compressing a description of this probability flow. This technique results in a map that simplifies and highlights structural regularities and their relationships, as demonstrated in a study of scientific communication captured through citation patterns of over 6,000 journals. The analysis revealed a multicentric organization with fields varying dramatically in size and degree of integration into the network of science. Notably, along the network’s backbone—encompassing physics, chemistry, molecular biology, and medicine—information flows bidirectionally, while a directional pattern of citation from applied fields to basic sciences was observed.⁷⁶ The methodology is then adapted for distributed implementation, allowing it to be utilized on expansive and intricate networks.⁷⁷ A novel community detection algorithm called Infomap-SA, which combines the Infomap approach with Simulated Annealing (SA) has been introduced.⁷⁸ This innovative method is founded on the concept of optimizing the modularity function. Using map equations and distributed implementation, this approach has been

improved.⁷⁹

Kernighan-Lin method

This method employs a greedy search algorithm to maximize modularity. Kernighan et al.⁸⁰ tackled the problem of graph partitioning with edge costs, seeking to optimize node distribution into subsets of specified sizes while minimizing the total cost of inter-partition edges. This approach aims to maximize modularity through iterative node reassignment across communities. The process begins with an initial configuration, followed by successive node movements until convergence is achieved. While this method does not guarantee a globally optimal solution, its simplicity and efficiency have led to its widespread adoption. An enhanced approach to conventional greedy-based community detection has been developed, incorporating an additional fine-tuning step. This modified strategy, when applied to the leading eigenvalue method, effectively eliminates the restrictive constraints that typically hinder traditional methods. The efficacy of this approach is evidenced by substantially improved modularity scores across both synthetic and real-world network analyses. The introduction of a final-tuning phase, which evaluates the potential reassignment of each node to any community, results in a more precise and impartial identification of community structures. This enhancement renders the method highly competitive with, and in some instances superior to, state-of-the-art community detection algorithms.⁸¹ A novel efficient method has been developed that combines Newman spectral community detection with a modified fine-tuning approach. This technique allows for fine-tuning of 10% to 20% of nodes, striking a balance between community detection accuracy and computational efficiency.⁸²

Label Propagation Algorithm

Data labeling plays a crucial role in machine learning, particularly in supervised learning algorithms where the model learns from labeled examples to make predictions on new, unseen data. The Label Propagation Algorithm (LPA) offers an efficient approach to this labeling process, especially in network or graph-based scenarios. In LPA, the process begins by

assigning a unique label to each node in the network. Subsequently, the algorithm iterates through the nodes, updating each node’s label based on the most frequent label among its neighboring nodes. This propagation of labels continues until a stable state is reached or a predefined number of iterations is completed. The beauty of this method lies in its ability to leverage the network structure to spread information, allowing a small set of initially labeled data to influence the labeling of a much larger set of unlabeled data. This approach is particularly valuable in semi-supervised learning contexts where obtaining labeled data can be expensive or time-consuming. By exploiting the underlying structure of the data, LPA can effectively propagate labels through the network, potentially revealing natural groupings or communities within the data that might not be immediately apparent through other methods.⁸³

The LPA has been innovatively extended to enhance community detection by integrating it with modularity maximization, resulting in a more robust and efficient method known as LPA-MNI (Label Propagation Algorithm with Modularity and Node Importance). This advanced approach begins by leveraging modularity maximization to identify initial community structures within the network, providing a solid foundation for the subsequent label propagation process. By assigning unique labels to nodes within these preliminary communities, the method significantly reduces the inherent randomness often associated with traditional LPA implementations. A key feature of LPA-MNI is its incorporation of node importance, quantified through degree centrality, which determines the sequence of label updates. This strategic ordering ensures a more systematic and stable propagation of labels throughout the network, leading to more consistent and reliable community detection results. The effectiveness of LPA-MNI has been rigorously validated through extensive experimentation on a diverse range of networks, encompassing both real-world datasets and synthetic benchmarks. These comprehensive tests have consistently demonstrated LPA-MNI’s superior performance in terms of accuracy, robustness to network variations, and computational efficiency when compared to traditional community detection algorithms. Moreover, this integrated approach shows particular promise in handling complex network structures where conventional methods may falter, offering a powerful tool for researchers and practitioners

in fields such as social network analysis, bioinformatics, and data mining, where accurate community detection is crucial for understanding system dynamics and informing decision-making processes.⁸⁴

The LPA for community detection has been refined by recasting it as an optimization problem. This novel approach, known as the Constrained Label Propagation Algorithm (CLPA), introduces a cost function with constraints to penalize less desirable outcomes. By doing so, CLPA offers a more nuanced and flexible method for refining community structures within networks. The algorithm’s effectiveness has been thoroughly evaluated through comparative analyses on both synthetic networks and real-world datasets. These comprehensive tests have not only demonstrated CLPA’s capacity to overcome the limitations inherent in standard LPA implementations but have also showcased its remarkable adaptability across various network topologies and structures. This versatility makes CLPA a powerful tool for researchers and practitioners dealing with complex network analysis, as it can effectively handle the diverse challenges presented by different types of networks while maintaining high accuracy in community detection.⁸⁵ A multitude of modifications and enhancements have been developed to improve upon the basic LPA approach for community detection. These advancements aim to address various limitations of the original algorithm and to optimize its performance across different types of networks and application scenarios.^{86–90}

2.7.2 Spectral clustering methods

Spectral clustering methods have emerged as a powerful and popular approach to community detection in networks, leveraging the spectral properties of graphs to identify distinct groups of nodes. These methods exploit the eigenvalues and eigenvectors of the graph Adjacency matrix to effectively partition nodes, making them particularly adept at uncovering communities with complex topological structures. The strength of spectral clustering lies in its versatility and adaptability. It can handle a wide range of network types - from sparse to dense, and even those with disconnected components - by capturing essential structural information through eigenvalues and eigenvectors. This flexibility extends to the choice of

similarity matrix, allowing spectral clustering to be applied across diverse domains, from social networks to biological systems. The spectral clustering process typically involves three key steps:

1. **Constructing adjacency matrix:** This matrix captures the relationship between node i and j . In the weighted format of this matrix, the component i, j is the strength of the relationship between the two aforementioned nodes.
2. **Eigenvalue decomposition:** This step involves decomposing the graph matrix to obtain eigenvalues and eigenvectors, with the number considered depending on the desired community count.
3. **Clustering eigenvectors:** Finally, clustering algorithms like k-means or normalized cut are applied to the eigenvectors to delineate distinct communities.

This chapter concludes with a different approach to spectral community detection.

Normalized cut algorithm

Normalized Cuts and Image Segmentation is a powerful technique that applies graph theory and spectral clustering to partition images into meaningful segments. The process begins by representing the image as a graph, where pixels are nodes connected by edges weighted based on similarity. The algorithm aims to partition this graph using the normalized cut criterion, which balances the dissimilarity between segments with the similarity within segments. This partitioning problem is then reformulated as an eigenvalue problem using the graph Laplacian matrix. By computing the second smallest eigenvector of the generalized eigenvalue system, the method obtains a solution that can be used to bipartition the graph. This process can be applied recursively for multi-way partitioning. The resulting graph partitions correspond to image segments, each representing a region with similar characteristics. This approach is notable for its consideration of both local and global information, robustness to noise, and ability to handle complex textures and gradual variations. While computationally intensive for large images, the method has been influential in computer vision and graph

theory, demonstrating the effectiveness of spectral methods in solving complex partitioning problems.⁹¹

RatioCut algorithm

RatioCut is a spectral clustering algorithm designed for graph partitioning and community detection. It aims to divide a graph into k clusters by minimizing the ratio of the cut size to the number of vertices in each partition. Unlike some other methods, RatioCut focuses on balancing the sizes of partitions in terms of vertex count rather than edge volume. The algorithm operates by constructing an unnormalized graph Laplacian matrix and finding its eigenvectors. Specifically, it utilizes the k smallest eigenvectors to embed the graph into a k -dimensional space, which is then clustered (typically using k -means) to determine the final graph partitioning. This approach tends to produce more balanced partitions in terms of vertex count, making it particularly useful when clusters of similar sizes are desired. RatioCut can be solved as a generalized eigenvalue problem and is generally less sensitive to outliers compared to some alternative clustering methods. While effective for many graph partitioning tasks, its performance may vary depending on the graph's density distribution and the expected variation in cluster sizes.⁹²

Eigenvector based clustering

This innovative study tackles the critical challenge of identifying small vertex separators in graphs, a problem with significant ramifications for the parallel factorization of sparse, symmetric matrices. The researchers introduce a novel algebraic method that harnesses the spectral characteristics of graphs, particularly focusing on the eigenvalues and eigenvectors of the Laplacian matrix. Their work establishes important lower bounds for separator sizes using Laplacian eigenvalues and showcases an efficient technique for computing Laplacian eigenvectors of grid graphs through Kronecker products of path graph eigenvectors. A cornerstone of their contribution is a heuristic algorithm designed for general graphs. This algorithm first derives an edge separator from a Laplacian eigenvector and then employs

maximum matching in a subgraph to determine the vertex separator. The study provides extensive comparative analysis, juxtaposing the quality of separators produced by their spectral method against those generated by alternative algorithms. Additionally, it delves into the computational aspects, examining both the time required and the necessary precision for Laplacian eigenvector calculations. A notable strength of this spectral approach lies in its adaptability to medium-scale multiprocessor systems, where it can be implemented with relative ease. This feature potentially broadens its practical applications in parallel computing scenarios, offering a valuable tool for researchers and practitioners in the field of graph theory and high-performance computing.^{93;94} The recursive spectral bisection method builds upon this approach, allowing for the identification of multi-level community structures within complex networks.^{95;96}

Graph embedding is a technique in machine learning and data science that aims to represent a graph or network structure in a lower-dimensional vector space while preserving important properties of the original graph.⁹⁷ Graph embedding with clustering is an advanced approach that integrates graph embedding techniques with clustering algorithms. This method transforms graph nodes into points in a feature space, where the features are strategically optimized to minimize the negative log-likelihood of preserving sampled vertex neighborhoods. In other words, it aims to create vector representations of nodes that maintain the structural relationships of the original graph while simultaneously identifying natural groupings or clusters within the data. By combining these two objectives, this approach enables a more comprehensive analysis of complex network structures, capturing both local relationships between nodes and global patterns in the graph. This makes it particularly useful for tasks such as community detection in social networks, functional group identification in biological networks, or any application where understanding both detailed node relationships and overall graph structure is crucial.⁹⁸

2.8 Comparing Community Detection Methods: An Integrated Process Control Perspective

Community detection approaches can be broadly extended into probabilistic techniques and deep learning-based algorithms. Each approach we’ve discussed has its unique advantages and limitations. From a control systems perspective, it’s crucial to find a solution that balances speed and accuracy, as any delay in receiving information from the controller can result in financial losses and reduced efficiency. It’s important to note that approaches based on fine-tuning are often less desirable due to the additional time they require. Rather than categorizing community detection methods based on their underlying principles, we classify them according to their ability to identify communities either in a single batch step or through a hierarchical process. Among the methods we’ve examined, only the eigenvalue-based community detection algorithm can identify communities in a single step. Other approaches typically require hierarchical clustering followed by fine-tuning. This characteristic makes the eigenvalue-based method particularly attractive for our purposes. The speed of community detection is paramount in real-time control applications. Faster algorithms allow for more frequent updates to the community structure, enabling the control system to adapt more quickly to changing conditions. This can lead to improved overall system performance and stability.

While accuracy is important, there’s often a trade-off between speed and precision. In many control applications, a slightly less accurate but much faster algorithm may be preferable to a highly accurate but slow one. The key is to find an approach that meets the minimum accuracy requirements while maximizing speed. Scalability is another crucial factor to consider. As industrial processes grow more complex and the number of variables increases, the chosen community detection method must be able to handle larger datasets without a significant drop in performance. Given these considerations, we have selected the eigenvalue-based community detection approach as our preferred method. This choice is based on its ability to find communities in a single step, its favorable balance of speed

and accuracy, and its potential for scalability. In the following chapter, we will provide a comprehensive mathematical overview of this eigenvalue-based approach. Furthermore, we will demonstrate how we have enhanced this method and applied it to a case study involving benzene alkylation. This practical application will illustrate the effectiveness of our chosen approach in a real-world industrial context. By implementing this improved eigenvalue-based community detection algorithm, we aim to enhance the efficiency and responsiveness of control systems in complex chemical processes. This, in turn, should lead to optimized operations, reduced costs, and improved overall performance in industrial settings.

Chapter 3

Spectral community detection approach

3.1 Weighted Adjacency Matrix

This chapter proposes a novel approach to optimal decomposition for distributed control of integrated process systems operating under dynamic conditions. Unlike previous studies that focused solely on the existence of variable relationships within network systems, we introduce a method that assesses the interaction intensities among manipulated inputs, state variables, and controlled outputs. This evaluation generates an adaptive graph representation of the process network, serving as the foundation for developing flexible, optimal decomposition strategies across various operational scenarios.

The challenge of defining optimal decomposition for weighted process networks is central to our investigation. Our aim is to identify a configuration of p -subsystems that maximizes network density. To quantify the effectiveness of our decomposition, we utilize the modularity concept, represented by Q , on a directed weighted graph. While existing literature has extensively explored methods for quantifying node interactions in control-affine state space models^{99–101}, our research extends this concept. We explore innovative weight assignment techniques for graph representations encompassing a broader range of models, including

those with non-control-affine state space characteristics.

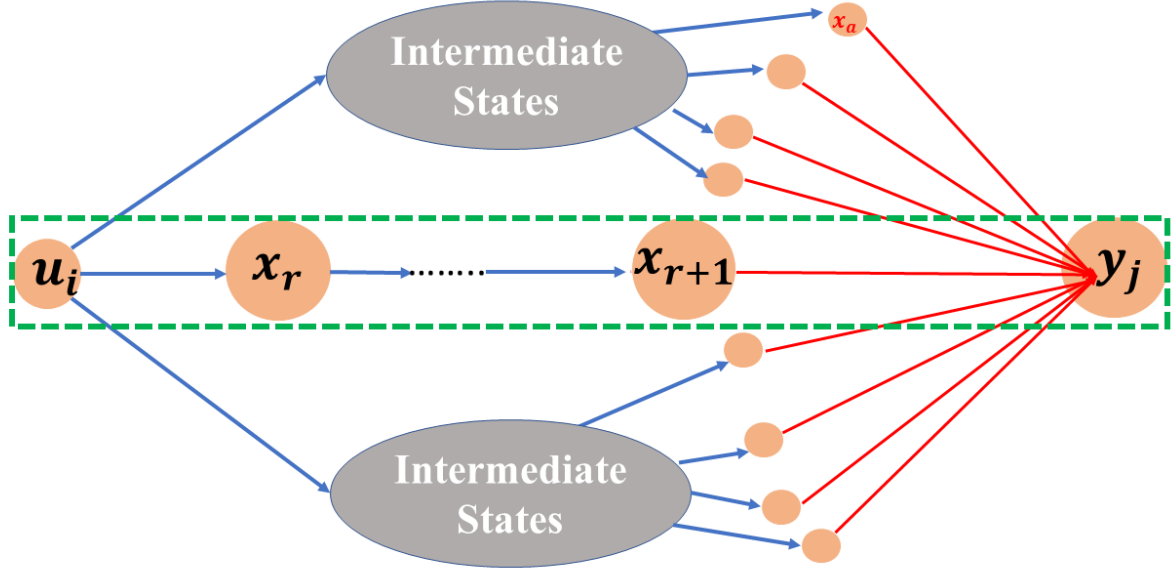


Figure 3.1: *Small network section representing communication in the entire system.*

Optimizing the modularity of a weighted graph decomposition necessitates the construction of a weighted adjacency matrix that accurately represents the strength of inter-vertex connections. This matrix encompasses three distinct weight categories:

- The influence of inputs on state variables
- The impact of individual state variables on other states
- The effect of states on controlled outputs

Figure 3.1 depicts a segment of an integrated network, emphasizing the vertices and edges that establish connections between the manipulated input u_i and the controlled output y_j . A green box highlights the shortest path from u_i to y_j , which serves as a basis for weight generation and can be extrapolated to other network segments. The weight quantifying the impact of manipulated input u_i on controlled output y_j is defined as: $\frac{\partial(\frac{\partial^r y_j}{\partial t^r})}{\partial u_i}$

This expression represents the steady-state gain of y_j with respect to u_i . In this context, r denotes the system's relative order for the input-output pair (u_i, y_j) ¹⁰². To determine this weight, we begin by computing the time derivative of the output y_j : $\frac{dy_j}{dt}$

$$\frac{dy_j}{dt} = \frac{dh_j(x)}{dt} = \sum_{i=1}^{K_1} \frac{\partial h_j(x)}{\partial x_i} \frac{\partial x_i}{\partial t}, \quad (3.1)$$

In this expression, the summation term K_1 encompasses all state variables that have a direct connection to the output node y_j . By focusing our analysis on the states within the shortest path—as indicated by the green box in Figure 3.1—we can refine Equation 3.1 to a more concise form:

$$\frac{dy_j}{dt} = \frac{\partial h_j(x)}{\partial x_r} \frac{\partial x_r}{\partial t} = \frac{dh_j(x)}{dx_r} (f_r(x) + g_r(x, u)), \quad (3.2)$$

This simplification stems from the observation in Figure 3.1 that the expression $g_r(x, u)$ is zero due to the lack of connection between the variables u_i and x_r . Therefore, we turn our attention to calculating the higher-order derivative of the output to establish an explicit relationship with the input variable u_i . As shown in Figure 3.1, the first explicit appearance of u_i in the derivative of y_j occurs in the r^{th} derivative,

$$\begin{aligned} \frac{dy_j}{dt} &= \frac{\partial h_j(x)}{\partial x_r} \frac{\partial x_r}{\partial t} = \frac{dh_j(x)}{dx_r} f_r(x), \\ \frac{d^2 y_j}{dt^2} &= \frac{d}{dt} \left(\frac{dh_j(x)}{dx_r} f_r(x) \right) = \frac{\partial^2 h_j(x)}{\partial x_r^2} f_r^2(x) + \frac{\partial h_j(x)}{\partial x_r} \frac{\partial f_r(x)}{\partial x_{r-1}} f_{r-1}(x), \\ &\vdots \\ \frac{d^r y_j}{dt^r} &= \frac{\partial}{\partial x_r} \left(\left(\frac{\partial}{\partial x_r} \cdots \left(\frac{\partial}{\partial x_r} \left(\frac{\partial h_j(x)}{\partial x_r} f_r^r(x) \right) \right) \right) \right) + \frac{\partial h_j(x)}{\partial x_r} \prod_{o=2}^r \frac{\partial f_o}{\partial x_{o-1}} (f_1(x) + g_1(x, u)), \end{aligned} \quad (3.3)$$

To derive the expression for the weights, we calculate the derivative of Equation 3.3 with respect to u_i ,

$$\frac{\partial(\frac{\partial^r y_j}{\partial t^r})}{\partial u_i} = \frac{\partial h_j(x)}{\partial x_r} \prod_{o=2}^r \frac{\partial f_o}{\partial x_{o-1}} \frac{dg_1(x, u)}{du_i}. \quad (3.4)$$

In Equation 3.4, the last term signifies the weight of the link between the input and its adjacent state. The first term represents the weight of the connection linking the output to its associated state, and the middle term indicates the intensity of inter-state relationships. Table 3.1 succinctly displays the calculated weights, derived from the state space model outlined in Equation 2.1.

Table 3.1: *Definition of edge-weights at a steady state operation point.*

Edge	Input-State	State-State	State-Output
Symbol	$u_i \rightarrow x_j$	$x_i \rightarrow x_j$	$x_i \rightarrow y_j$
Weight	$ \frac{\partial g_j}{\partial u_i} $	$ \frac{\partial f_j}{\partial x_i} $	$ \frac{\partial h_j}{\partial x_i} $

3.2 Community Detection Algorithm

This section employs spectral community detection techniques to analyze a weighted directed graph representation of the processing system.¹⁹ The methodology is outlined in detail in the provided algorithm, which takes as input the graph $G(V, E, W)$ and the desired number of subsystems, P .

Using the weighted network structure outlined earlier, we can construct the distributed control structure and its corresponding weighted adjacency matrix. To identify the most effective decomposition approach, we develop a comprehensive definition of modularity specifically tailored for weighted directed networks. This definition can be expressed as follows:

$$Q = \frac{1}{m} \sum_{ij} \left[A_{ij} - \frac{d_i d_j}{m} \right] \delta_{g_i, g_j}, \quad (3.5)$$

where:

- d_i represents the total weight of incoming edges to node i
- d_j signifies the total weight of outgoing edges from node j

Algorithm 1 Spectral Community Detection

Require: $G(V, E, W), P$

Ensure: Communities

```
1: Initialize Communities as an empty list
2: Compute spectral properties  $S$ 
3: Assign a vertex vector  $r_i$  for each vertex  $v$  based on  $S$ 
4: Initialize  $R_s$  with an initial guess value
5: repeat
6:   for each vertex  $v$  in  $V$  do
7:     Find the community  $C$  maximizing the inner product between  $r_i$  and  $R_C$ 
8:     Allocate  $v$  to the community with the highest inner product
9:   end for
10:  Update  $R_s$  based on current Communities
11: until convergence
12: return Communities
```

- m denotes the sum of all edge weights in the network
- δ is the Kronecker delta function

Equation 3.5 defines modularity by comparing two ratios:

1. The actual proportion of edge weights within subsystems
2. The expected proportion if edges were randomly distributed while maintaining each node's original weight distribution

In this equation:

- A_{ij} represents the weight of the edge from node j to node i
- $\frac{d_i d_j}{m}$ gives the expected edge weight between nodes i and j under random redistribution
- δ_{g_i, g_j} ensures that only node pairs within the same community contribute to the sum

The modularity matrix is expressed as $B_{ij} = A_{ij} - \frac{d_i d_j}{m}$, and its symmetric counterpart is denoted by $B_1 = B + B^T$. As a result, modularity can be reevaluated using this symmetric modularity matrix,

$$Q = \frac{1}{m} \sum_{ij} B_{1,ij} \delta_{g_i, g_j}. \quad (3.6)$$

According to line 3 of Algorithm 1, each node in the control system is allocated a vector based on the system's spectral characteristics. The vector for each node is then produced using the modularity formula and spectral property of the adjacency matrix¹⁹. Through eigen decomposition, the matrix B_1 can be represented as:

$$B_{1,ij} = \sum_{l=1}^N \lambda_l U_{il} U_{jl}, \quad (3.7)$$

where λ_l represents the l^{th} eigenvalue of B_1 , and U_{il} signifies the element corresponding to the orthogonal eigenvector matrix U . Furthermore, the Kronecker delta can be reexpressed as:

$$\delta_{g_i, g_j} = \sum_{s=1}^P \delta_{s, g_i} \delta_{s, g_j}. \quad (3.8)$$

By integrating Equations 3.8 and 3.7 into Equation 3.6, we can restate the modularity as:

$$Q = \frac{1}{m} \sum_{ij=1} \sum_{l=1}^N \lambda_l U_{il} U_{jl} \sum_{s=1}^P \delta_{s, g_i} \delta_{s, g_j} = \frac{1}{m} \sum_{l=1}^N \lambda_l \sum_{s=1}^P \left(\sum_{i=1}^N U_{il} \delta_{s, g_i} \right)^2. \quad (3.9)$$

Equation 3.9 serves as the foundation for the spectral community detection approach, allocating a vector to each node. This equation suggests that to optimize modularity, one should include all positive eigenvalues in defining each node's vector, given the presence of the term λ_l . However, this approach may lead to a time-intensive and less efficient community detection process. To mitigate this issue, we propose an energy function as follows:

$$E = \frac{\sum_{i=1}^O \lambda_{N-i}}{\sum_{i=1}^N \lambda_{N-i+1}} > \epsilon, \quad (3.10)$$

to capture the O most significant eigenvalues. In this context, ϵ constrains the number of positive dominant eigenvalues, retaining only the most influential ones in defining each node's vector. This method enhances the speed of the community detection process, maintaining efficiency while preserving accuracy within an acceptable range determined by ϵ . Given that the eigenvalues are positive, the term λ_l can be relocated to the innermost summation in Equation 3.9. Considering this and utilizing Equation 3.10, we can reformulate Equation 3.9

as follows:

$$Q = \frac{1}{m} \sum_{l=1}^O \sum_{s=1}^P \left(\sum_{i=1}^N \sqrt{\lambda_l} U_{il} \delta_{s,gi} \right)^2. \quad (3.11)$$

As a result, we define a vector for each node, denoted as r_i , where $[r_i]l = \sqrt{\lambda_l} U_{il}$. Moreover, we can further simplify Equation 3.11 as follows

$$Q = \frac{1}{m} \sum_{s=1}^P \sum_{l=1}^O \left(\sum_i [r_i]l \right)^2 = \frac{1}{m} \sum_{s=1}^P \left| \sum_i r_i \right|^2. \quad (3.12)$$

We can rewrite Equation 3.12 by defining $R_s = \sum_i r_i$, where the summation is performed over the node vectors:

$$Q = \frac{1}{m} \sum_s |R_s|^2. \quad (3.13)$$

Equation 3.13 expresses modularity in terms of the subsystem vectors, determining the most appropriate subsystem (either A or B) for node i . Equation 3.14 calculates the modularity change resulting from moving node i from subsystem B to subsystem A :

$$Q = \frac{1}{m} [|\mathbf{R}_A|^2 + |\mathbf{R}_B + \mathbf{r}_i|^2 - |\mathbf{R}_A + \mathbf{r}_i|^2 - |\mathbf{R}_B|^2] = \frac{1}{m} [\mathbf{R}_B^T \cdot \mathbf{r}_i - \mathbf{R}_A^T \cdot \mathbf{r}_i]. \quad (3.14)$$

The optimal sub-community for node i is chosen by identifying the one that produces the largest increase in modularity. As outlined in¹⁹ and Algorithm 1, the spectral community detection method employs an iterative approach using subsystem vectors. These vectors are initially set to values that comply with Equation 3.15. Given that the modified modularity matrix is symmetric, the uniform vector is always one of its eigenvectors. This condition requires that the sum of all elements in any other eigenvector must equal zero. As a result,

we can establish the following constraints for the subsystem's vector¹⁹:

$$\begin{aligned}
\sum_{i=1}^n |r_i| &= \sqrt{\lambda} \sum_{i=1}^n U_{il} = 0, \\
\sum_{i=1}^n r_i &= 0, \\
\sum_s R_s &= \sum_s \sum_{i \in s} r_i = \sum_{i=1}^n r_i = 0.
\end{aligned} \tag{3.15}$$

Equation 3.15 suggests that for community detection approaches based on spectral decomposition, only $k - 1$ initial subsystem vectors are necessary, given that the angle between any two of them is greater than 90 degrees; otherwise, two groups might merge during the first iteration. The process involves calculating the inner product of each representative vector with the subsystem vector, which helps determine the most suitable group for each node. Nodes are assigned to the group whose inner product with the representative vector is the highest. Subsequently, the vectors from all nodes within a group are combined to form new community vectors $\mathbf{R}$. This process of refining and assigning nodes is iterated until no further changes occur. To ensure the controllability of each community, we have also implemented the following constraint:

$$\forall i, \quad n_{\text{inputs},i} \geq n_{\text{outputs},i}. \tag{3.16}$$

The optimization problem is first addressed without taking this constraint into account. Subsequently, the algorithm verifies if the criterion is met. In cases where the criterion is not fulfilled, nodes with the least impact on the modularity value are identified. This is done by ranking the inner product of each node with its corresponding group vector. This ranking aids in determining which nodes can be transferred to another community with minimal penalty. These nodes are then reallocated to satisfy the criterion^{103;104}.

Remark 2. *While modularity is recognized as the most effective measure for evaluating network structure, it's important to acknowledge other significant metrics in assessing network*

community detection, such as "Coverage" and "Conductance". "Coverage" represents the ratio of intra-community edges to the total number of edges in a network, indicating the degree of interaction within communities. On the other hand, "Conductance" assesses a community's links to the broader network by measuring the proportion of inter-community edges to the total edges emanating from a community. Although these metrics are essential for evaluating community detection quality, they are complemented by modularity. Modularity offers a more comprehensive evaluation by contrasting the observed community structure of a network with a random structure, thus assessing the overall quality of partitioning^{105;106}.

3.3 Comparative Study: Enhanced Spectral Community Detection Approach vs. Traditional Methods

A collection of 20 random graphs was created, each containing 15,000 nodes. In these graphs, the intra-community edge formation probability, P_{in} , ranged from 0.45 to 0.9, while the inter-community edge formation probability, P_{out} , spanned from 0.05 to 0.33. For each graph, community detection was performed using three distinct methods to identify three subsystems. The first method incorporates all positive eigenvalues in Equation 3.9. The second applies the P-rank approximation, as proposed by¹⁹, and the third employs the enhanced spectral community detection technique. To evaluate overall performance and identify a community detection method that optimizes both accuracy and computational efficiency, a metric combining modularity and computation time was devised. This measure, denoted as Φ , is calculated as the ratio of computation time to modularity, with a lower Φ value indicating better performance. For the enhanced approach, ϵ is set to 90%. Figure 3.2 displays the community detection outcomes, where Q , T , and Φ represent modularity, computation time, and the performance indicator, respectively. Furthermore, the subscript t indicates the traditional approach⁵⁹, N signifies the P-rank approximation approach (Newman's method)¹⁹, and I denotes our proposed energy-based enhanced approach. The mean value across all graphs is computed and utilized to assess each community detection method¹⁸.

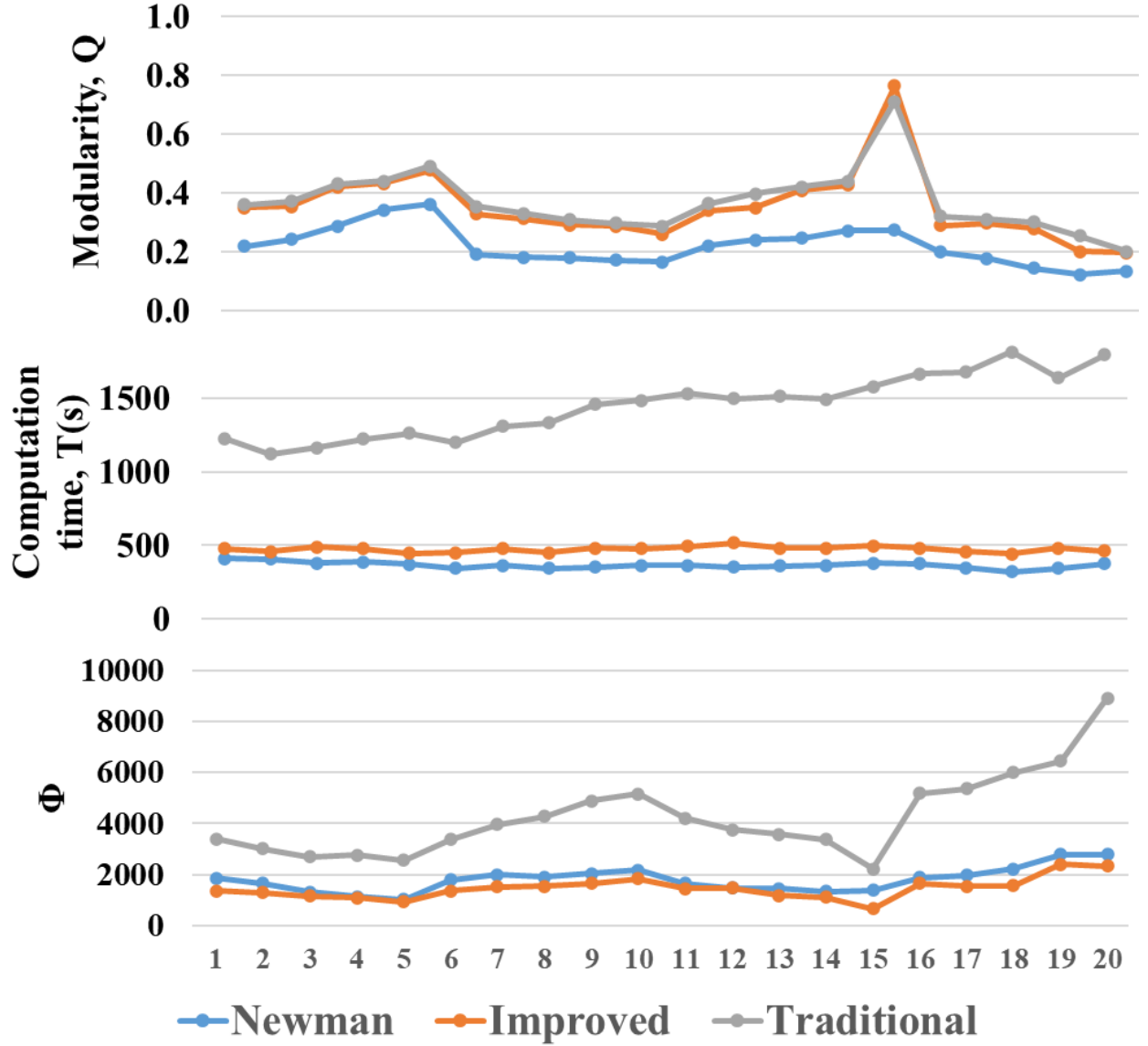


Figure 3.2: *Community detection index results for 20 different random graphs.*

Table 3.2 presents the Φ index for the three community detection methods, derived from Figure 3.2. A lower Φ value demonstrates the enhanced performance of the improved spectral community detection approach.

Method	Newman	Improved	Traditional
Φ	1795.99	1459.55	4261.35

Table 3.2: Φ value for different method.

3.4 Identifying the Optimal Number of Communities:

Elbow method

The identification of the optimal number of communities plays a crucial role in spectral community detection, as outlined in Algorithm 1. To determine this, the elbow method proves useful by analyzing how modularity changes as the community count increases incrementally. This approach involves creating a graph that plots the number of communities against the corresponding modularity shift. Typically, an elbow point becomes apparent on this graph, indicating where further increases in community count result in diminishing returns on modularity improvement. The underlying principle suggests that adding more communities beyond this elbow point does not significantly enhance the network’s partitioning efficiency, thus implying an optimal community count has been reached^{107;108}.

Chapter 4

Case Study: An Integrated Process System

This chapter applies an advanced spectral community detection method to design an optimized distributed control structure tailored to the integrated benzene alkylation process. We also use the elbow technique to identify the ideal number of communities needed for this process across different operational states. To assess our control design strategy's efficacy, we examine its closed-loop performance under various conditions, focusing on medium and low recycle ratios.

4.1 Process Description

The benzene alkylation system comprises four continuously stirred tank reactors (CSTRs) and a flash separator, as illustrated in Figure [4.1](#). This process involves four key components: benzene (A), ethylene (B), ethylbenzene (C), and 1,3-diethylbenzene (D).

In the initial three CSTRs, two chemical reactions take place:

Reaction 1: $A + B \rightarrow C$,

Reaction 2: $C + B \rightarrow D$,

The fourth CSTR exclusively hosts a single reaction:

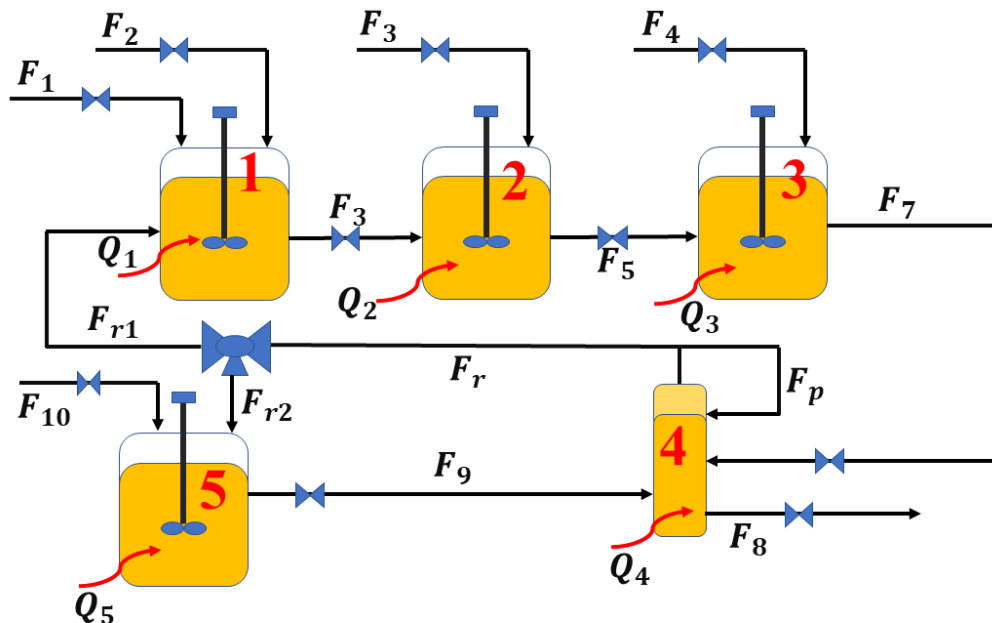


Figure 4.1: *Integrated benzene alkylation process.*

Reaction 3: $A + D \rightarrow C$

In this system, ethylbenzene (C) is the target product, while diethylbenzene (D) is a secondary product¹⁰⁹. The process begins with pure benzene entering the first CSTR and pure ethylene being introduced to the initial three CSTRs. The output from the third CSTR, a mixture of benzene, ethylene, ethylbenzene, and 1,3-diethylbenzene, flows into the separator. Ethylbenzene is extracted from the separator's bottom, while most of the benzene is recirculated to the first and fourth CSTRs. Additionally, 1,3-diethylbenzene is channeled into the final CSTR to enhance ethylbenzene production. The process is mathematically modeled using 30 ordinary differential equations (ODEs). These equations capture the system's key state variables, including holdup (V), temperature (T), and the concentrations of the four components in each CSTR and the separator^{110;111}. The control goal is to maintain specific steady-state values for holdup, temperature, and ethylbenzene concentration across the four CSTRs and the separator (15 controlled outputs). This is accomplished by adjusting 12 flow rates and 5 heat inputs to the process units (17 manipulated inputs). The appendix provides a detailed state-space model of the benzene alkylation process and the associated parameter

values.

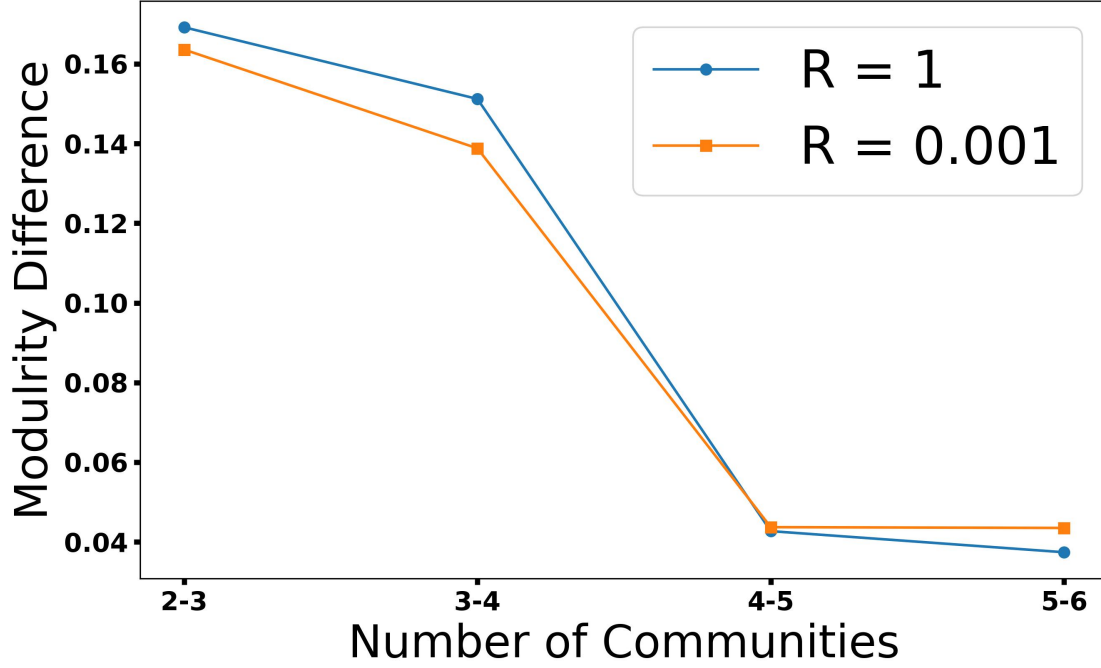
4.2 Distributed Control Structure

The case study explored two operational scenarios: moderate recycling (recycle ratio $R = \frac{F_{r1}}{F_{r2}} = 1$) and minimal recycling ($R = 0.001$). `rblueCommunities` are pre-established for each operational point. When operational changes occur, the community detection algorithm is rerun, enabling controllers to adapt. This process reorganizes communities into optimal clusters that reflect the new conditions. Tables 4.1 and 4.2 show steady-state values for all state variables and controlled inputs under these conditions. Figure 4.2 demonstrates the application of the elbow method to determine the optimal number of subsystems for distributed control architecture, revealing 3 subsystems with varying configurations for both operational points. System decomposition was achieved using an enhanced spectral community detection method. This method utilized a weighted adjacency matrix constructed from data in Table 3.1 and the state space model detailed in the appendix, applied to two distinct operational points. The resulting decompositions were then compared to those obtained using an unweighted adjacency matrix, which remains constant across all operational points, as reported in ¹¹⁰.

Table 4.3 presents a detailed allocation of manipulated inputs (in red) and controlled outputs (in blue) among subsystems for each decomposition, alongside their respective modularity values. The enhanced modularity observed in spectral decompositions derived from weighted graph representations indicates a superior distributed architecture based on weighted networks. Figure 4.2 depicts the resulting multi-agent distributed control structures.

Table 4.1: *Steady state values of the state variables.*

States	R=1	R=0.001
$V_1(m^3)$	1	1
$V_2(m^3)$	1	1
$V_3(m^3)$	1	1
$V_4(m^3)$	3	3
$V_5(m^3)$	1	1
$T_1(K)$	474.36	481.95
$T_2(K)$	473.65	480.36
$T_3(K)$	469.76	472.87
$T_4(K)$	465.42	484.19
$T_5(K)$	467.58	472.81
$C_{A1}(mol/m^3)$	9.101×10^3	8.155×10^3
$C_{B1}(mol/m^3)$	22.380	22.530
$C_{C1}(mol/m^3)$	1.116×10^3	1.564×10^3
$C_{D1}(mol/m^3)$	3.126×10^2	3.126×10^2
$C_{A2}(mol/m^3)$	7.548×10^3	5.636×10^3
$C_{B2}(mol/m^3)$	23.730	24.734
$C_{C2}(mol/m^3)$	1.906×10^3	2.854×10^3
$C_{D2}(mol/m^3)$	3.731×10^2	5.552×10^2
$C_{A3}(mol/m^3)$	6.163×10^3	3.585×10^3
$C_{B3}(mol/m^3)$	25.160	27.746
$C_{C3}(mol/m^3)$	2.614×10^3	3.881×10^3
$C_{D3}(mol/m^3)$	5.056×10^2	7.690×10^2
$C_{A4}(mol/m^3)$	1.756×10^3	1.720×10^3
$C_{B4}(mol/m^3)$	14.030	12.253
$C_{C4}(mol/m^3)$	5.402×10^3	5.472×10^3
$C_{D4}(mol/m^3)$	7.396×10^2	7.05×10^2
$C_{A5}(mol/m^3)$	5.805×10^3	5.279×10^3
$C_{B5}(mol/m^3)$	4.630	4.845
$C_{C5}(mol/m^3)$	3.716×10^3	4.110×10^3
$C_{D5}(mol/m^3)$	2.091×10^2	1.987×10^2



caption Elbow method for $R = 1$ and $R = 0.001$.

Table 4.2: *Steady state values of the manipulated inputs.*

Inputs	R=1	R=0.001
$F_1(m^3/s)$	7.1×10^{-3}	7.1×10^{-3}
$F_2(m^3/s)$	8.967×10^{-4}	8.967×10^{-4}
$F_3(m^3/s)$	1.4×10^{-2}	8×10^{-3}
$F_4(m^3/s)$	8.967×10^{-4}	8.967×10^{-4}
$F_5(m^3/s)$	1.48×10^{-2}	8.8×10^{-3}
$F_6(m^3/s)$	8.967×10^{-4}	8.967×10^{-4}
$F_7(m^3/s)$	1.57×10^{-2}	9.7×10^{-3}
$F_8(m^3/s)$	1.2×10^{-2}	1.2×10^{-2}
$F_9(m^3/s)$	8.3×10^{-3}	8.3×10^{-3}
$F_{10}(m^3/s)$	2.3×10^{-3}	2.3×10^{-3}
$F_{r1}(m^3/s)$	6×10^{-3}	6×10^{-3}
$F_{r2}(m^3/s)$	6×10^{-3}	6×10^{-6}
$Q_1(J/s)$	-4.4×10^6	-4.4×10^6
$Q_2(J/s)$	-4.6×10^6	-4.6×10^6
$Q_3(J/s)$	-4.7×10^6	-4.7×10^6
$Q_4(J/s)$	9.2×10^6	9.2×10^6
$Q_5(J/s)$	5.6×10^6	5.6×10^6

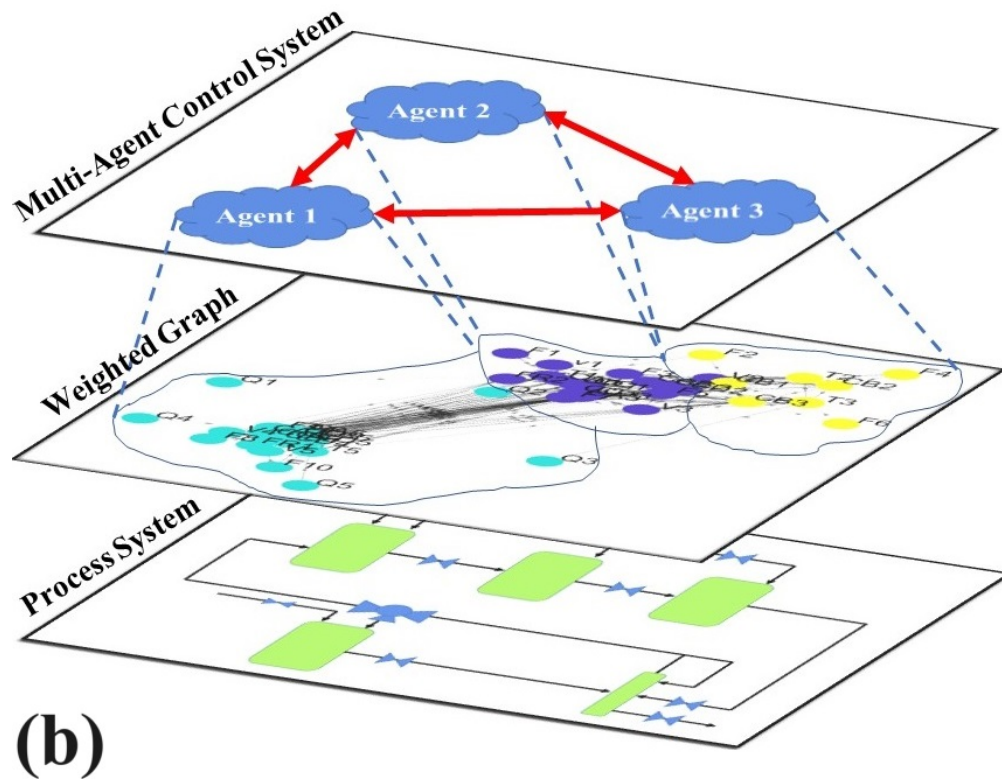
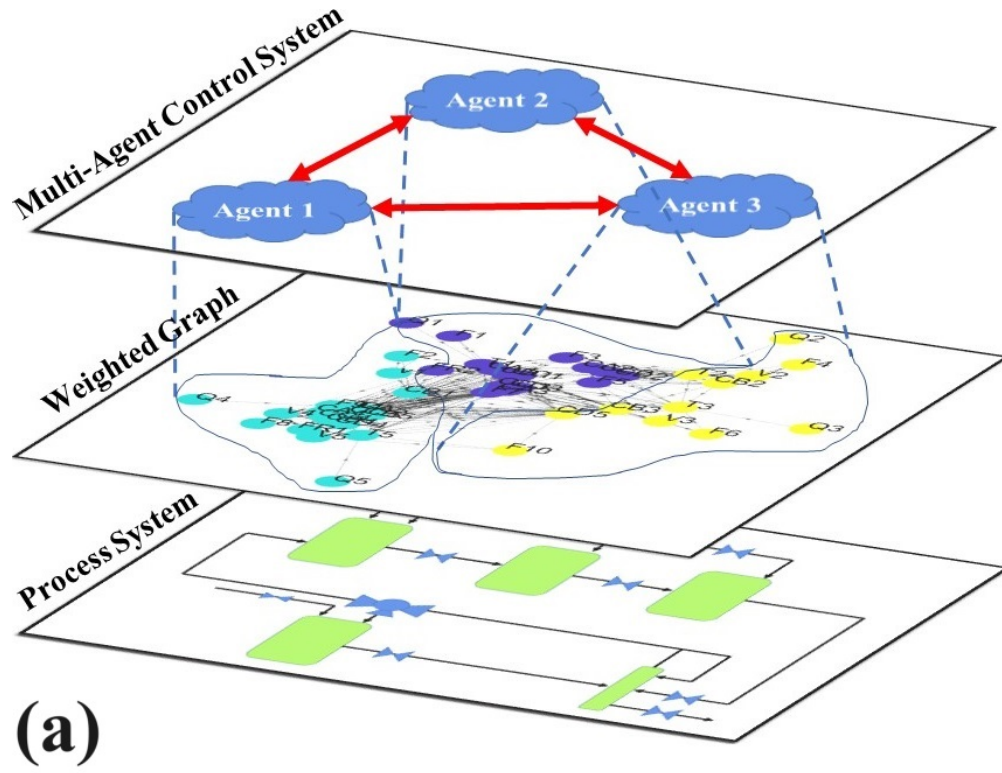


Figure 4.2: Multi-agent distributed control architectures for (a) $R = 1$ and (b) $R = 0.001$.

Table 4.3: *Distributed control structures.*

Subsystem	Weighted (R=0.001)	Weighted (R=1)	Unweighted
C_1	$F_1 F_3 F_5$ $F_7 F_{R_2}$ $V_3 T_1$ $C_{C_1} C_{C_2} C_{C_3}$	$F_1 F_3 F_5$ $F_7 F_{R_2}$ $C_{C_1} C_{C_2}$ $T_1 C_{C_3}$	$F_1 F_2 F_3$ $F_4 F_{R_2} Q_1$ $V_1 V_2 C_{C_2}$ $T_1 T_2 C_{C_1}$
C_2	$F_2 F_4$ $V_2 T_2$	$F_4 F_6 F_{10}$ $V_2 T_2 T_3$	$F_5 F_6 F_7$ $Q_3 V_3 T_3$ C_{C_3}
C_3	$F_6 F_8 F_9$ $F_{10} F_{R_1} Q_1$ $Q_2 Q_3 Q_4$ $Q_5 V_4 V_5$ $T_3 T_4 T_5$ $V_1 C_{C_4} C_{C_5}$	$F_2 F_8 F_9$ $F_{R_1} Q_1 Q_2$ $Q_3 Q_4 Q_5$ $V_3 V_4 V_5$ $T_4 T_5 C_{C_4}$ $C_{C_4} V_1$	$F_8 F_9 F_{10}$ $F_{R_1} Q_4 Q_5$ T_4 $V_4 V_5 T_4$ $T_5 C_{C_4} C_{C_5}$
Modularity	0.6025	0.6017	0.2953

Chapter 5

Simulation Result

We have developed iterative DMPC structures based on the system decompositions outlined in Section 4.2 to compare community detection results between weighted and unweighted networks. The number of agents in each DMPC setup corresponds to the number of subsystems, as shown in Table 4.3. We assume all state variables are measurable in real-time throughout the process. The prediction horizon is set to 10 minutes with a 1-minute sampling interval, and we use $N = 10$ samples within this horizon. The diagonal elements of matrices $P_{(i)}$ and $W_{(i)}$ are limited to $[0, 1]$ as all variables are dimensionless and pre-scaled. These values indicate each variable’s relative importance in the objective function. This approach ensures appropriate weighting of different system components during optimization, regardless of their original units or magnitudes. Control action bounds are set to 20% of steady-state values. For outputs with non-self-regulating dynamics (holdups), we developed 5 nonlinear constraints based on the nearest inputs within the distributed architecture. We used IPOPT¹¹² to solve the optimization problem in each iteration. The closed-loop system’s performance was consistently evaluated by monitoring holdups, temperatures, and ethylbenzene concentrations. Figures 5.2 and 5.1 show temporal profiles of select manipulated inputs and outputs for medium recycling. Figures 5.4 and 5.3 display similar profiles for low recycling.

The figures illustrate that state regulation errors smoothly converge to zero, while manip-

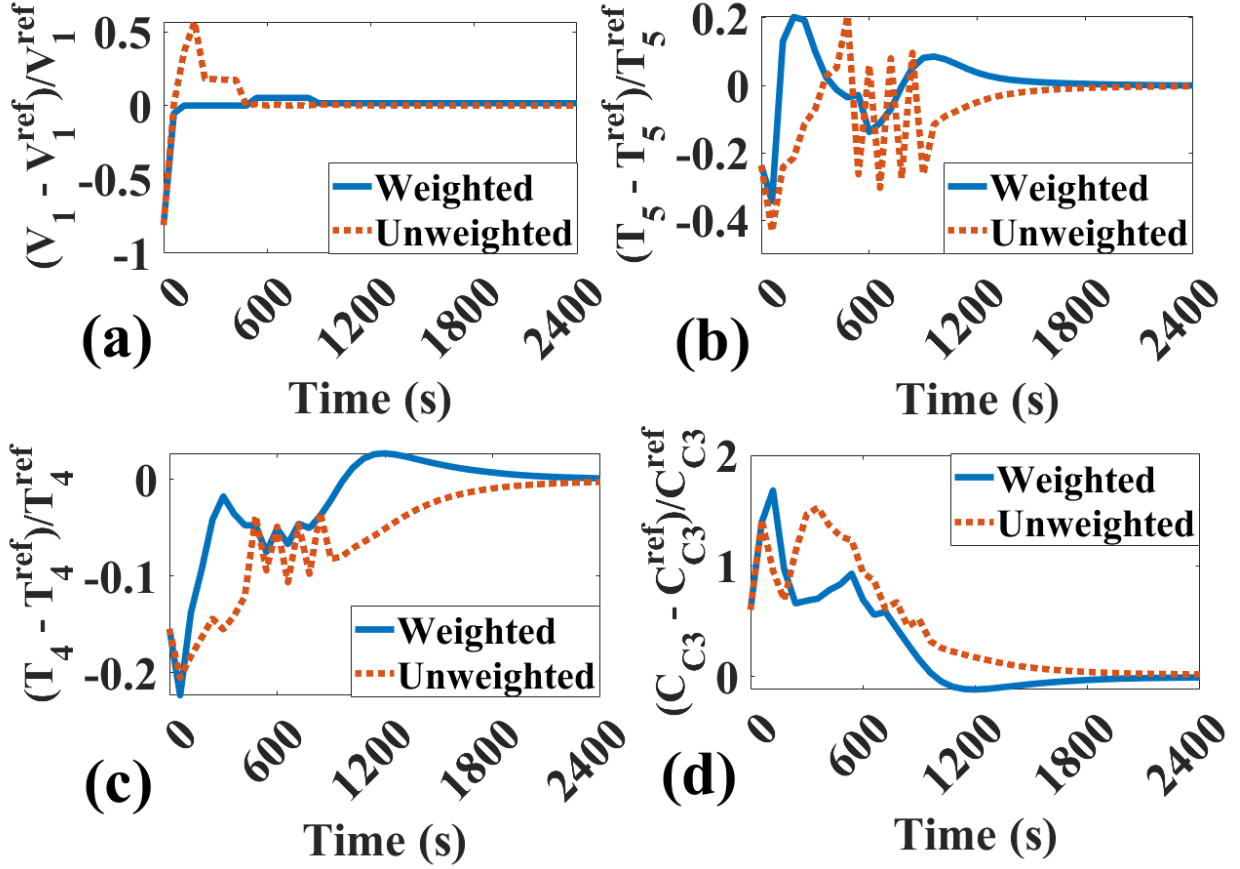


Figure 5.1: Selected temporal profiles of the state variables under DMPC based on the decomposition obtained weighted graph representation (for $R = 1$) and unweighted graph representation: (a) holdup in the first CSTR, (b) temperature in fourth CSTR, (c) temperature in the separator, and (d) concentration of desired product in the third CSTR.

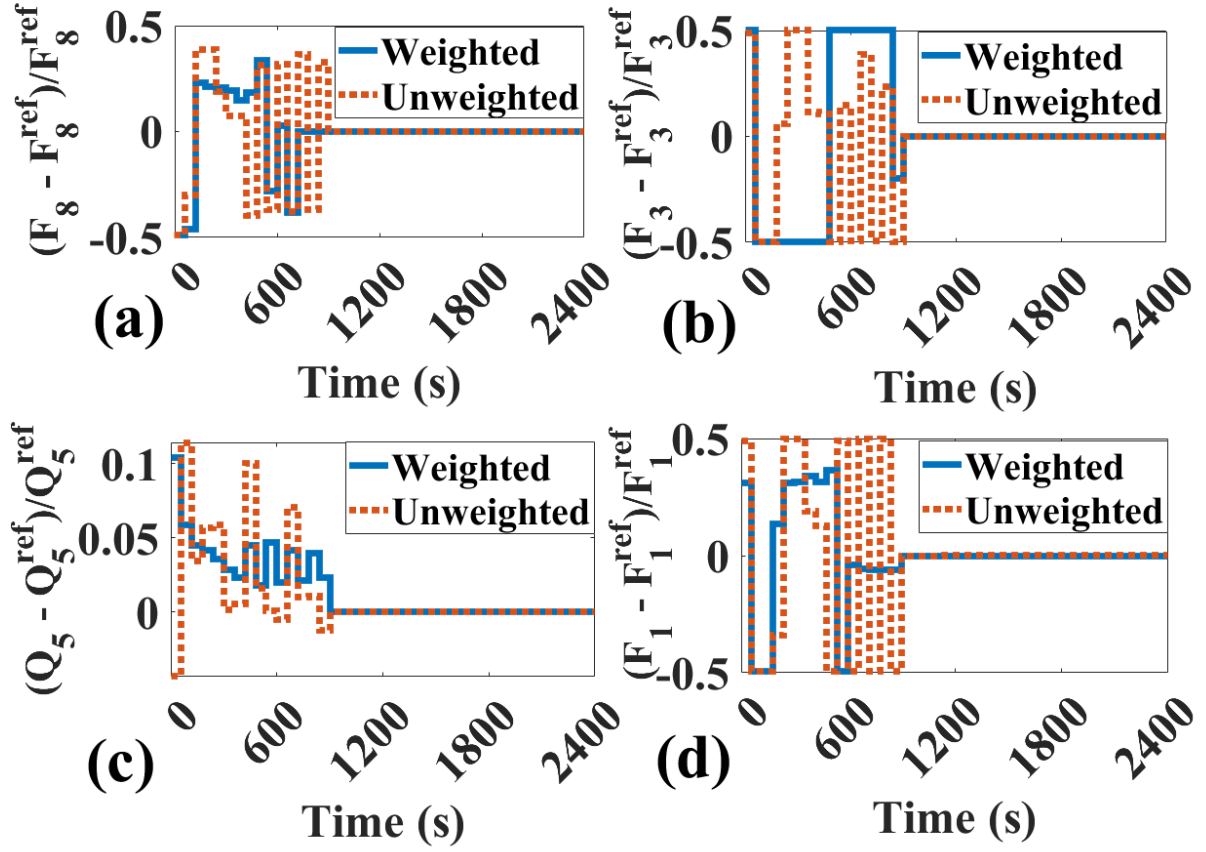


Figure 5.2: Temporal profiles of selected manipulated inputs under DMPC based on the decomposition obtained from weighted graph representation (for $R = 1$) and unweighted graph representation: (a) effluent flow of separator, (b) outlet flow from first CSTR to second CSTR, (c) heat to the fourth CSTR, and (d) feed to first CSTR.

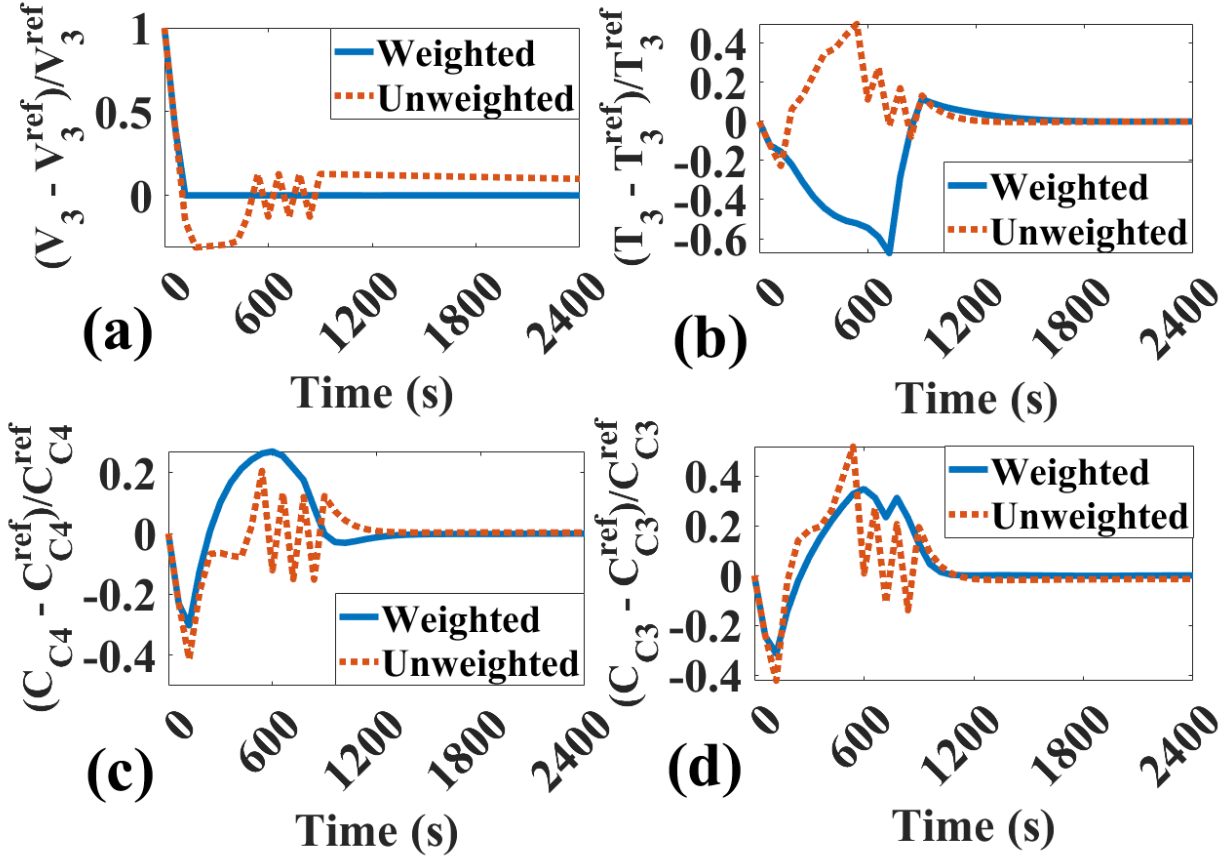


Figure 5.3: Temporal profiles of selected state variables under DMPC based on the decomposition obtained from weighted graph representation (for $R = 0.001$) and unweighted graph representation: (a) holdup in the third CSTR, (b) temperature in third CSTR, (c) concentration of desire product in the separator, and (d) concentration of desired product in the third CSTR.

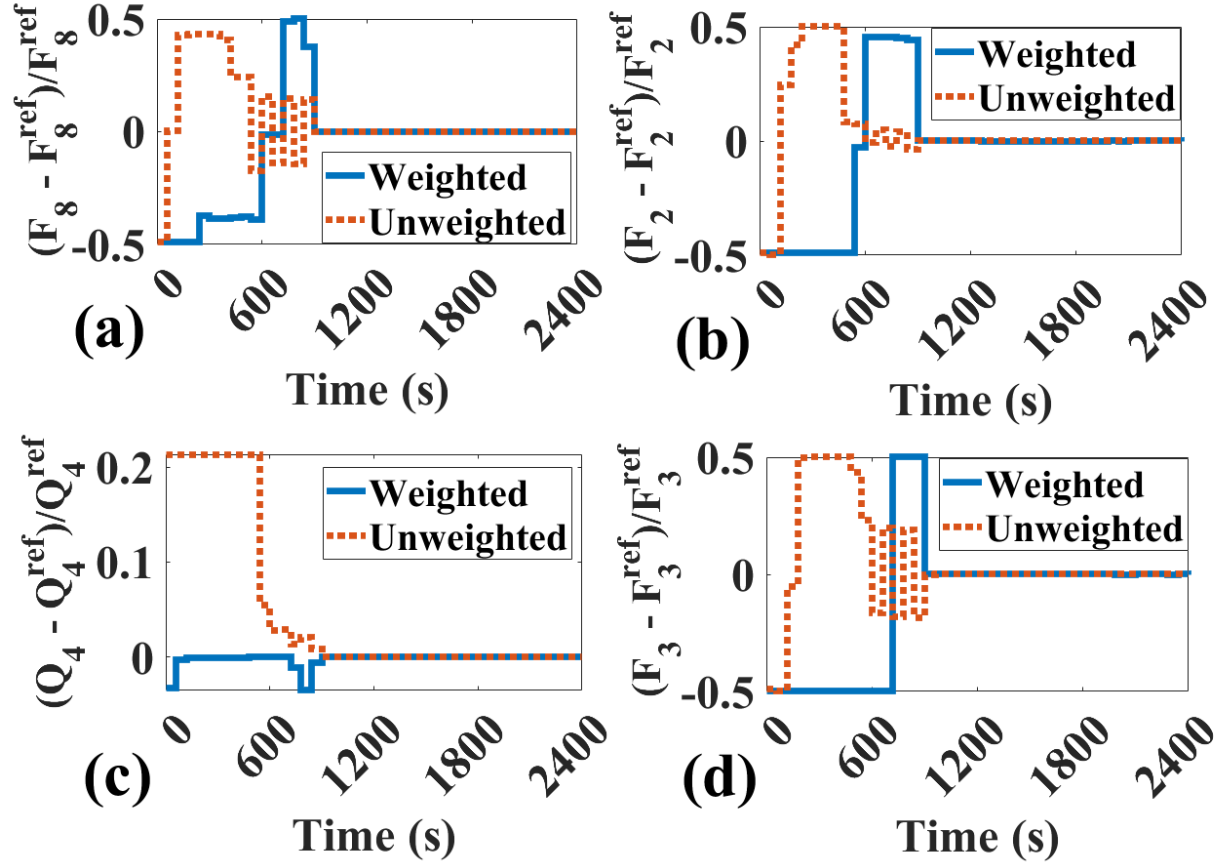


Figure 5.4: Temporal profiles of selected manipulated inputs under DMPC based on the decomposition obtained from weighted graph representation (for $R = 0.001$) and unweighted graph representation: (a) effluent flow of separator, (b) feed to first CSTR, (c) heat to the separator, and (d) effluent flow of first CSTR.

ulated inputs gradually approach their steady-state values. We’ve employed a dimensionless performance index (DPI), as defined in equation 5.1, which combines the integral square error (ISE) and the integral square control action (ISC) for each simulation¹¹⁰. Tables 5.1 and 5.2 provide a comprehensive performance evaluation for the two distinct operating conditions.

$$DPI = ISE + ISC = \int_0^T (\|y(t) - y^*\|^2 + \|u(t) - u^*\|^2) dt. \quad (5.1)$$

Table 5.1: *Closed-loop performance for $R = 1$.*

Network/Index	ISE	ISC	ISE+ISC
Weighted	3.12×10^3	6.51×10^2	3.77×10^3
Unweighted	4.11×10^3	1.05×10^3	5.16×10^3
Improvement	24%	38%	26.9%

Table 5.2: *Closed-loop performance for $R = 0.001$.*

Network/Index	ISE	ISC	ISE+ISC
Weighted	8.42×10^2	7.6×10^2	1.6×10^3
Unweighted	9.04×10^2	1.19×10^3	2.09×10^3
Improvement	6.8%	35.9%	23.3%

For the medium recycling scenario ($R = 1$), the DMPC using weighted graph decomposition shows a 24% enhancement in output regulation and a 38% decrease in control action energy. In the low recycling condition, the weighted graph-based DMPC exhibits improvements of 6.8% in output regulation and 35.9% in control action energy. These performance gains stem from weighted decomposition, which allows each controller agent to make decisions with reduced interaction with other agents.

Chapter 6

Conclusion

This report provides a comprehensive review of control strategies for integrated processes and community detection, extending the current state-of-the-art distributed control architecture based on unweighted graphs. Specifically, the research introduces an innovative technique for multi-agent DMPC, leveraging an enhanced spectral community detection method rooted in weighted control networks. A mathematical framework was developed to determine network weights for general state space models. The approach dynamically assigns controlled outputs and manipulated inputs to controller agents, adaptively defining their interactions based on specific operational points. The method's effectiveness was thoroughly evaluated and validated through its application to the closed-loop performance of benzene alkylation processes, demonstrating significant improvements in control efficiency and adaptability under varying operating conditions.

Bibliography

- [1] Michael Baldea. From process integration to process intensification. *Computers & Chemical Engineering*, 81:104–114, 2015.
- [2] Jiří Jaromír Klemeš, Petar Sabev Varbanov, and Zdravko Kravanja. Recent developments in process integration. *Chemical Engineering Research and Design*, 91(10):2037–2053, 2013.
- [3] Michael Baldea and Prodromos Daoutidis. *Dynamics and nonlinear control of integrated process systems*. Cambridge University Press, 2012.
- [4] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5-6):1327–1349, 2021.
- [5] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.
- [6] Rodrigo Lopez-Negrete, Fernando J D’Amato, Lorenz T Biegler, and Aditya Kumar. Fast nonlinear model predictive control: Formulation and industrial process applications. *Computers & Chemical Engineering*, 51:55–64, 2013.
- [7] Yang Wang and Stephen Boyd. Fast model predictive control using online optimization. *IEEE Transactions on control systems technology*, 18(2):267–278, 2009.
- [8] Gabriele Pannocchia, James B Rawlings, and Stephen J Wright. Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43(5):852–860, 2007.
- [9] Elizabeth A Leicht and Mark EJ Newman. Community structure in directed networks. *Physical review letters*, 100(11):118703, 2008.

- [10] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.
- [11] James B Rawlings and Bhavik R Bakshi. Particle filtering and moving horizon estimation. *Computers & chemical engineering*, 30(10-12):1529–1541, 2006.
- [12] Christopher V Rao, James B Rawlings, and David Q Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE transactions on automatic control*, 48(2):246–258, 2003.
- [13] Xunyuan Yin, Benjamin Decardi-Nelson, and Jinfeng Liu. Subsystem decomposition and distributed moving horizon estimation of wastewater treatment plants. *Chemical Engineering Research and Design*, 134:405–419, 2018.
- [14] Xunyuan Yin and Jinfeng Liu. Subsystem decomposition of process networks for simultaneous distributed state estimation and control. *AIChE Journal*, 65(3):904–914, 2019.
- [15] Davood B Pourkargar, Manjiri Moharir, Ali Almansoori, and Prodromos Daoutidis. Distributed estimation and nonlinear model predictive control using community detection. *Industrial & Engineering Chemistry Research*, 58(30):13495–13507, 2019.
- [16] Davood B Pourkargar, Ali Almansoori, and Prodromos Daoutidis. Distributed model predictive control of process networks: Impact of control architecture. *IFAC-PapersOnLine*, 50(1):12452–12457, 2017.
- [17] Davood B Pourkargar, Ali Almansoori, and Prodromos Daoutidis. Impact of decomposition on distributed model predictive control: A process network case study. *Industrial & Engineering Chemistry Research*, 56(34):9606–9616, 2017.
- [18] AmirMohammad Ebrahimi and Davood B. Pourkargar. Multi-agent distributed control of integrated process networks using an adaptive community detection approach. *Digital Chemical Engineering Journal*, 2024.

- [19] Xiao Zhang and Mark EJ Newman. Multiway spectral community detection in networks. *Physical Review E*, 92(5):052808, 2015.
- [20] Tom Van Gerven and Andrzej Stankiewicz. Structure, energy, synergy, time the fundamentals of process intensification. *Industrial & engineering chemistry research*, 48(5):2465–2474, 2009.
- [21] David Reay, Colin Ramshaw, and Adam Harvey. *Process Intensification: Engineering for efficiency, sustainability and flexibility*. Butterworth-Heinemann, 2013.
- [22] Panagiotis D Christofides, Riccardo Scattolini, David Munoz de la Pena, and Jin-feng Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.
- [23] Babatunde Ayodeji Ogunnaike and W Harmon Ray. Process dynamics, modeling, and control. (*No Title*), 1994.
- [24] Jose A Romagnoli and Ahmet Palazoglu. *Introduction to process control*. CRC press, 2005.
- [25] Dale E Seborg, Thomas F Edgar, Duncan A Mellichamp, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 2016.
- [26] Juan Pablo Manzo Hernández, Julio César Solano Vargas, Juan Manuel Barrera Fernández, Daniel Moreno Orduña, and Cristian Hamilton Sánchez Saquí. A novel adaptive pid controller with new seesaw algorithms using alternative derivatives. *Systems Science & Control Engineering*, 12(1):2363625, 2024.
- [27] Chao Liu, XiaoXia Qiu, and Yao Mao. The pid high-type controller based on lqr and model-assisted leso. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, page 09596518241263538, 2024.
- [28] Panagiotis D Christofides and Nael El-Farra. *Control of nonlinear and hybrid process*

- systems: Designs for uncertainty, constraints and time-delays*, volume 324. Springer Science & Business Media, 2005.
- [29] Michael A Henson. Nonlinear model predictive control: current status and future directions. *Computers & Chemical Engineering*, 23(2):187–202, 1998.
 - [30] James B Rawlings. Tutorial: Model predictive control technology. In *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, volume 1, pages 662–676. IEEE, 1999.
 - [31] Frank Allgower, Rolf Findeisen, Zoltan K Nagy, et al. Nonlinear model predictive control: From theory to application. *Journal-Chinese Institute Of Chemical Engineers*, 35(3):299–316, 2004.
 - [32] Feiran Sun, Tao Liu, Bo Song, Yan Cui, Zoltan K Nagy, and Rolf Findeisen. Multi-objective optimization based nonlinear model predictive control of seeded cooling crystallization process with application to β form l-glutamic acid. *Chemical Engineering Science*, page 120475, 2024.
 - [33] Rodrigo Lopez-Negrete, Fernando J. D’Amato, Lorenz T. Biegler, and Aditya Kumar. Fast nonlinear model predictive control: Formulation and industrial process applications. *Computers and Chemical Engineering*, 51:55–64, 4 2013. ISSN 00981354. doi: 10.1016/j.compchemeng.2012.06.011.
 - [34] Eduardo Camponogara, Dong Jia, Bruce H Krogh, and Sarosh Talukdar. Distributed model predictive control. *IEEE control systems magazine*, 22(1):44–52, 2002.
 - [35] Rudy R Negenborn and Jose Maria Maestre. Distributed model predictive control: An overview and roadmap of future research opportunities. *IEEE Control Systems Magazine*, 34(4):87–97, 2014.
 - [36] Matthias A Müller and Frank Allgöwer. Economic and distributed model predictive control: Recent developments in optimization-based control. *SICE Journal of Control, Measurement, and System Integration*, 10(2):39–52, 2017.

- [37] Walid Al-Gherwi, Hector Budman, and Ali Elkamel. Robust distributed model predictive control: A review and recent developments. *The Canadian Journal of Chemical Engineering*, 89(5):1176–1190, 2011.
- [38] Darioush Razmi, Oluleke Babayomi, Alireza Davari, Tohid Rahimi, Yuntao Miao, and Zhenbin Zhang. Review of model predictive control of distributed energy resources in microgrids. *Symmetry*, 14(8):1735, 2022.
- [39] Yuandan Lin, Eduardo D Sontag, and Yuan Wang. A smooth converse lyapunov theorem for robust stability. *SIAM Journal on Control and Optimization*, 34(1):124–160, 1996.
- [40] José L Massera. Contributions to stability theory. *Annals of Mathematics*, 64(1):182–206, 1956.
- [41] Ping Zhou, Xikui Hu, Zhigang Zhu, and Jun Ma. What is the most suitable lyapunov function? *Chaos, Solitons & Fractals*, 150:111154, 2021.
- [42] James B Rawlings, Dennis Bonné, John B Jorgensen, Aswin N Venkat, and Sten Bay Jorgensen. Unreachable setpoints in model predictive control. *IEEE Transactions on Automatic Control*, 53(9):2209–2215, 2008.
- [43] Riccardo Scattolini. Architectures for distributed and hierarchical model predictive control—a review. *Journal of process control*, 19(5):723–731, 2009.
- [44] Arthur Richards and Jonathan P How. Robust distributed model predictive control. *International Journal of control*, 80(9):1517–1531, 2007.
- [45] Tamás Keviczky, Francesco Borrelli, and Gary J Balas. Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42(12):2105–2115, 2006.
- [46] William B Dunbar and Richard M Murray. Distributed receding horizon control with application to multi-vehicle formation stabilization. 2004.

- [47] William B Dunbar and Subhas Desa. Distributed mpc for dynamic supply chain management. In *Assessment and future directions of nonlinear model predictive control*, pages 607–615. Springer, 2007.
- [48] Mehmet Mercangöz and Francis J Doyle III. Distributed model predictive control of an experimental four-tank system. *Journal of process control*, 17(3):297–308, 2007.
- [49] Aswin N Venkat, James B Rawlings, and Stephen J Wright. Stability and optimality of distributed model predictive control. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6680–6685. IEEE, 2005.
- [50] Jiefeng Hu, Yinghao Shan, Josep M Guerrero, Adrian Ioinovici, Ka Wing Chan, and Jose Rodriguez. Model predictive control of microgrids—an overview. *Renewable and Sustainable Energy Reviews*, 136:110422, 2021.
- [51] Ying Ding, Liang Wang, Yongwei Li, and Daoliang Li. Model predictive control and its application in agriculture: A review. *Computers and Electronics in Agriculture*, 151:104–117, 2018.
- [52] Davood B. Pourkargar, Ali Almansoori, and Prodromos Daoutidis. Distributed model predictive control of process networks: Impact of control architecture. volume 50, pages 12452–12457. Elsevier B.V., 7 2017. doi: 10.1016/j.ifacol.2017.08.1920.
- [53] Jiakang Li, Songning Lai, Zhihao Shuai, Yuan Tan, Yifan Jia, Mianyang Yu, Zichen Song, Xiaokang Peng, Ziyang Xu, Yongxin Ni, et al. A comprehensive review of community detection in graphs. *Neurocomputing*, page 128169, 2024.
- [54] Carlos A Gomez-Urbe and George C Verghese. Dynamic community detection in power system network models. In *2011 North American Power Symposium*, pages 1–6. IEEE, 2011.
- [55] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

- [56] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.
- [57] Shaowei Chen, Youjian Feng, and Jiming Liu. Optimization-based community detection in complex networks using semidefinite programming. *Journal of Global Optimization*, 63(4):851–867, 2015.
- [58] Mark E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103:8577–8582, 6 2006. ISSN 00278424. doi: 10.1073/pnas.0601602103.
- [59] Mark E.J. Newman. Spectral methods for community detection and graph partitioning. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 88, 10 2013. ISSN 15393755. doi: 10.1103/PhysRevE.88.042822.
- [60] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media: Performance and application considerations. *Data mining and knowledge discovery*, 24:515–554, 2012.
- [61] Michael J Barber. Modularity and community detection in bipartite networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76(6):066102, 2007.
- [62] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.
- [63] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, 1(1):46–65, 2014.
- [64] Hoi-To Wai, Santiago Segarra, Asuman E Ozdaglar, Anna Scaglione, and Ali Jadbabaie. Blind community detection from low-rank excitations of a graph filter. *IEEE Transactions on signal processing*, 68:436–451, 2019.

- [65] Linli Xu, Wenye Li, and Dale Schuurmans. Fast normalized cut with linear constraints. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2866–2873. IEEE, 2009.
- [66] Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 298–301. IEEE, 1989.
- [67] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.
- [68] Zengyang Shao, Lijia Ma, Qiuzhen Lin, Jianqiang Li, Maoguo Gong, and Asoke K Nandi. Pmcdm: Privacy-preserving multiresolution community detection in multiplex networks. *Knowledge-Based Systems*, 244:108542, 2022.
- [69] Xinyu Que, Fabio Checconi, Fabrizio Petrini, and John A Gunnels. Scalable community detection with the louvain algorithm. In *2015 IEEE international parallel and distributed processing symposium*, pages 28–37. IEEE, 2015.
- [70] Sayan Ghosh, Mahantesh Halappanavar, Antonino Tumeo, Ananth Kalyanaraman, Hao Lu, Daniel Chavarria-Miranda, Arif Khan, and Assefaw Gebremedhin. Distributed louvain algorithm for graph community detection. In *2018 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 885–895. IEEE, 2018.
- [71] Vincent A Traag. Faster unfolding of communities: Speeding up the louvain algorithm. *Physical Review E*, 92(3):032801, 2015.
- [72] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.
- [73] Fabian Nguyen. *Leiden-based parallel community detection*. PhD thesis, Bachelor’s thesis. Karlsruhe Institute of Technology, 2021 (zitiert auf S. 31), 2021.

- [74] Subhajit Sahu. Addressing internally-disconnected communities in leiden and louvain community detection algorithms. *arXiv preprint arXiv:2402.11454*, 2024.
- [75] Subhajit Sahu. A starting point for dynamic community detection with leiden algorithm. *arXiv preprint arXiv:2405.11658*, 2024.
- [76] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.
- [77] Jianping Zeng and Hongfeng Yu. A distributed infomap algorithm for scalable and high-quality community detection. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–11, 2018.
- [78] Fang Hu and Yuhua Liu. A novel algorithm infomap-sa of detecting communities in complex networks. *J. Commun.*, 10(7):503–511, 2015.
- [79] Park K Fung. Infocflow: A distributed algorithm to detect communities according to the map equation. *Big Data and Cognitive Computing*, 3(3):42, 2019.
- [80] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.
- [81] Guiqiong Xu, Jiawen Guo, and Pingle Yang. Tns-lpa: an improved label propagation algorithm for community detection based on two-level neighbourhood similarity. *Ieee Access*, 9:23526–23536, 2020.
- [82] Vinícius da F Vieira, Carolina R Xavier, and Alexandre G Evsukoff. Efficient community detection in large scale networks. In *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, pages 669–674. IEEE, 2013.
- [83] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *ProQuest number: information to all users*, 2002.

- [84] Huan Li, Ruisheng Zhang, Zhili Zhao, and Xin Liu. Lpa-mni: an improved label propagation algorithm based on modularity and node importance for community detection. *Entropy*, 23(5):497, 2021.
- [85] Michael J Barber and John W Clark. Detecting network communities by propagating labels under constraints. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 80(2):026129, 2009.
- [86] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76(3):036106, 2007.
- [87] Jinfeng Liu, Xianzhong Chen, David Muñoz de la Peña, and Panagiotis D Christofides. Sequential and iterative architectures for distributed model predictive control of non-linear process systems. *AIChE Journal*, 56(8):2137–2149, 2010.
- [88] Jierui Xie and Boleslaw K Szymanski. Community detection using a neighborhood strength driven label propagation algorithm. In *2011 IEEE Network Science Workshop*, pages 188–195. IEEE, 2011.
- [89] Qingshou Wu, Rongwang Chen, Lijin Wang, and Kun Guo. A label propagation algorithm for community detection on high-mixed networks. *Concurrency and Computation: Practice and Experience*, 33(9):e6141, 2021.
- [90] Vincent A Traag and Lovro Šubelj. Large network community detection by fast label propagation. *Scientific Reports*, 13(1):2701, 2023.
- [91] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.
- [92] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

- [93] Alex Pothén, Horst D Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM journal on matrix analysis and applications*, 11(3):430–452, 1990.
- [94] Ulzii-Utas Narantsatsralt and Sanggil Kang. Social network community detection using agglomerative spectral clustering. *Complexity*, 2017(1):3719428, 2017.
- [95] Daniel A Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of 37th conference on foundations of computer science*, pages 96–105. IEEE, 1996.
- [96] Douglas O Cardoso, João Domingos Gomes da Silva Junior, Carla Silva Oliveira, Celso Marques, and Laura Silva de Assis. Greedy recursive spectral bisection for modularity-bound hierarchical divisive community detection. *Statistics and Computing*, 34(4):145, 2024.
- [97] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.
- [98] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 65–72, 2019.
- [99] Sujit S Jogwar. Distributed control architecture synthesis for integrated process networks through maximization of strength of input–output impact. *Journal of Process Control*, 83:77–87, 2019.
- [100] Wentao Tang and Prodromos Daoutidis. Network decomposition for distributed control through community detection in input–output bipartite graphs. *Journal of Process Control*, 64:7–14, 4 2018. ISSN 09591524. doi: 10.1016/j.jprocont.2018.01.009.
- [101] Davood B Pourkargar and Sujit S Jogwar. Distributed model predictive control of integrated process networks: Optimal decomposition for varying operating point. In *2021 American Control Conference (ACC)*, pages 801–807. IEEE, 2021.

- [102] Costas Kravaris and Jeffrey C Kantor. Geometric methods for nonlinear process control. 1. background. *Industrial & Engineering Chemistry Research*, 29(12):2295–2310, 1990.
- [103] Amirmohammad Ebrahimi and Davood B. Pourkargar. Distributed model predictive control of integrated process networks based on an adaptive community detection approach. In *2024 American Control Conference (ACC)*, pages 208–213. IEEE, 2024.
- [104] Amirmohammad Ebrahimi and Davood B Pourkargar. Distributed estimation and control of process networks using adaptive community detection. *IFAC-PapersOnLine*, 58(14):754–760, 2024.
- [105] Ulrik Brandes. *Network analysis: methodological foundations*, volume 3418. Springer Science & Business Media, 2005.
- [106] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
- [107] Deny Jollyta, Syahril Efendi, Muhammad Zarlis, and Herman Mawengkang. Analysis of an optimal cluster approach: a review paper. In *Journal of Physics: Conference Series*, volume 2421, page 012015. IOP Publishing, 2023.
- [108] AmirMohammad Ebrahimi and Davood B. Pourkargar. An adaptive distributed architecture for multi-agent state estimation and control of complex process systems. *Chemical Engineering research and design*, 2024.
- [109] Hongjun You, Wenyu Long, and Yi Pan. The mechanism and kinetics for the alkylation of benzene with ethylene. *Petroleum science and technology*, 24(9):1079–1088, 2006.
- [110] Davood B. Pourkargar, Manjiri Moharir, Ali Almansoori, and Prodromos Daoutidis. Distributed estimation and nonlinear model predictive control using community detection. *Industrial and Engineering Chemistry Research*, 58:13495–13507, 7 2019. ISSN 15205045. doi: 10.1021/acs.iecr.9b00820.

- [111] Jinfeng Liu, Xianzhong Chen, David Muñoz de la Peña, and Panagiotis D. Christofides. Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. *AIChE Journal*, 56:2137–2149, 8 2010. ISSN 00011541. doi: 10.1002/aic.12155.
- [112] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.
- [113] Panagiotis D. Christofides, Riccardo Scattolini, David Muñoz de la Peña, and Jinfeng Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers and Chemical Engineering*, 51:21–41, 4 2013. ISSN 00981354. doi: 10.1016/j.compchemeng.2012.05.011.
- [114] Won Jae Lee and Gilbert F Froment. Ethylbenzene dehydrogenation into styrene: kinetic modeling and reactor simulation. *Industrial & engineering chemistry research*, 47(23):9183–9194, 2008.

Appendix A

Process Model Description

Appendix: Process Model Description

Applying standard modeling assumptions^{110;113;114} for material and energy balance, a set of ordinary differential equations (ODEs) can be derived to describe the dynamics of holdup, temperature, and the component concentrations within the CSTRs and separator,

$$\begin{aligned}
\frac{dV}{dt} &= F_1 + F_2 + F_{r2} - F_3 \\
\frac{dT_1}{dt} &= \frac{Q_1 + F_1 C_{A0} H_A(T_{A0}) + F_2 C_{B0} H_B(T_{B0}) + \sum_i^{A,B,C,D} F_{r2} C_{ir} H_i(T_4) - F_3 C_{i1} H_i(T_1)}{\sum_i^{A,B,C,D} C_{i1} C_{pi} V_1} \\
&\quad + \frac{-\Delta H_{r1} r_1(T_1, C_{A1}, C_{B1}) - \Delta H_{r2} r_2(T_1, C_{B1}, C_{C1}, C_{D1})}{\sum_i^{A,B,C,D} C_{i1} C_{pi}} \\
\frac{dC_{A1}}{dt} &= \frac{F_1 C_{A0} + F_{r2} C_{Ar} - F_3 C_{A1}}{V_1} - r_1(T_1, C_{A1}, C_{B1}) \\
\frac{dC_{B1}}{dt} &= \frac{F_2 C_{B0} + F_{r2} C_{Br} - F_3 C_{B1}}{V_1} - r_1(T_1, C_{A1}, C_{B1}) - r_2(T_1, C_{B1}, C_{C1}, C_{D1}) \\
\frac{dC_{C1}}{dt} &= \frac{F_{r2} C_{Cr} - F_3 C_{C1}}{V_1} + r_1(T_1, C_{A1}, C_{B1}) - r_2(T_1, C_{B1}, C_{C1}, C_{D1}) \\
\frac{dC_{D1}}{dt} &= \frac{F_{r2} C_{Dr} - F_3 C_{D1}}{V_1} + r_2(T_1, C_{B1}, C_{C1}, C_{D1}) \\
\frac{dV_2}{dt} &= F_3 + F_4 - F_5 \\
\frac{dT_2}{dt} &= \frac{Q_2 + F_4 C_{B0} H_B(T_{B0}) + \sum_i^{A,B,C,D} F_3 C_{i1} H_i(T_1) - F_5 C_{i2} H_i(T_2)}{\sum_i^{A,B,C,D} C_{i2} C_{pi} V_2} \\
&\quad + \frac{-\Delta H_{r1} r_1(T_2, C_{A2}, C_{B2}) - \Delta H_{r2} r_2(T_2, C_{A2}, C_{B2}, C_{D2})}{\sum_i^{A,B,C,D} C_{i2} C_{pi}} \\
\frac{dC_{A2}}{dt} &= \frac{F_3 C_{A1} - F_5 C_{A2}}{V_2} - r_1(T_2, C_{A2}, C_{B2}) \\
\frac{dC_{B2}}{dt} &= \frac{F_3 C_{B1} + F_4 C_{B0} - F_5 C_{B2}}{V_2} - r_1(T_2, C_{A2}, C_{B2}) - r_2(T_2, C_{B2}, C_{C2}, C_{D2}) \\
\frac{dC_{C2}}{dt} &= \frac{F_3 C_{C1} - F_5 C_{C2}}{V_2} + r_1(T_2, C_{A2}, C_{B2}) - r_2(T_2, C_{B2}, C_{C2}, C_{D2}) \\
\frac{dC_{D2}}{dt} &= \frac{F_3 C_{D1} - F_5 C_{D2}}{V_2} + r_2(T_2, C_{B2}, C_{C2}, C_{D2}) \\
\frac{dV_3}{dt} &= F_5 + F_6 - F_7 \\
\frac{dT_3}{dt} &= \frac{Q_3 + F_6 C_{B0} H_B(T_{B0}) + \sum_i^{A,B,C,D} F_5 C_{i2} H_i(T_2) - F_7 C_{i3} H_i(T_3)}{\sum_i^{A,B,C,D} C_{i3} C_{pi} V_3} \\
&\quad + \frac{-\Delta H_{r1} r_1(T_3, C_{A3}, C_{B3}) - \Delta H_{r2} r_2(T_3, C_{A3}, C_{B3}, C_{D3})}{\sum_i^{A,B,C,D} C_{i3} C_{pi}} \\
\frac{dC_{A3}}{dt} &= \frac{F_5 C_{A2} - F_7 C_{A3}}{V_3} - r_1(T_3, C_{A3}, C_{B3}) \\
\frac{dC_{B3}}{dt} &= \frac{F_5 C_{B2} + F_6 C_{B0} - F_7 C_{B3}}{V_3} - r_1(T_3, C_{A3}, C_{B3}) - r_2(T_3, C_{B3}, C_{C3}, C_{D3}) \\
\frac{dC_{C3}}{dt} &= \frac{F_5 C_{C2} - F_7 C_{C3}}{V_3} + r_1(T_3, C_{A3}, C_{B3}) - r_2(T_3, C_{B3}, C_{C3}, C_{D3}) \\
\frac{dC_{D3}}{dt} &= \frac{F_5 C_{D2} - F_7 C_{D3}}{V_3} + r_2(T_3, C_{B3}, C_{C3}, C_{D3})
\end{aligned}$$

$$\begin{aligned}
r_1(T, C_A, C_B) &= 0.0840 \exp\left(\frac{-9502}{RT}\right) C_A^{0.32} C_B^{1.5} \\
r_2(T, C_B, C_C, C_D) &= \frac{0.0850 \exp\left(\frac{-20643}{RT}\right) C_B^{2.5} C_C^{0.5}}{1 + k_{EB2} C_D} \\
r_3(T, C_A, C_D) &= \frac{66.1 \exp\left(\frac{-61280}{RT}\right) C_A^{1.0218} C_D}{1 + k_{EB3} C_A}
\end{aligned} \tag{A2}$$

where

$$\begin{aligned}
k_{EB2} &= 0.152 \exp\left(\frac{-3933}{RT}\right) \\
k_{EB3} &= 0.490 \exp\left(\frac{-50870}{RT}\right)
\end{aligned} \tag{A3}$$

Variables	Definition
F_3, F_5, F_7, F_9	flow rates of the streams leaving the units
$F_1, F_2, F_4, F_6, F_{10}$	flow rates of the streams coming to the units
F_r, F_{r1}, F_{r2}	flow rates of the recycle streams
Q_1, Q_2, Q_3, Q_4, Q_5	external heat flows to each units
$C_{A1}, C_{B1}, C_{C1}, C_{D1}$	concentrations of A, B, C, D in the first CSTR
$C_{A2}, C_{B2}, C_{C2}, C_{D2}$	concentrations of A, B, C, D in the second CSTR
$C_{A3}, C_{B3}, C_{C3}, C_{D3}$	concentrations of A, B, C, D in the third CSTR
$C_{A4}, C_{B4}, C_{C4}, C_{D4}$	concentrations of A, B, C, D in the flash tank separator
$C_{A5}, C_{B5}, C_{C5}, C_{D5}$	concentrations of A, B, C, D in the fourth CSTR
$C_{Ar}, C_{Br}, C_{Cr}, C_{Dr}$	concentrations of A, B, C, D in the recycle stream
V_1, V_2, V_3, V_4, V_5	hold-ups in each unit
T_1, T_2, T_3, T_4, T_5	temperatures in each unit
T_{ref}	reference temperature
$H_{\text{vap},A}, H_{\text{vap},B}, H_{\text{vap},C}, H_{\text{vap},D}$	enthalpies of vaporization of A, B, C, D in the first CSTR
$H_{\text{vap},A}, H_{\text{vap},B}, H_{\text{vap},C}, H_{\text{vap},D}$	enthalpies of vaporization of A, B, C, D in the first CSTR
$\Delta H_{r1}, \Delta H_{r2}, \Delta H_{r3}$	heat of reactions 1, 2, 3
T_{A0}, T_{B0}, T_{D0}	temperatures of the pure feed streams

Table A.1: Variables and their definitions in the process flow diagram.

Assuming linear dependency of the enthalpies of components on the temperature, we have

$$H_i(T) = H_{\text{ref},i} + C_{pi}(T - T_{\text{ref}}), \quad i = A, B, C, D \tag{A4}$$

where C_{pi} denotes the constant heat capacity of component i , and $H_{ref,i}$ (for $i = A, B, C, D$) signifies the enthalpies of the species at the reference temperature, T_{ref} . Additionally, we hypothesize a linear relationship between the separator's temperature and the species' relative volatilities.

$$\begin{aligned}
\alpha_A &= 0.0449T_4 + 10 \\
\alpha_A &= 0.0260T_4 + 10 \\
\alpha_A &= 0.0065T_4 + 0.5 \\
\alpha_A &= 0.0058T_4 + 0.25
\end{aligned} \tag{A5}$$

The concentration of species within the recycle stream can be determined by

$$C_{ir} = \frac{M_i}{\sum_j^{A,B,C,D} \frac{M_j}{C_{j0}}}, i = A, B, C, D \tag{A6}$$

where C_{j0} for $j = A, B, C, D$ denotes the molar densities of the individual pure species, and M_i is the molar flow rates of these species within the overhead stream of the flash tank separator.

$$M_i = k \frac{\alpha_i(F_7C_{i3} + F_9C_{i5}) \sum_j^{A,B,C,D} (F_7C_{j3} + F_9C_{j5})}{\sum_j^{A,B,C,D} \alpha_j(F_7C_{j3} + F_9C_{j5})}, i = A, B, C, D \tag{A7}$$

In the process description, k represents the proportion of the overhead stream recycled back to the CSTRs and remains constant. The parameters of the model are presented in Table [A.2](#).

Table A.2: *Process Parameters*

Parameters	Values	Parameters	Values
$H_{\text{vap},A}$	$3.073 \times 10^4 \text{ J mol}^{-1}$	$H_{\text{vap},B}$	$1.35 \times 10^4 \text{ J mol}^{-1}$
$H_{\text{vap},C}$	$4.226 \times 10^4 \text{ J mol}^{-1}$	$H_{\text{vap},D}$	$4.55 \times 10^4 \text{ J mol}^{-1}$
$H_{\text{ref},A}$	$7.44 \times 10^4 \text{ J mol}^{-1}$	$H_{\text{ref},B}$	$5.91 \times 10^4 \text{ J mol}^{-1}$
$H_{\text{ref},C}$	$2.02 \times 10^4 \text{ J mol}^{-1}$	$H_{\text{ref},D}$	$-2.89 \times 10^4 \text{ J mol}^{-1}$
$C_{p,A}$	$184.6 \text{ J mol}^{-1} \text{ K}^{-1}$	$C_{p,B}$	$59.1 \text{ J mol}^{-1} \text{ K}^{-1}$
$C_{p,C}$	$247 \text{ J mol}^{-1} \text{ K}^{-1}$	$C_{p,D}$	$301.3 \text{ J mol}^{-1} \text{ K}^{-1}$
C_{A0}	$1.126 \times 10^4 \text{ mol m}^{-3}$	C_{B0}	$2.028 \times 10^4 \text{ mol m}^{-3}$
C_{D0}	6485 mol m^{-3}	T_{ref}	450 K
T_{A0}	$473 \times 10^4 \text{ K}$	T_{B0}	473 K
T_{D0}	473 K	k	0.8
ΔH_1	$-1.53 \times 10^5 \text{ J mol}^{-1}$	ΔH_2	$-1.118 \times 10^5 \text{ J mol}^{-1}$
ΔH_3	$4.141 \times 10^5 \text{ J mol}^{-1}$	R	$8.314 \text{ J mol}^{-1} \text{ K}^{-1}$