Cloud-Based Model Predictive Control

Establishing a Fully Distributed Architecture for Nonlinear MPC

BY

Sébastien Norman Favreau Skoko

A THESIS SUBMITTED TO THE

Department of Chemical Engineering

IN CONFORMITY WITH THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

ON THE DATE OF

October 13th 2017

Queen's University

Kingston, Ontario, Canada

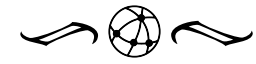
Abstract

Over the last two decades, distributed model predictive control (MPC) has become an increasingly popular research topic. Distributed MPC constitutes a valuable tool, providing computationally efficient optimal control algorithms that can successfully account for the interactions of complex, interconnected processes. Achieving closed-loop stability and centralized performance in nonlinear, distributed or decentralized systems remains an obstacle to existing distributed MPC formulations.

In this thesis a discrete-time cloud-based model predictive control architecture for application to unstable nonlinear systems under process constraints is developed using a cooperative, distributed optimization approach. Distributed time-varying extremum-seeking, a form of model-free optimal control, is integrated to a conventional MPC scheme to solve the optimization problem at each time step. Each of the control horizon's inputs is assigned to an agent that shares its local cost information over a network. Agents use this data to estimate the average total cost of the system, which they minimize by a gradient-descent control law based on local approximations of the average total cost gradient.

To demonstrate the effectiveness of the discrete-time cloud-based MPC architecture and its stability in closed-loop, three case studies are performed. The first study considers a nonlinear MIMO system subjected to input constraints. The numerical results indicate that the proposed controller recovers the performance of its centralized

counterpart. The second case study considers a non-isothermal exothermic CSTR. The simulation demonstrates that the cloud-based architecture can effectively stabilize systems with complex dynamics of higher order. The third case study investigates the architecture's applicability to economic MPC and shows that the developed controller is capable of stabilizing a system to a periodic orbit when suitable constraints are applied.



Acknowledgments

I would like to offer my most gracious thanks to my supervisor, Dr. Martin Guay, for his indispensable guidance and patience over the duration of this endeavor. As an instructor, he introduced me to the delightful field of control theory and, as a mentor, he pushed me to consider the more complex problems that truly ignited my passion for the topic.

I would like to thank Dr. Nicolas Hudon as well for his shining optimism and resounding charisma. His positive encouragement helped fuel many late nights and work-filled weekends. In addition, many thanks go to my colleagues Isaac, Judith, Dan, Hussain, Mohammad, little Mia, big Mia, and Sayed in the G37 office for their delightful company and productive discussions. I would also like to thank my good friends Jamie, Cameron, Praf, Mike, Kyle, and Thanushan for their invaluable support and for patiently hearing out my many frustrations as I progressed through my graduate studies and research.

Finally, I would like to express my deepest gratitude to my parents, Joylyn and Mark, and to my brothers, Zacharie and Jonathon, for their boundless love and unconditional faith. It was thanks to their steadfast encouragement that I was capable of embarking on this journey and to their unwavering support that I was able to see it through.

Contents

List of Figures

List of Tables

List of Abbreviations

ATC	Average Total Cost
CBMPC	Cloud-Based Model Predictive Control (Controller)
DAC	Dynamic-Average Consensus
DARE	Discrete-time Algebraic Riccati Equation
DESC	Distributed Extremum-Seeking Control (Controller)
DMPC	Distributed Model Predictive Control
EBESC	Estimation-Based Extremum-Seeking Control
ESC	Extremum-Seeking Control (Controller)
LQR	Linear Quadratic Regulator
LTI	Linear Time-Invariant
MAS	Multi-Agent System
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
PBESC	Perturbation-Based Extremum-Seeking Control
PE	Persistence of Excitation
RLS	Recursive Least-Squares
ROA	Region of Attraction
RTO	Real-Time Optimization
TESC	Terminal Extremum-Seeking Controller
TVESC	Time-Varying Extremum-Seeking Control

List of Symbols

$\mathbf{0}$	Zero vector
$\mathbf{1}$	Vector of ones
$\mathbf{A}$	State-coefficient matrix in LTI model/Adjacency matrix
b_i^+/b_i^-	Upper/lower bound on the constrained variable z_i
$\mathbf{B}$	Input-coefficient matrix in LTI model
c	Bias in CBMPC local stage cost
$\boldsymbol{d}$	Dither signal vector
D	Dither signal amplitude
$\mathbf{D}$	Degree matrix
e	Parameter estimation error
$\boldsymbol{f}(\cdot,\cdot)$	Mapping describing system dynamics
$\boldsymbol{g}_i(\cdot,\cdot)$	Mapping of agent i's predicted state
$h(\cdot)$	State-valued local cost function
$h_{\mathrm{T}}(\cdot)$	Terminal cost function
$\mathbf{I}$	Identity Matrix
J	Total cost
$\widehat{J}_i$	Agent i's estimate of the average total cost
k	Sampling instant
k_g	Proportional gain in extremum-seeking control law
$\mathbf{K}$	State-feedback gain matrix
$\ell(\cdot)$	General ESC cost function

$\ell(\cdot,\cdot,\cdot)$	CBMPC local cost function
$\ell_u(\cdot)$	Local input-valued stage cost
$\ell_{\tilde{u}}(\cdot)$	Local input difference-valued stage cost
$\ell_x(\cdot)$	Local state-valued stage cost
$\mathbf{L}$	Laplacian matrix
m	Number of inputs
m_c	Number of constrained variables
n	Number of states
p	Number of outputs or agents/Length of prediction horizon
$\mathbf{P}$	Terminal cost matrix
$\mathbf{Q}$	State weight matrix
$\boldsymbol{Q}_i$	Agent-specific state weight matrix
$\mathbf{R}$	Input weight matrix
$\mathbf{S}$	Input difference weight matrix
Δt	System sampling period
Δt_{D}	Distributed optimizer's sampling period
$\boldsymbol{u}$	Input vector
$\boldsymbol{x}$	State vector
$\boldsymbol{y}$	Local cost vector
$\boldsymbol{z}$	Vector of constrained variables
α	Forgetting factor for the covariance matrix update law
α_s	Filter parameter for the covariance matrix update law
β_{I}	Integral gain for dynamic average consensus algorithm
β_{P}	Proportional gain for dynamic average consensus algorithm
γ_θ	Radius of the parameter estimation uncertainty set
δ	Integration variable
ε	Time-scale separation factor
ϵ	Small step in from a constrained variable's bound

$\boldsymbol{\theta}$	Gradient of average total cost
θ_0	Initial value for parameter estimates
θ_i^0	Drift in the average total cost gradient
$\boldsymbol{\theta}_i^1$	Gradient of average total cost with respect to the local input
$\bar{\boldsymbol{\theta}}_i$	Estimated average total cost gradient prior to projection onto uncertainty set
$\widehat{\boldsymbol{\theta}}_i$	Estimated average total cost gradient
κ	Iteration of an extremum-Seeking control routine
λ_i	i^{th} eigenvalue of a matrix
$\boldsymbol{\mu}$	Barrier function weight vector
$\boldsymbol{\nu}$	Vector of i.i.d. numbers
$\xi_i(\cdot)$	Variable-specific logarithmic barrier function
$\boldsymbol{\pi}(\cdot)$	Steady-state map
$\boldsymbol{\rho}$	Integrator variables for dynamic average consensus algorithm
σ_0	Initial value of the covariance matrix's diagonal elements
σ_d^2	Variance of i.i.d. numbers
$\boldsymbol{\Sigma}$	Covariance matrix in parameter estimation routine
$\boldsymbol{\Sigma}_{Ch}$	Upper Cholesky factor of the covariance matrix
$\Upsilon(\cdot)$	Convex input penalty function for CBMPC local costs
$\boldsymbol{\phi}$	Regressor vector in parameter estimation routine
$\Phi(\cdot)$	Logarithmic barrier function
$\overline{\Phi}(\cdot)$	Mean-centered logarithmic barrier function
$\Psi(\cdot)$	Switch function
ω	Agent i's dither signal frequency vector
Ω	Concatenation of every agent's dither signal frequency vector

Simulation Variables

C_a	Concentration of species a
$C_{a,0}$	Feedstock concentration of species a
$\overline{C}_{a,0}$	Time-averaged feedstock concentration of species a
c_p	Heat capacity
E_a	Activation energy
F	Reactor-side flow rate
F_i	Jacket-side flow rate
ΔH_r	Heat of reaction
k_0	Reaction rate constant Arrhenius coefficient
M	Operating period length
Q	Heat flow
ρ	Density
R	Universal gas constant
T	Reactor temperature
T_0	Feedstock temperature
T_j	Coolant temperature
$T_{j,0}$	Coolant feedstock temperature
UA	Overall heat transfer coefficient
V	Reactor volume
V_j	Cooling jacket volume

Sets/Manifolds

$\mathbb{B}$	Denotes a ball
$\boldsymbol{\Theta}$	Estimated parameter uncertainty set
$\boldsymbol{\Lambda}$	Spectrum of a matrix
$\mathbb{N}_0$	Set of natural numbers including 0 ($\{0, 1, 2, \ldots\}$)
$\mathbb{Q}$	Set of rational numbers

$\mathbb{R}$	Set of real numbers
$\mathcal{S}$	Steady-State manifold
$\mathbf{U}$	Collection of inputs predicted by MPC controller
$\mathbb{U}$	Connected and compact set of feasible inputs
$\widetilde{\mathbb{U}}$	Connected and compact set of feasible input differences
$\mathbf{X}$	Collection of states predicted by MPC controller
$\mathbb{X}$	Connected and compact set of feasible states
$\mathbb{X}_p$	Terminal constraint set (region)
$\mathbb{Z}$	Set of integers

Style Guide

Fonts and Cases

a	Lowercase denotes a scalar
$\boldsymbol{a}$	Bold, lowercase denotes a vector
$\mathbf{A}$	Bold, uppercase denotes a matrix
$\mathcal{A}$	Curly, uppercase denotes a set or manifold
$\mathbb{A}$	Blackboard font, uppercase denotes a set

Accents

$\dot{a}$	Designates a time derivative
$\hat{a}$	Designates an estimate
$\tilde{a}$	Designates a difference between two variables
$\bar{a}$	Designates an average or normalized value
a^*	Designates an optimum value
a'	Designates a first derivative
a''	Designates a second derivative
$\boldsymbol{a}^\top$	Designates a vector's transpose

Operations/Operators

Δ	Forward difference between time steps
∇	Gradient of a function
$\oplus/\ominus$	Minkowski sum/difference
$\triangleq$	Definition
$\lvert\cdot\rvert$	Absolute value (Euclidean 1-norm)
$\lVert\cdot\rVert$	Euclidean 2-norm
$\lVert\cdot\rVert_{\mathbf{P}}^2$	Weighted inner product of a vector with itself where $\mathbf{P} = \mathbf{P}^\top$ is a weighting matrix
argmax	Maximizing argument
argmin	Minimizing argument
diag	Diagonal matrix
lim	Limit
Chol	Cholesky factor of a matrix (upper)
max	Maximum value
min	Minimum value
Proj	Projection operator
Rank	Rank of a matrix
spec	Spectrum of a matrix

{ Chapter 1 }

Introduction

The ever-increasing demands on performance, efficiency and sustainable operation in the process industries, have significantly been increasing the appeal of optimal control strategies since the turn of the century. In the chemical and process industries, these demands have manifested a pressing need to run systems as safely and profitably as possible while taking a diverse range of environmental, fiscal and operational constraints into consideration. The field of optimal control has been studied heavily over the last century, giving rise to several promising techniques that offer appealing, cost-effective solutions to some of these issues.

By providing the means to operate in an optimal fashion, properly construed control systems allow the performance of existing facilities to be improved without making modifications to the process or having to replace current unit operations with costly new equipment. However, there are still some obstacles to be overcome when applying the tools of optimal control to large-scale problems. For instance, as problems grow in dimensionality and nonlinearity, the time required to find optimal solutions can increase significantly – which can limit the application of optimal control frameworks in systems with fast dynamics. In addition, the failure of a centralized controller could have an enormous impact on the health and safety of operators or civilians near the facility and cause irreparable damage to equipment and infrastructure. Furthermore,

depending on the nature of the controller, it may not always be possible to account for time-varying optimality criteria effectively.

1.1 Motivation

When faced with a task of overwhelming breadth and difficulty, few would argue that breaking it up into several more tractable ventures and assigning these to the members of a team working collaboratively is a more effective strategy than assigning everything to one individual. Hence, there has been a steadily growing interest in the field of distributed control and optimization strategies in industry [24, 153]. By treating a global, large-scale problem as a collection of interacting subsystems, one can assign individual specialized decision makers to monitor and regulate them without having to consider the full magnitude of the problem [26]. Distributed controllers offer a number of advantages over their centralized counterparts not limited to greater computational efficiency, a lesser impact of local failures, and the ability to scale-up the controller without needing to modify the existing framework [155, 153].

Model predictive control (MPC) is an optimal control framework that has become very well-established in industry since its initial development in the late 1970s. It has reached a stark degree of popularity due to its ability to handle multi-input multi-output (MIMO) problems subjected to constraints with ease [16]. Evidently, combining these strengths with the benefits of distributed control would produce a methodology of considerable interest for many applications from chemical processing and resource allocation to flight navigation and multi-vehicle formation control – among many more [24, 153, 115, 37]. Hence, it comes as no surprise that the development of stable and

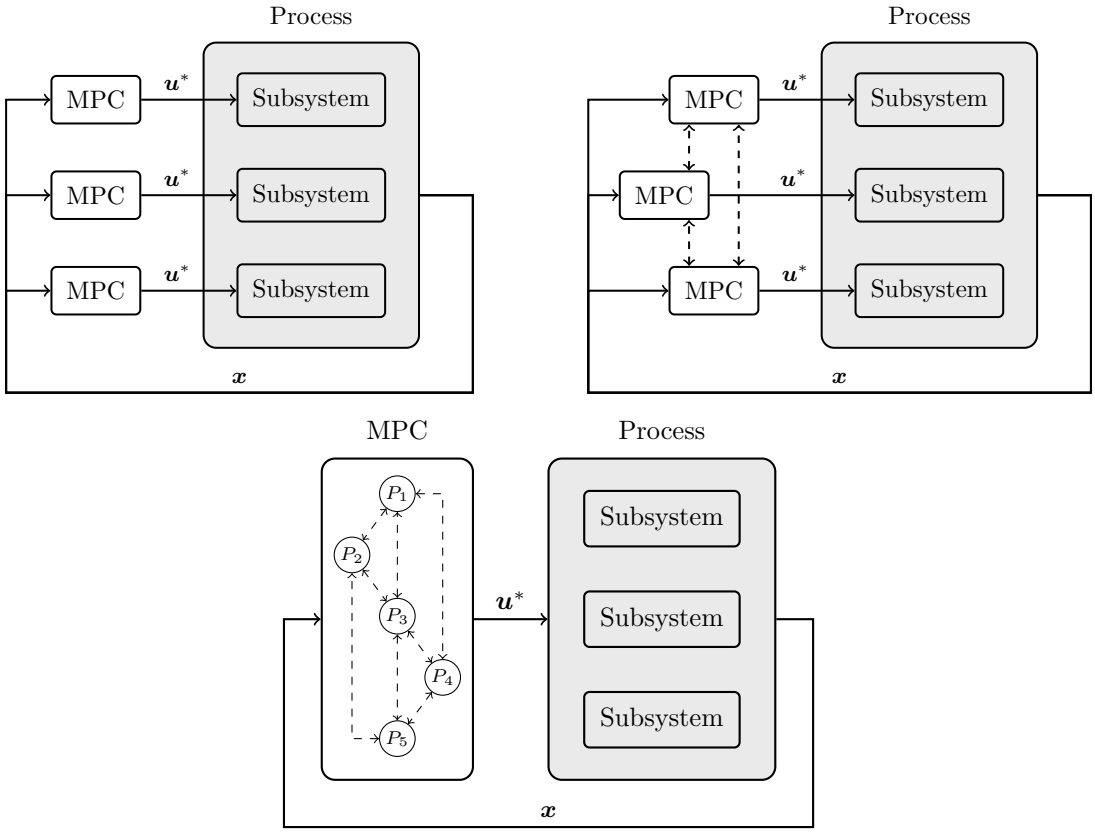


Figure 1.1: In a non-cooperative DMPC scheme (top left) local controllers do not communicate. For a cooperative DMPC setup (top right), local controllers share information to consider their impact on one another. In distributed optimization MPC (bottom), the optimization process is decomposed into smaller problems that are assigned to separate processing units.

effective distributed MPC (DMPC) architectures has become an increasingly popular topic in the literature over the last decade.

The main approaches taken to subdividing the MPC problem fall into three main categories, depicted in Figure 1.1: non-cooperative DMPC, cooperative DMPC, and distributed optimization MPC. In non-cooperative architectures, controllers are only concerned with a local subsystem and do not consider their impact on others, guiding the global system to a Nash equilibrium [153, 45]. In cooperative MPC, controllers communicate with one another to coordinate their efforts and drive their respective

subsystems to conditions that achieve a system-wide optimum [141]. On the other hand, the distributed optimization approach involves decomposing the optimization problem solved by a central MPC so that it can be solved by the agents of a distributed computation system [24].

To date, there are still a number of issues to be resolved in the field of DMPC. While non-cooperative approaches have been successful for problems such as collision avoidance in multi-vehicle formation control, a Nash equilibrium does not always constitute the most optimal solution when dealing with systems such as refineries and chemical plants. Cooperative MPC has also seen many difficulties in regards to proving the algorithms' stability and demonstrating convergence to the centralized control solution. As for distributed optimization-based MPC, a large number of iterations is often required for the controller to find a solution at each time step. In addition, suitable decompositions for the optimization problem can be excessively difficult to derive in many cases. Finally, extending stability and convergence results to the nonlinear case has proven to be a monumental task for all three approaches [24, 153].

In this thesis, a fully distributed architecture for MPC is developed for application to nonlinear systems subject to process constraints. The resulting so-called cloud-based MPC (CBMPC) framework is based on the aforementioned distributed optimization approach, although the framework can easily be adapted for the cooperative control of plant comprised of many different subsystems. By integrating a distributed extremum-seeking controller (ESC) to the MPC framework as its optimizer, the CBMPC architecture can converge to the same solutions produced by its centralized equivalent and emulate the behavior and performance of conventional MPCs. In addition, CBMPC is extended to the problem of economic MPC, in which the cost function used by the

controller as a metric for optimality, is based on process economics – thereby allowing a plant to be operated such that operational costs are minimized and revenue is maximized.

1.2 Thesis Organization

The remainder of this work is divided into 4 chapters.

Chapter 2 presents a review of the relevant literature pertaining to the topics of consensus in multi-agent systems, model predictive control, and extremum-seeking control. In the first section, the mathematical theory related to multi-agent systems is summarized, followed by summary of the various consensus-based strategies and algorithms that can be used to coordinate the efforts of agents working collaboratively to control a networked system. In the second section, a brief history of MPC is provided in addition to the many theoretical developments upon which this thesis's work is based. The stability of nonlinear MPC is covered along with the theory relating to both distributed and economic MPC. Finally, a discussion of the history and development of ESC is included to provide an overview of the mathematical theory that forms a basis for CBMPC.

Chapter 3 outlines the proposed CBMPC architecture and develops the mathematical foundations for the controller. The chapter begins by laying out the assumptions pertaining to the construction of a suitable cost function and shows how its dynamics may be parameterized to generate an exact input-output model that provides a basis for the extremum-seeking-based optimization. Following this, the components and general structure of the CBMPC algorithm are described in detail. The chapter ends

with a discussion of logarithmic barrier functions and of how process constraints may be considered by the architecture through the use of interior-point methods.

In Chapter 4, three case studies are performed to the end of testing CBMPC and validating the framework's performance. In the first case study, CBMPC is implemented to control a MIMO system with first-order nonlinear dynamics. The results are compared to the state and input trajectories generated by the corresponding centralized MPC to validate the framework's performance. In the second study, CBMPC is used to control a single-input single-output (SISO) system with third-order nonlinear dynamics at the desired set-points. In the third and final case study, CBMPC is extended to the case of economic MPC and the architecture's ability to stabilize the system to both an unknown, economically optimal steady-state and to a stable, economically optimal periodic orbit.

Chapter 5 provides the conclusion of the thesis, which includes a summary of its main contributions. Recommendations for future work to improve the CBMPC architecture's performance and range of implementation are also briefly discussed.

{ Chapter 2 }

Literature Review

This chapter provides an overview of the foundational material for this thesis's contributions. Section 2.1 provides a comprehensive review of the mathematical theory pertaining to consensus algorithms for multi-agent systems applicable to distributed control and optimization. Section 2.2 addresses the topic of Model Predictive Control (MPC) and provides an overview of the standard MPC problem addressed in the chapters that follow, various methods for establishing closed-loop stability in nonlinear systems, the fundamentals of economic MPC, and a review of several distributed MPC (DMPC) architectures that have been developed to date. Section 2.3 is dedicated to the topic of Extremum-Seeking Control (ESC) and provides a summary of its early developments, its core structure, and its extension to distributed settings.

2.1 Consensus in Multi-Agent Systems

When taking a distributed approach, one must begin by subdividing the *global* system into distinct, *local* subprocesses. The result, known as a *multi-agent system* (MAS), uses *agents* to represent individual decision makers (controllers) that are only responsible for addressing one subsystem or subprocess in particular [93]. The agents of a MAS are typically defined as being autonomous (or at least partially so), as not having

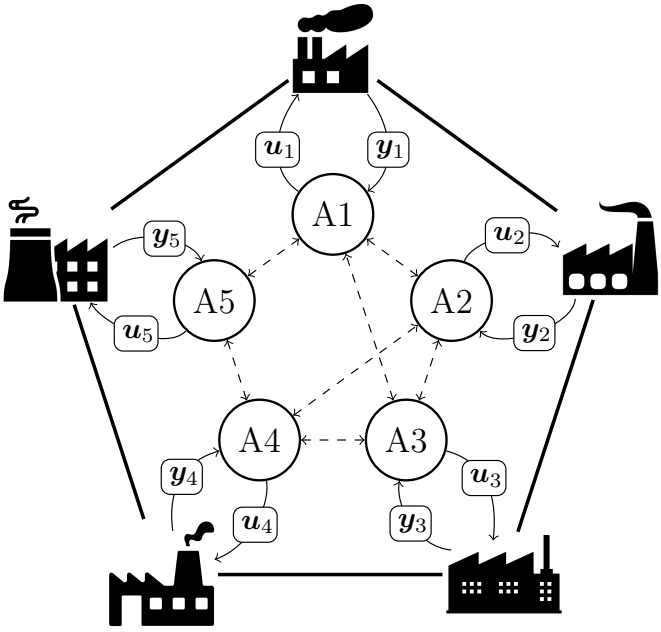


Figure 2.1: An example of a distributed control setup depicted as a MAS. The global system is made up of the five plants on the pentagon's vertices. Each agent (labeled A1 to A5) measures their respective local outputs $\boldsymbol{y}_i$ and send local inputs $\boldsymbol{u}_i$ to their assigned subsystems. The arrows with dotted lines show which agents are sending information to one another (icon sources, clockwise from top of pentagon: [Im5, Im2, Im3, Im4, Im1]).

a complete knowledge of the global system (information constraints), and as having no defined leader or supervisory capabilities [155].

Figure 2.1 shows an example of a distributed control scheme represented by a MAS. The more complex global system is controlled by several independent controllers (A1 to A5) that each measure local outputs, send local inputs to each subsystem, and share any required information with one another. This simplifies the processes observed by each agent, reducing the complexity associated with selecting suitable inputs. Since agents only consider their local variables and the information provided to them via a communication network, they do not require an explicit understanding of other agents' behavior to operate successfully.

There are two main approaches that can be used when defining how agents make decisions and select inputs, namely *cooperative* and *non-cooperative* strategies [38]. In a non-cooperative approach (also known as a competitive strategy), agents are designed to act selfishly and only consider their local problem when choosing inputs and making decisions. Thus, they make selections that benefit them individually and do not account for any negative impacts on other agents [11]. In a cooperative strategy, on the other hand, agents are designed to act more altruistically and consider their impact on the global system when selecting their inputs. This way, the decisions made are meant to bring the largest benefit to the system as a whole [38, 32]. While non-cooperative strategies have been shown to be highly effective in domains such as economics [38], the main objective for the global control of a MAS that represents an industrial plant, factory, supply train, etc., would be concerned with reaching a solution that achieves system-wide optimal performance. Hence, cooperative strategies are more suitable for the distributed control laws developed in this thesis [144, 129].

2.1.1 Representing Networks with Graphs

Given the aforementioned limitation on local knowledge (sometimes termed a *privacy constraint*), agents operating cooperatively may lack some information necessary to achieve a system-wide optimum. It follows that a suitable communication network must be created so that agents may communicate missing information to one another and work effectively. Graph theory provides a framework to describe networks numerically [14]. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is made up of a set of vertices $\mathcal{V}$ whose elements v_i represent agents and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ whose elements $e_{i,j}$ are be used to designate the flow of information from agent i to agent j, with $i, j \in \{1, \ldots, N\}$ and N being

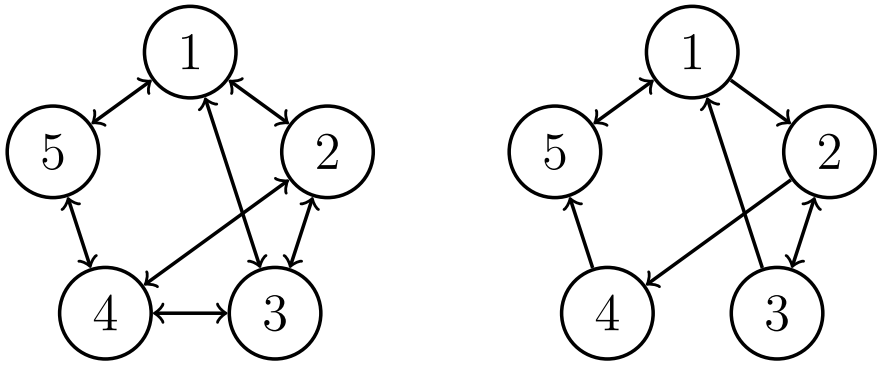


Figure 2.2: The arrows in the graphs above denote the directions in which information is flowing in the network. On the left, all communications are two-way and so the graph is said to be *undirected*. The graph on the right, however, depicts a *directed* communication network in which agents do not always receive information from the agents they send information to, and vice-versa.

the total number of agents (vertices). When all communication pathways are two-way, i.e. $e_{i,j} \in \mathcal{E} \Leftrightarrow e_{j,i} \in \mathcal{E}$, the graph is said to be *undirected*. Alternatively, the graph is said to be *directed* if the communication pathways are only one-way and agents do not necessarily receive information from all the agents they send information to and vice-versa, i.e. $e_{i,j} \in \mathcal{E} \not\Leftrightarrow e_{j,i} \in \mathcal{E}$ (see Figure 2.2). In some cases, weights $w_{i,j}$ are assigned to each edge $e_{i,j} \in \mathcal{E}$ to produce a *weighted* graph. When the weights are taken to be 1 for each edge, the graph is termed *unweighted* [14].

A graph representing a network can then be described by a number of matrices. Of interest are:

i) the square *Adjacency Matrix*, $\mathbf{A} \in \mathcal{M}_{p\times p}(\mathbb{R}_{\geqslant 0})$, for which each element is set as $a_{ij} = w_{i,j}$ if $e_{i,j} \in \mathcal{E}$ and, if $e_{i,j} \notin \mathcal{E}$, then $a_{i,j} = 0$. If the graph is undirected, $\mathbf{A}$ is symmetric and $\mathbf{A} = \mathbf{A}^\top$;

ii) the diagonal *Degree Matrix*, $\mathbf{D} \in \mathcal{M}_{p\times p}(\mathbb{R}_{\geqslant 0})$, where each element $d_{ii} = \sum_{j=1}^{p} w_{i,j}$. When the graph is unweighted, the diagonal elements are simply the total number of agents to which agent i sends information to;

iii) and the *Laplacian Matrix*, $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Note that an undirected graph will have $\text{Rank}(\mathbf{L}) = p - 1$ while a *directed* graph (for which $\mathbf{A} \neq \mathbf{A}^\top$) will have $\text{Rank}(\mathbf{L}) < p - 1$. By the definitions of $\mathbf{A}$ and $\mathbf{D}$, it follows that $\mathbf{L}(\mathbf{1}) = \mathbf{0}$ and so a Laplacian matrix is said to be *weight-balanced* when $\mathbf{L}^\top(\mathbf{1}) = \mathbf{0}$. For the graphs depicted in Figure 2.2, the Laplacians would be

$$\mathbf{L}_{ud} = \begin{bmatrix} 3 & -1 & -1 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix} \text{ and } \mathbf{L}_d = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ 0 & 2 & -1 & -1 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2.1.2 Consensus-Based Cooperative Strategies

One popular method for coordinating the actions of a MAS's agents in a cooperative fashion is via consensus algorithms [118]. These provide a convenient means for agents to estimate an unknown variable and drive their local estimates to a collective agreement using only local information. As a result, agents can achieve some understanding of the global system's state without having to know it explicitly. This is a very appealing attribute since it provides a machinery for breaking down large, interconnected problems in a scalable manner. Is also offers the advantages of distributed computing, such as computational efficiency and resilience to faults, communication failures, and unknown disturbances [143]. Since they were first established, consensus algorithms have been successfully applied in a large variety of fields including spacecraft formation control [35], energy management [66], crowd-sourcing [160] and even in the medical field for managing type 2 diabetes therapy [112].

When applied to the problem of distributed optimization, a consensus algorithm provides agents with an estimate of the total global cost to be minimized when they share local cost information [114]. To converge successfully, however, the communication network's topology must meet a *connectivity* requirement. A network described by an undirected graph is said to be connected if there exists a sequence of edges that can join one agent to every other agent in the network. In the directed case, for any pair of nodes (i, j) there must be a set of edges connecting agent i to agent j and another sequence linking agent j to agent i. For both cases, this condition implies that the information provided by each agent propagates through the entirety of the network and that agents are sufficiently informed when making their estimates.

Early work on consensus was limited to *static-average* algorithms which guarantee that each agent estimates the average of their initial states. The most common algorithm yields first order, linear dynamics and required that the network be connected and weight-balanced [119]. These results were extended to nonlinear dynamics, achieving convergence when the update law is based on some convex combination of states at the previous iteration and the network remained connected – although they would not necessarily converge on the exact average of the initial states [106].

As consensus methods were developed further, focus shifted to algorithms capable of tracking the dynamic average of time-varying signals. In the work of Freeman et al. [49], consensus via first- and second-order linear dynamics is explored and convergence is proven in continuous time when $\mathbf{L}$ was weight-balanced and connected, and the input signals were slowly time-varying. Algorithms working in discrete-time were later developed using n^{th} order update dynamics that converged on the average of input signals with bounded n^{th} differences [161]. In Kia et al. [81], consensus algorithms

with n^{th} order dynamics in both continuous and discrete time were proposed. Their results bounded the algorithms' convergence rates and tracking errors with respect to $\mathbf{L}$'s eigenvalues and the bound on the n^{th} difference (or derivative), providing insight on the impact of the network's topology on algorithm performance. The algorithms described by Kia et al. are of particular interest to this work as they provide a highly effective means for agents in a MAS to estimate the system's total cost based on limited local cost data they share with one another.

2.2 Model Predictive Control

MPC is a model-based optimal control framework that was first proposed by ADERSA in 1978 [131, 123] and Shell Oil in 1979 [27, 123]. This technique utilizes a dynamic model of the process to be controlled and a sequence of future inputs that make up a *control horizon* to predict the plant's output over a given *prediction horizon*. These future inputs and predicted outputs are used as the arguments of a *cost* (or *objective*) function that provides a metric for the viability of the plant's predicted trajectory. Once a sequence of inputs shown to minimize the a user-defined cost function is determined, the first input on the control horizon is implemented by the controller. This process is then repeated at the next time step [16, 129, 55].

MPC is now firmly established in industry for plant-wide control due to its many appealing characteristics [4, 48, 126]. MPC is capable of handling large MIMO problems with ease and can easily accommodate large time delays [16, 129]. By its predictive nature, MPC inherently introduces a measure of feed-forward control to its input efforts as well [16]. However, the performance of MPC is heavily dependent on the accuracy of

the models used for making predictions, which can be difficult to develop and maintain in practice [16]. In addition, highly nonlinear and interconnected dynamics introduce varying degrees of non-convexity to the dynamic optimization problem. This can lead to poor computational performance and limit the application of MPC for fast systems [105, 126, 55].

2.2.1 Nonlinear Model Predictive Control in Discrete Time

The systems considered for nonlinear MPC (NMPC) in discrete-time are those described by time-invariant models of the form

$$\boldsymbol{x}[k+1] = \boldsymbol{x}[k] + \boldsymbol{f}\left(\boldsymbol{x}[k], \boldsymbol{u}[k]\right) \tag{2.1}$$

where k is the sample time, $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^n$ is the state and $\boldsymbol{u} \in \mathbb{U} \subseteq \mathbb{R}^m$ is the input. The vector field $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is such that there exists a pair $(\boldsymbol{x}^*, \boldsymbol{u}^*) \in \{\mathbb{X} \times \mathbb{U}\}$ for which $\boldsymbol{f}\left(\boldsymbol{x}^*, \boldsymbol{u}^*\right) = \boldsymbol{0}$ and is assumed to be smooth. The feasible sets for the state and input variables are taken to be connected and convex. The traditional optimization problem solved at each time step involves the constrained minimization of a convex (typically quadratic) cost function over a finite horizon:

$$\begin{aligned} \min_{\boldsymbol{u}_i \in \mathbb{U}} J[k] = & \sum_{i=0}^{p-1} h\left(\boldsymbol{x}_i[k], \boldsymbol{u}_i[k]\right) + h_{\mathrm{T}}\left(\boldsymbol{x}_p[k]\right) \\ & \boldsymbol{x}_{i+1}[k] = \boldsymbol{x}_i + \boldsymbol{f}\left(\boldsymbol{x}_i[k], \boldsymbol{u}_i[k]\right) \ \forall\ i \in \{0, \ldots, p\} \\ \text{subject to} \quad & \boldsymbol{x}_0[k] = \boldsymbol{x}_{meas} \\ & \boldsymbol{x}_i[k] \in \mathbb{X} \ \forall\ i \in \{0, \ldots, p\} \end{aligned} \tag{2.2}$$

where $\boldsymbol{x}_{meas}$ is the measured state, $h : \mathbb{X} \times \mathbb{U} \to \mathbb{R}_{\geqslant 0}$ is the stage cost, and $h_{\mathrm{T}} : \mathbb{X} \to \mathbb{R}_{\geqslant 0}$ is the terminal stage cost used to guarantee closed-loop stability (see Section 2.2.2).

The most common stage cost employed in MPC is the quadratic given by

$$h\left(\boldsymbol{x}_i[k], \boldsymbol{u}_i[k]\right) \triangleq \|\boldsymbol{x}_i[k] - \boldsymbol{x}_{sp}\|^2_{\mathbf{Q}} + \|\boldsymbol{u}_i[k] - \boldsymbol{u}_{sp}\|^2_{\mathbf{R}} \tag{2.3}$$

where $\boldsymbol{x}_{sp} \in \mathbb{X}$ is the state's set point, $\boldsymbol{u}_{sp} = \boldsymbol{\pi}\left(\boldsymbol{x}_{sp}\right)$ is the steady-state input corresponding to $\boldsymbol{x}_{sp}$ given by the map $\boldsymbol{\pi} : \mathbb{X} \to \mathbb{U}$, and $\mathbf{Q}$, $\mathbf{R}$ are symmetric positive semi-definite and positive-definite weighting matrices, respectively.

The formulation of the input penalty above poses a problem, however. If the system is subjected to disturbances, the steady-state input corresponding to the desired set-point would be altered. In NMPC, incremental changes in the input can be used to introduce integral action to the control effort, allowing $\boldsymbol{u}_{sp}$ to be identified on-line [129, 55]. This approach involves penalizing the *change* in the input, producing a stage cost of the form

$$h\left(\boldsymbol{x}_i[k], \boldsymbol{u}_i[k]\right) \triangleq \|\boldsymbol{x}_i[k] - \boldsymbol{x}_{sp}\|^2_{\mathbf{Q}} + \|\boldsymbol{u}_i[k] - \boldsymbol{u}_i[k-1]\|^2_{\mathbf{S}} \tag{2.4}$$

where $\mathbf{S} \in \mathcal{M}_{m\times m}$ is a symmetric positive definite weighting matrix. Note that this scenario does not require $\boldsymbol{u}_{sp}$ to be known explicitly.

2.2.2 Closed-Loop Stability for Nonlinear MPC

Since solving infinite-horizon optimization problems online is impractical in practice, MPC typically considers the finite-horizon case with a formulation that seeks an approximation of the infinite horizon problem. Early forms of MPC used open-loop predictions as the arguments of the cost function and, due to the predictions over a finite-horizon, stability was not guaranteed [98]. Historically, the most common approaches to establish stability criteria for MPC involved making slight modifications

to the problem (2.2). The earliest of these methods involved considering the cost function as a Lyapunov function for the system and placing an equality constraint on the terminal state. Chen and Shaw [18] were the first to use this method, demonstrating stability for the continuous-time, unconstrained case. Constrained, time-varying nonlinear discrete-time systems were addressed by Keerthi and Gilbert [79], using the cost as a Lyapunov function. This approach became the standard method to guarantee stability in MPC [98].

Another approach, proposed by Bitmead et al. [12] for unconstrained linear systems, allowed the terminal equality constraint to be abandoned in favor of a quadratic terminal cost function $h_T : \mathbb{R}^n \to \mathbb{R}_{\geqslant 0}$ defined as

$$h_T(\boldsymbol{x}_p) = \frac{1}{2}\boldsymbol{x}_p^\top \mathbf{P} \boldsymbol{x}_p$$

where $\mathbf{P}$ is a symmetric positive-definite matrix generated by solving an algebraic Riccati equation. Built on the work of Kalman [72] regarding the optimal control of linear systems with quadratic cost functions, Bitmead et al. used the solution $\mathbf{P}$ to determine a stabilizing state-feedback control law given by

$$u(\boldsymbol{x}) = -\mathbf{K}\boldsymbol{x}$$

where $\mathbf{K} \in \mathcal{M}_{m\times n}(\mathbb{R})$ is a static, optimal gain matrix determined using $\mathbf{P}$ (see [72]).

Yet another method, developed for application to constrained, nonlinear continuous-time systems by Michalska and Mayne [103], involves constraining the last state on the prediction horizon to a *terminal region*, $\mathbb{X}_p \subset \mathbb{X}$, rather than using a terminal cost. In this approach, the terminal region is defined as a neighborhood of the origin in which a linear approximation of the nonlinear dynamics can make further predictions

with sufficient accuracy. Hence, this region is positively invariant when the input is produced by the feedback law utilizing the optimal gain matrix $\mathbf{K}$. Hence, stability is guaranteed if the system can be steered to $\mathbb{X}_p$ in finite-time, which Michalska and Mayne accomplish by using prediction horizons of variable length. This form of "dual-mode" MPC was later extended to the constrained nonlinear systems in discrete-time by Chisci et al. [22].

Finally, the works of De Nicolao et al. [29] and Chen and Allgöwer [19] used a combination of the terminal cost function and the terminal region to establish the stability of constrained nonlinear systems in continuous-time, and also in discrete time in the work of Chen and Allgöwer. In these approaches, the terminal cost is employed such that the total cost represents an approximation of the infinite-horizon control problem when the terminal region constraint, $\boldsymbol{x}_p \in \mathbb{X}_p$, is met. However, there are many arguments made in the literature that the terminal region constraint is met when the prediction horizon is of a sufficient length, implying that the constraint does need to be actively enforced [122, 69, 98, 55].

2.2.3 Economic Model Predictive Control

Since MPCs determines inputs by solving an optimization problem, it is inherently well suited for taking economics under consideration. If the objective function is chosen to represent some aspect of the process's economics, then inputs could be chosen on the basis of maximizing profit, minimizing operating costs, or such that certain economic constraints are met [43, 128].

Numerous investigations examining both experimental data and theoretical simulations have demonstrated that operating a plant periodically is a much more effective

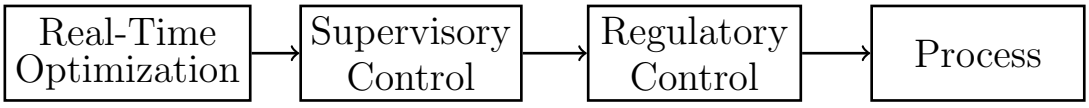


Figure 2.3: A depiction of the multi-level hierarchy used in early attempts to integrate economic optimization with MPC installations.

strategy for optimal economic performance than steady-state operation – see for instance [34, 87, 10, 137, 140, 121, 15, 111, 88, 96, 139] (of these, [137, 121, 15, 111, 88] also propose control methods for the desired dynamic plant operation). Such transient operations pose a new set challenges for MPC; classical notions of stability are all based on steady-state plant operation and can not be applied to prove a system's closed-loop stability under economic MPC in time-varying cases.

Consequently, the earliest attempts to integrate economic optimization with effective process control involved the implementation of multi-level hierarchal frameworks [43]. In these architectures, such as the one shown in Figure 2.3 (taken from Marlin and Hrymak [97]), the different levels operate on different time scales to account for different aspects of the plant's operation. At the highest level, real-time optimization (RTO) samples the system operating on the scale of several hours (days) and computes optimal set-points for the plant based on up-to-date information affecting the process's profitability or operating costs [43, 97]. These set-points are fed to the supervisory control layer – commonly consisting of a traditional MPC – which determines the optimal inputs to reach these set points, operating on a time-scale of minutes to hours. These inputs (e.g. a flow rate) are then passed to the regulatory control layer which operates on a scale of seconds to implement these inputs in the process (e.g. a PID controller that regulates the flow rate using a control valve). However, as the demands on the process industry continued to increase, so did the need for architectures that could

integrate economic optimization to the control effort more effectively, further improve the plant's performance, and react to changing conditions [128, 71, 138, 28].

It was not until recent years that studies establishing sound stability criteria for a variety of economic MPC formulations began to appear in the literature [43]. One method involved using an infinite horizon in the optimization problem. For instance, Huang et al. [68] established robust stability for a cyclic steady-state by proving input-to-state stability with respect to a bounded disturbance. To do so, the infinite-horizon problem was approximated by using a large, finite-length horizon. In the work of Angeli et al., a finite horizon was used in conjunction with an auxiliary control law that approximated the contribution of the remaining infinitely long tail to the total cost [7]. Similarly, the developments of Mendoza-Serrano and Chmielewski [101] and Omell and Chmielewski [120] applied and verified the concept of breaking the infinite series of predictions into a finite horizon and an infinitely long tail for application to HVAC systems and Integrated Gasification Combined Cycle power generation plants.

Other means of establishing stability in economic MPC architectures involved the use of terminal constraints. In one approach, Ferramosca et al. [46] adopt a point-wise equality constraint for the terminal state of the form $\boldsymbol{x}_p[k] = \boldsymbol{x}_{p+1}[k]$, allowing the predicted open-loop trajectory to converge to a steady-state manifold. This allows the controller to accommodate time-varying factors affecting economic optimality [46]. In the work of Angeli et al. [7], a time-average constraint on the input is included to place an upper bound on the total control effort that can be applied over a defined operating period. Combining this with a terminal equality constraint that sets the terminal state equal with economically optimal steady-state corresponding to the desired average input, they analyzed the formulation's asymptotic average performance compared to

the optimal steady-state operation [7, 43]. Similarly, Müller et al. constrained the terminal state to the economically optimal steady-state and incorporated a self-tuning terminal cost [109]. It was also shown that including a time-varying constraint on the output ensures the fulfillment of time-average constraints placed on either states or inputs by Müller et al. [110]. They also establish a dissipativity condition indicative that a system will asymptotically converge to the economically optimal steady-state and additional average constraints that can be used to show convergence when the dissipativity criterion is not met.

A number of other publications are concerned with extending Lyapunov-based MPC techniques to economic MPC to define sufficient criteria for closed-loop stability. The works of Heidarinejad et al. [67] and Ellis et al. [44] capitalize on high-gain observers to create a state-estimation-based Lyapunov technique for economic MPC, accounting for bounded process and measurement noise via receding-horizon estimations that quickly converge on the actual system states [44]. In [3], Alanqar et al. define a Lyapunov-based economic MPC formulation by using a multitude of empirical linear time-invariant models to expand the state-space over which predictions of a nonlinear system's behavior can be used effectively. Lyapunov-based methods are also used by Chen et al., who outline a distributed architecture for economic MPC and establish its stability [20].

The formulation developed by Grüne abandons the Lyapunov- and terminal constraint-based approaches that have so far been adopted by the majority to establish the stability of economic MPC. These results hold a considerable advantage in that the terminal region can be expanded to include the entirety of the feasible state-space and that suitable control Lyapunov functions do not need to be established (an often arduous

endeavor) [54]. Presently, the central disadvantages of such methods stem from the requirement of a suitably long prediction horizon (which increases the problem's complexity) and the requirement that a number of other controllability requirements be satisfied [54, 43].

The aforementioned publications constitute a broad sampling of the literature dedicated to outlining stability criteria for economic MPC. For a more detailed survey, the reader is referred to the work of Ellis et al. [43] and the references therein.

2.2.4 Distributed Model Predictive Control

As explained in Section 2.1 there are two main approaches to distributed control strategies, namely non-cooperative and cooperative strategies. A number of non-cooperative distributed MPCs (DMPC) have been developed since the early 2000s. For instance, a framework for handling linear state-coupled subsystems is proposed in the work of Jia and Krogh [70]. An algorithm for handling linear systems in discrete-time is outlined in by Camponogara et al. [17] and Keviczky et al. develop a DMPC for application to dynamically decoupled linear subsystems linked through their cost functions and constraints in [80]. Dunbar and Murray describes a scheme for dynamic- and constraint-decoupled systems that couple the global system's state vector via the cost function in the context of multi-vehicle formation control in [37], with the results extended to dynamically-coupled nonlinear subsystems in the work of Dunbar [36]. The architecture presented by Farina and Scattolini integrates the "tube-based" approach to robust MPC published by Mayne et al. [99] to their DMPC algorithm for linear state-coupled subsystems subjected to bounded disturbances in discrete-time [45]. Note that in general, such non-cooperative approaches can only achieve a Nash equilibrium if the local

control efforts remain selfish [24]. This method is particularly well suited for applications in areas such as multi-vehicle trajectory planning and collision avoidance [24, 37] and competitive markets [38].

For large-scale industrial systems, however, cooperative approaches would be better equipped for achieving maximum performance as a Nash equilibrium may not always represent the best possible solution. In a cooperative setting, the agents of a DMPC algorithm consider their impact on other subsystems through inter-agent communication and work towards minimizing a global cost rather than selecting inputs on the basis of minimizing their respective local costs. Early cooperative DMPC models, the first of which was proposed by Venkat et al. [152], were iterative in nature. This method, refined further by Rawlings and Stewart [130], dealt with linear subsystems subjected to bounded disturbances coupled only through the input and required that each agent has knowledge of the global system dynamics and total cost. Local controllers determined optimal input trajectories by fixing the inputs manipulated by other agents to the last known value, shared their predicted trajectories with one another, and determined a new optimal trajectory based on their weighted sum [130]. If a termination condition was met the inputs were sent to their local subsystems and, if not, the process is repeated. Another iterative DMPC is outlined by Stewart et al. for linear systems in which the closed-loop performance approaches that of the centralized controller's [141]. These results were extended to nonlinear systems by Stewart et al. [142]. However, the results of the analysis could not demonstrate that the iterative DMPC could recover the performance of the centralized control system [142].

A cooperative DMPC is developed in the work of Liu et al. [91] using Lyapunov techniques for constrained and coupled nonlinear systems, but as with [142], convergence

to the performance of the corresponding centralized controller could not be guaranteed due to the non-convexity of the nonlinear problem. Later frameworks for linear input-couple subsystems based on agent negotiation, suggested by Maestre et al. [94, 95], did not necessitate that local controllers posses full knowledge of the global system's dynamics and used communications between the agents to settle on a solution.

Another approach that was adopted for defining DMPC architectures involved distributing the optimization problem rather than designing individual MPCs for each local subsystem. This involved a decomposition of the large-scale problem to formulate a number of smaller ones. For example, Negenborn outlines a number of of augmented Lagrangian formulations for the optimization problem [115] and Rantzer proposes a dual decomposition method to relax the couplings between subsystems using Lagrange multipliers and verifies the construct's practical stability in [127]. Alternatively, the methods described in Scheu et al. [134] and Scheu and Marquardt [135] rely on a gradient-based optimization built around the exchange of sensitivities. A primal-dual active-set distributed optimization method for DMPC applied to linear systems with quadratic costs subjected to bounded disturbances was developed by Koehler et al. [82]. Another approach, based on the method of alternating direction of multipliers is presented in Wang and Chong [154]. While using a distributed optimization approach may facilitate the task of establishing the resulting DMPC's stability, it has been observed that more iterations may be required at each time step to achieve convergence. This prompted a number of efforts to develop faster computation algorithms, such as those described by Necoara et al. [113] and Ławryńczuk [85].

More recently, the work of Gao et al. proposes a cooperative, iterative DMPC for nonlinear systems with second-order dynamics and guarantees the controller's stability

by means of properly constructed terminal costs, terminal regions, auxiliary controllers (that approximate the infinite horizon problem) and compatibility constraints that govern consensus between the agents' proposed solutions. Dutta et al. presents an adaptive DMPC for application to strongly coupled nonlinear systems with potentially time-varying dynamics in [39]. For more details on the multitude of DMPC frameworks that have been developed to date, the reader is referred to Venkat et al. [153] and Christofides et al. [24].

2.3 Extremum-Seeking Control

Model-based control methods display numerous strengths and a large degree of research has been purposed with improving the robustness of these methods with respect to noisy signals, model inaccuracies, and unknown disturbances [147, 83, 133]. However, the performance of model-based control methods remains dependent on the availability of a model and, to a somewhat lesser extent, its accuracy [16, 147]. This intrinsic performance requirement can, at times, manifest itself as a monumental obstacle to overcome as there exists a large number of systems and applications that are either too complex to model or that remain poorly understood. While there are tools available to help cope with such situations – for instance, empirical models built on experimental data can be developed for highly complex or poorly understood systems – their use may not always be feasible for reasons not limited to economic constraints, computational difficulties, and hardware limitations [123, 147, 83].

The need for controllers in applications where the use of model-based control methods proved infeasible sparked a number of efforts to develop and further refine a variety

of *model-free* control systems. These efforts gave rise to many powerful frameworks including neural networks, intelligent PID controllers, adaptive and machine learning-based control methods, extremum-seeking control (ESC) and more. Of primary interest to the work developed in the chapters that follow is ESC, a form of optimal control that uses output feedback to guide the system to a steady-state that minimizes (maximizes) a given cost (performance) function – without requiring any knowledge of the system's dynamics.

The first developments of ESC theory were based on the work of Leblanc [86], which addressed issues related to powering electric train cars. The mechanism he developed inspired the work of Kazakevich, who saw how the concept's governing principles could be applied to control theory [74, 76, 75, 78, 77]. While there was some scattered work further developing these results in the decades that followed [107, 13, 100], a more sustained interest in ESC did not appear until the turn of the century when Krstić and Wang [84] published a formal definition of the ESC problem along with a complete proof of the controller's stability. For a more detailed history of ESC, the reader is referred to Tan et al. [145].

2.3.1 The General Extremum-Seeking Problem

Consider the general time-invariant, continuous time, nonlinear system given by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.5}$$

$$y = h(\boldsymbol{x}) \tag{2.6}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, and $y \in \mathbb{R}$ are the state, input, and output, respectively. The vector field $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is taken to be smooth while the scalar cost

function $h : \mathbb{R}^n \to \mathbb{R}$ is chosen to be smooth and convex. The primary task of an ESC unit is to guide the system to a steady-state pair $(\boldsymbol{x}^*, \boldsymbol{u}^*)$, i.e. a pair for which $\boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{u}^*) = \boldsymbol{0}$, where $\operatorname{argmin}\{y(\boldsymbol{x})\} = \boldsymbol{x}^*$. This problem can be re-casted by defining a map $\boldsymbol{\pi}(\boldsymbol{u}) = \boldsymbol{x}_{s.s.}$, where $\boldsymbol{x}_{s.s.}$ is the steady-state corresponding to the input $\boldsymbol{u}$. Using this definition, the steady-state cost can be expressed as

$$y = h(\boldsymbol{x}_{s.s.}) = h(\boldsymbol{\pi}(\boldsymbol{u})) = \ell(\boldsymbol{u}) \tag{2.7}$$

where $\ell : \mathbb{R}^m \to \mathbb{R}$ is defined as $\ell \triangleq h \circ \boldsymbol{\pi}$. With a properly constructed optimization problem now posed, a control algorithm can be designed to drive the input $\boldsymbol{u}$ to one that minimizes the steady-state cost, $\boldsymbol{u}^*$.

While many variants of ESC have been developed since its first establishment, the majority are built on three main components:

i) an estimation routine that provides the controller with an approximation of the cost's gradient, $\frac{\partial \ell}{\partial \boldsymbol{u}}$;

ii) a dither signal that provides the persistency of excitation – a well established requirement for many adaptive control frameworks (see Assumption 3.10 on 51) – required for closed-loop stability;

iii) and a gradient descent (ascent) control law that drives the output to an extremum.

Perturbation-Based Extremum Seeking

The ESC proposed by Krstić and Wang [84], colloquially known as *perturbation-based ESC* (PBESC), is shown by the block diagram in Figure 2.4. They begin by selecting a

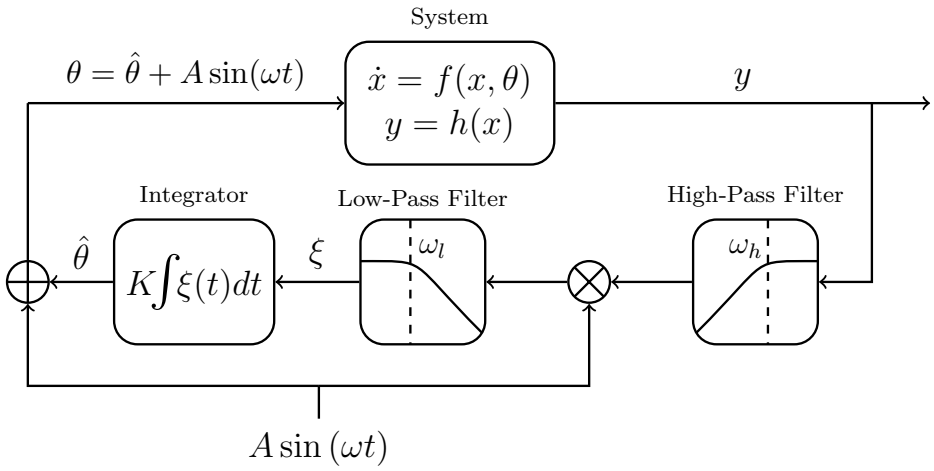


Figure 2.4: A block diagram of for a perturbation-based extremum-seeking controller. Note that ω_h and ω_l are the cut-off frequencies for the high- and low-pass filters, respectively, ω is the frequency of the dither signal, and the junction with a rotated cross represents the multiplication of the incident signals.

static, state-feedback control law parameterized by $\theta \in \mathbb{R}^m$ and introducing a periodic disturbance to the system by means of a dither signal with known frequency $\omega \in \mathbb{R}^m_{\geqslant 0}$. For the controller to be effective, the dither signal frequency must be in a range that provides a time-scale separation between itself and the plant's dynamics. Next, the system's output is passed through a high-pass filter to eliminate the signal's DC bias[1] to produce a sinusoid signal that will either be in or out of phase with the dither signal. Thus, the DC bias of the signals' product ξ – extracted by a low-pass filter – becomes a rough estimate of the gradient of an output sensitivity-based update law for θ. By integrating ξ, θ is driven along the gradient of h towards the optimum value θ^* that maximizes (minimizes) the performance (cost) function [84].

Once the proof of ESC's convergence was published for general nonlinear SISO systems, many more publications pertaining to ESC began to appear. Rotea [132]

[1] "When discussing periodic functions, the DC bias (a.k.a. DC component or DC offset) refers to the mean value of the waveform." Taken from: [73].

extended Krstić and Wang's results to the MIMO case and also demonstrated that the technique remained proficient when there was noise in the output signal. Following that, Choi et al. [23] developed an ESC framework for application in the discrete time setting for systems with stable and sufficiently fast dynamics. The work of Ariyur and Krstić [8] extended the application of PBESC to systems with time-varying extrema and, a few years later, developed a *slope-seeking* control architecture through a generalization of the PBESC scheme in [9]. Tan et al. [146] investigated the relationship between tuning parameters and the magnitude of the controller's stable region of attraction while the work of Chioua et al. [21] clarified the dependence of the optimal solution's error on the dither signal's frequency. More recently, a proportional-integral PBESC scheme was proposed by Guay [58] that eliminates the need for time-scale separation, greatly improving the performance of conventional PBESC and demonstrating that the size of the stable region of attraction is proportional to the ratio of the dither signal's amplitude to the controller's proportional gain.

Estimation-Based Extremum Seeking

The dependence on time-scale separation has long been the primary limitation of ESC schemes, since this often results in slower performance and longer convergence times. Prior to the publication of Guay's PI-PBESC in 2016, there was therefore a good deal of interest in finding ways to improve ESC's performance. This prompted the early work of Guay and Zhang [65], who developed an *estimation-based ESC* (EBESC) method capable of estimating the output's gradient directly using adaptive control methods. This family of ESC was confirmed to provide a better overall performance over PBESC as the gradient estimates were more accurate – the only trade-off being that EBESC

is more difficult to tune [60].

This initial EBESC technique was refined by DeHaan and Guay [31], who integrated the ability for constraint handling to the framework by introducing barrier functions to the cost. In the years that followed, the minimum perturbation required for convergence of the EBESC algorithm was investigated by Adetola and Guay [1]. The convergence of parameter estimates in finite-time, in addition to the robustness of the technique, was also explored in the work of Adetola and Guay [2]. The powerful result developed by Nešić et al. [116] indicated that a functional ESC can be constructed by combining any arbitrary parameter estimation routine with an optimization algorithm. Making increased tuning difficulties associated with EBESC less prominent was the work of Moshksar and Guay [108] which proposed an ESC algorithm that only required one tuning parameter.

In the developments of Moase et al. [104] and Ghaffari et al. [51], an optimization algorithm based on Newton's method that additionally approximated the output's Hessian also provided an ESC with increased performance. However, ESC's reliance on time-scale separation was not eliminated for the first time until the work of Zhang and Ordóñez [159]. Unfortunately, their method utilized a state regulator that required measurements of both the output and states to function correctly. By establishing a PI control law for EBESC applied to control-affine nonlinear systems, Guay and Dochain [62] were capable of removing the need for a time-scale separation without requiring state measurements. After EBESC was extended to the discrete-time setting by Guay [57], the PI version was also formulated for application to control-affine, nonlinear discrete-time systems by Guay and Burns [61].

2.3.2 Distributed Extremum-Seeking Control

ESC is a model-free control framework that relies on output-feedback, making it very well positioned for handling distributed control problems. Since no explicit knowledge of the cost function, the system's states or its dynamics is needed for the selection of inputs, a cooperative distributed setting would only require information relating to local costs or local inputs to be shared amongst the agents of a MAS.

Early work on the topic of distributed ESC relied primarily on the sharing of local cost information over a suitable communication network, which allowed agents to estimate the system's *average* total cost by means of a consensus algorithm (such as those discussed in Section 2.1.2). As the local estimates converge to the true average cost, agents can work collaboratively to drive the total cost to the desired optimum through the use perturbation-based ESCs. For instance, Li et al. outlined a scheme for application to mobile communication networks and proved the distributed controller's convergence to the unknown extremum in [89]. Another framework with proven convergence is applied to an optimal resource allocation scenario by Poveda and Quijano [125] and a similar method proposed by Xu and Soh incorporates constraints to the distributed controller in [157]. A distributed ESC for application to the maximization of power generation in wind farms is presented with proven practical stability by Menon and Baras [102] and another developed by Lia et al. in [90] is concerned with the problem of multi-vehicle formation control.

The wind farm problem discussed by Menon and Baras [102] helped spur the initial development of an estimation-based distributed ESC by Ebegbulem, who presented a complete proof of the architecture's stability and convergence in Ebegbulem [40]. Its

application to the maximization of power generation in wind farms, presented in Ebegbulem [42], saw a significant improvement in the system's closed-loop performance. Another distributed controller based on estimation-based proportional-integral ESC was developed by Vandermeulen et al. for application to nonlinear control-affine systems with unstable dynamics for the continuous-time case in Guay et al. [56] and for discrete-time settings in Vandermeulen et al. [149]. Of particular interest to Vandermeulen was the problem of high-altitude balloon formation control, who applied the developed distributed ESC to the scenario with great success in Vandermeulen et al. [150, 151] for the continuous- and discrete-time cases, respectively. For a complete and detailed proof of the framework's convergence and stability, the reader is referred to Vandermeulen [148].

In the work of Ye and Hu [158], a distributed perturbation-based ESC is developed for constrained optimization and its capacities for smart-grid energy consumption control are investigated. In [124], Poveda et al. extends perturbation-based distributed ESC to account for systems with time-varying communication networks. An estimation-based distributed ESC for nonlinear systems with costs based on the input effort rather than the state is proposed by Dougherty and Guay for scenarios in which local costs are communicated between agents in [33]. An architecture for systems with an input-based cost in which the inputs are shared by agents (rather than local costs) over an *unknown* network is outlined in the work of Ebegbulem [41].

The work of Vandermeulen et al. is of particular interest to the developments in the chapters that follow [149]. The discrete-time distributed ESC they propose will provide the foundation upon which the distributed ESC-based optimizer integrated in the proposed cloud-based MPC algorithm is built.

{ Chapter 3 }

Cloud-Based Model Predictive Control

As stated in Section 2.2.4, no techniques have been developed to date that consider distributing the conventional MPC problem between more than a plant's physical subsystems. In addition, of the DMPC frameworks that have been established in literature, most are not able to handle highly coupled nonlinear systems and only consider problems involving linear models [24]. While it is still quite useful in many cases to subdivide a plant into subsystems to be controlled by their assigned agents (as depicted in Figure 2.1), decentralizing the decision making process further could lead to even shorter computation times, allow MPC to be implemented in a broader range of applications, reduce the impact of local failures even further, and provide a flexible foundation for the framework to be installed on a cloud server or on other types of distributed computation systems. To accomplish this, each point on the MPC control horizon can be treated as an agent in a MAS. This allows a plant or one of its subsystems to decentralize the task of selecting a large number of optimal inputs to an even greater extent.

In this chapter, the general architecture for a fully distributed, *cloud-based* model predictive controller (CBMPC) is proposed for n^{th}-order unstable nonlinear systems. The resulting controller is one in which a central optimizer no longer has to solve for

the entire set of inputs making up the control horizon that minimizes the desired objective function. This technique allows a properly configured distributed computation system to consider prediction horizons of an arbitrarily large length without causing a significant increase in computation time. Instead of having a central optimizer to minimize a cost function at each iteration, agents are used to select one input vector each and need only consider the impact of their assigned input on a local estimate of the average total cost (ATC). By sharing local cost information with the network and using the data to continuously update their respective estimates, agents can approximate the impact of their contributions on the global system and select their inputs to cooperatively minimize the total cost – without needing to know it explicitly.

This work is based on the integration of decentralized, time-varying extremum-seeking control methods with conventional nonlinear MPC schemes. The CBMPC architecture outlined in the sections that follow is considered for application to nonlinear, discrete-time systems and establishes a firm foundation for installation on a cloud server. It can be nested in a subsystem-based DMPC framework to decentralize the MPC to the greatest possible extent. In addition, it can be adapted for application to economic MPC formulations.

In Section 3.1, a formal description of the problem is provided. Following that the CBMPC framework is defined in Section 3.2, which outlines the various optimization subroutines and control algorithms used by the controller and how they interact. Then, Section 3.3 describes a flexible method for accommodating constraints without having to make any changes to the previously defined optimization algorithms or control laws. Lastly, Section 3.4 provides a brief summary of the CBMPC algorithm.

Notation

In the developments that follow, the sampling instant is denoted by $k \in \mathbb{N}_0$ while iterations of any optimization routine carried out *at* time k is denoted by $[\kappa|k]$, with $\kappa \in \mathbb{N}_0$. The state and input vectors at time $k+i$, predicted at the sampling instant k, are formally denoted by $\boldsymbol{x}[k+i|k]$ and $\boldsymbol{u}[k+i|k]$, respectively. For notational convenience, $\boldsymbol{x}[k+i|k]$ is denoted by $\boldsymbol{x}_i[k]$ and $\boldsymbol{u}[k+i|k]$ by $\boldsymbol{u}_i[k]$. The collection of predicted state and input vectors at sampling instant k, denoted by $\mathbf{X}$ and $\mathbf{U}$, respectively, are defined as

$$\begin{aligned}\mathbf{X}[k] &= \{\boldsymbol{x}[k|k], \boldsymbol{x}[k+1|k], \ldots, \boldsymbol{x}[k+p|k]\} \\ &= \{\boldsymbol{x}_0[k], \boldsymbol{x}_1[k], \ldots, \boldsymbol{x}_p[k]\}\end{aligned} \tag{3.1}$$

$$\begin{aligned}\mathbf{U}[k] &= \{\boldsymbol{u}[k|k], \boldsymbol{u}[k+1|k], \ldots, \boldsymbol{u}[k+p-1|k]\} \\ &= \{\boldsymbol{u}_0[k], \boldsymbol{u}_1[k], \ldots, \boldsymbol{u}_{p-1}[k]\}\end{aligned} \tag{3.2}$$

Note that $\mathbf{X}$ includes the measured state $\boldsymbol{x}_0$. References to individual elements in a collection are indicated by a subscript (e.g. $\boldsymbol{x}_1[k] \Leftrightarrow \mathbf{X}_2[k]$) while references to a connected set of elements are shown as subscripts separated by a colon. For instance, $\mathbf{U}_{2:p}[k]$ denotes the series of elements $\{\boldsymbol{u}_1[k], \ldots, \boldsymbol{u}_{p-1}[k]\}$. Unless stated otherwise $\|\cdot\|$ indicates the Euclidean 2-norm. The weighted inner product of a vector $\boldsymbol{x} \in \mathbb{R}^n$, given by $\boldsymbol{x}^\top \mathbf{P} \boldsymbol{x}$ where $\mathbf{P} \in \mathcal{M}_{n\times n}(\mathbb{R})$ denotes a symmetric weighting matrix, is denoted by $\|\boldsymbol{x}\|_{\mathbf{P}}^2$. Changes in a variable from one time step to the next are denoted by $\Delta \cdot [k] = \cdot[k+1] - \cdot[k]$, while $\Delta \cdot [\kappa|k] = \cdot[\kappa+1|k] - \cdot[\kappa|k]$ designates the change in a variable from one step to the next of an optimization routine.

3.1 Problem Description

The proposed architecture for CBMPC is built on two optimization routines. The first solves the standard MPC problem and determines the set of inputs that minimizes an appropriate cost function via a distributed extremum-seeking controller (DESC). A network of p agents will be used to represent each point on the MPC control horizon; this concept, along with the corresponding input/output pairing assignments, is illustrated in Figure 3.1. The second consists of an independent, terminal extremum-seeking controller (TESC) that supplies the DESC's terminal agent (representing the last step on the control horizon) with an initial condition to be used at the DESC's next invocation. This accommodates the intrinsic receding-horizon nature of MPC and ensures that the terminal agent's optimization trajectory at time $k+1$ is directed towards a solution that decreases the terminal cost further.

3.1.1 Defining Local Objective Functions

CBMPC is considered for application to nonlinear systems with global dynamics described by discrete-time models of the form

$$\boldsymbol{x}[k+1] = \boldsymbol{x}[k] + \boldsymbol{f}\left(\boldsymbol{x}[k], \boldsymbol{u}[k]\right) \tag{3.3}$$

where $k \in \mathbb{N}_0$ is the sampling instant, $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^n$ is the system's state and $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^m$ is the input. It is assumed that the vector-valued function $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is smooth and that $\mathbb{X}$ and $\mathbb{U}$ are both connected and compact. Note that this architecture only considers scenarios with prediction and control horizons *of equal length*. It must be noted that this design constraint can be relaxed but at the cost of increasing the

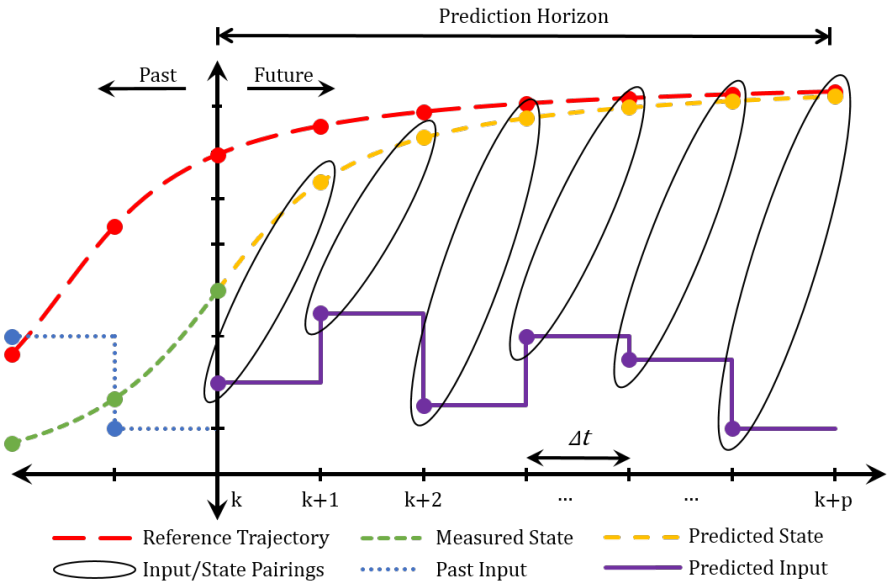


Figure 3.1: Each input-state pairing depicted above shows which state on the prediction horizon responds to which input. Each agent manipulates their assigned input to minimize the contribution of its local output, a function of the input-state pair, to the total cost.

complexity and, hence, the description of the approach.

To keep the framework as flexible as possible, general design criteria for local cost functions are established and summarized in Definition 3.3. The description of the technique allows any tailored cost appropriate for a given application to be selected at the discretion of the user.

Definition 3.1 – The Feasible Set of Discrete Input Velocities

Let the difference between two feasible inputs be given by $\widetilde{\boldsymbol{u}} = \boldsymbol{u}_1 - \boldsymbol{u}_2$, *with* $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathbb{U}$. *Then, the set of all such differences is given by the Minkowski difference* $\widetilde{\mathbb{U}} = \mathbb{U} \ominus \mathbb{U}$. *As* $\mathbb{U}$ *is non-empty, connected and compact, it follows that* $\widetilde{\mathbb{U}}$ *is also non-empty, connected and compact. In addition,* $\widetilde{\mathbb{U}}$ *is such that* $\mathbf{0} \in \widetilde{\mathbb{U}}$.

Definition 3.2 – The Steady-State Static Map

Let $\mathcal{S} = \{ (\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{X} \times \mathbb{U} \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \mathbf{0} \}$ *be the steady-state manifold for the nonlinear*

system (3.3). *It is assumed that for each* $\boldsymbol{x} \in \mathbb{X}$ *there exists a unique input* $\boldsymbol{u} \in \mathbb{U}$ *for which the pair* $(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{S}$. *Thus, each feasible steady-state pair can be represented by a static map* $\boldsymbol{\pi} : \mathbb{U} \to \mathbb{X}$.

Definition 3.3 – The General Local Stage Cost

Let the sets $\mathbb{X}$, $\mathbb{U}$, *and* $\widetilde{\mathbb{U}}$ *be defined as they are above. Then, the general local stage cost* ℓ *can be defined as the scalar function*

$$\ell := \begin{cases} \mathbb{X} \times \mathbb{U} \times \widetilde{\mathbb{U}} \to \mathbb{R} \\ (\boldsymbol{x}, \boldsymbol{u}, \widetilde{\boldsymbol{u}}) \quad \mapsto \ell_x(\boldsymbol{x}) + \ell_u(\boldsymbol{u}) + \ell_{\widetilde{u}}(\widetilde{\boldsymbol{u}}) + c \end{cases} \tag{3.4}$$

where $c \in \mathbb{R}$ *is a bias and*

i) $\ell_x : \mathbb{X} \to \mathbb{R}$, *the* state stage cost, *is chosen to be strongly convex and of class* $\boldsymbol{C}^2$;

ii) $\ell_u : \mathbb{U} \to \mathbb{R}$, *the* input stage cost, *is a strongly convex function of class* $\boldsymbol{C}^2$;

iii) $\ell_{\widetilde{u}} : \widetilde{\mathbb{U}} \to \mathbb{R}_{\geqslant 0}$, *the* input velocity stage cost, *is a strictly increasing, positive definite function of class* $\boldsymbol{C}^2$ *for which* $\operatorname{argmin}\{\ell_{\widetilde{u}}(\widetilde{\boldsymbol{u}})\} = \mathbf{0}$ *is the function's unique minimizer and* $\ell_{\widetilde{u}}(\mathbf{0}) = 0$.

While there are many functions that meet the requirements of Definition 3.3, for the sake of simplicity local costs inspired from a typical nonlinear MPC objective function are considered for the remainder of this work. This typical cost function is given by

$$J[k] = \sum_{i=0}^{p-1} \|\boldsymbol{x}_i[k] - \boldsymbol{x}_{sp}\|_{\mathbf{Q}}^2 + \|\boldsymbol{u}_i[k] - \boldsymbol{u}_{sp}\|_{\mathbf{R}}^2 + \sum_{i=0}^{p-2} \|\boldsymbol{u}_{i+1}[k] - \boldsymbol{u}_i[k]\|_{\mathbf{S}}^2 + \|\boldsymbol{x}_p[k] - \boldsymbol{x}_{sp}\|_{\mathbf{P}}^2 \tag{3.5}$$

where the subscript "*sp*" denotes a set point, $\mathbf{Q} \in \mathcal{M}_{n\times n}(\mathbb{R}_{\geq 0})$, $\mathbf{R} \in \mathcal{M}_{m\times m}(\mathbb{R}_{>0})$ and $\mathbf{S} \in \mathcal{M}_{m\times m}(\mathbb{R}_{>0})$ are diagonal weighting matrices and the terminal weight matrix $\mathbf{P} \in \mathcal{M}_{n\times n}(\mathbb{R})$ is symmetric positive-definite and solves the discrete-time algebraic

Riccati equation:

$$\mathbf{P} = \mathbf{A}^\top \mathbf{P}\mathbf{A} - \left(\mathbf{A}^\top \mathbf{P}\mathbf{B}\right)\left(\mathbf{R} + \mathbf{B}^\top \mathbf{P}\mathbf{B}\right)^{-1}\left(\mathbf{B}^\top \mathbf{P}\mathbf{A}\right) + \mathbf{Q} \tag{3.6}$$

As explained in Section 2.2.2, the term weighted by $\mathbf{P}$ (the terminal cost), is required for closed-loop stability of the MPC controller. The matrices $\mathbf{A} \in \mathcal{M}_{n\times n}(\mathbb{R})$ and $\mathbf{B} \in \mathcal{M}_{n\times m}(\mathbb{R})$ above are found by linearizing (3.3) about the desired steady-states to produce a linear time-invariant (LTI) state-space approximation of the dynamics in the form

$$\Delta \boldsymbol{x}[k] \cong \mathbf{A}\boldsymbol{x}[k] + \mathbf{B}\boldsymbol{u}[k] \tag{3.7}$$

It is desired to express the total MPC cost as a sum of agents' local costs, as expressed by

$$J\,[k] = \sum_{i=1}^{p} y_i[k] \tag{3.8}$$

To facilitate the incorporation of constraints in the resulting controller, the quadratic input error term in (3.5) ($\|\boldsymbol{u}_i[k] - \boldsymbol{u}_{sp}\|^2_{\boldsymbol{R}}$) is generalized to a convex, scalar function of the input $\Upsilon : \mathbb{U} \to \mathbb{R}$. Section 3.3 provides a more detailed explanation for this choice. Thus, the proposed local cost function for the DESC's agents is given by (3.9).

$$y_i[k] = y_0[k] + \|\boldsymbol{x}_i[k+1] - \boldsymbol{x}_{sp}\|^2_{\boldsymbol{Q}_i} + \|\Delta \boldsymbol{u}_i[k]\|^2_{\mathbf{S}} + \Upsilon\left(\boldsymbol{u}_i[k]\right) \tag{3.9}$$

$$y_0 = \frac{1}{p}\,\|\boldsymbol{x}_0[k] - \boldsymbol{x}_{sp}\|^2_{\mathbf{Q}} \tag{3.10}$$

The bias term y_0 above stems from the measured state's contribution to the total cost. With the chosen input/output pairings (see Figure 3.1) the measured state remains unaccounted for and so its contribution is simply distributed evenly between the agents.

Note that agent p's state's contribution corresponds to the terminal cost and requires the terminal weighting matrix $\mathbf{P}$. Thus, the state argument in (3.9) is assigned the agent-based weighting matrix $\boldsymbol{Q}_i$, defined here as

$$\boldsymbol{Q}_i = \begin{cases} \mathbf{Q} & i \in \{\, 1, \ldots, p-1 \,\} \\ \mathbf{P} & i = p \end{cases} \tag{3.11}$$

3.1.2 Generating an Input/Output Model

The goal of the DESC's agents is to take an initial guess for the input, $\boldsymbol{u}_i^0[k]$, and drive it to an optimal input $\boldsymbol{u}_i^*[k]$ using the DESC algorithm. The notation $[\kappa|k]$ will be used to denote iteration κ of an optimization routine invoked at sampling instant k. The measured initial condition $\boldsymbol{x}_0[k]$ is held constant throughout the DESC computation cycle. Therefore, the optimization dynamics are given by

$$\begin{aligned} \boldsymbol{x}_{i+1}[\kappa|k] &= \boldsymbol{x}_i[\kappa|k] + \boldsymbol{f}\left(\boldsymbol{x}_i[\kappa|k]\,, \boldsymbol{u}_i[\kappa|k]\right) && (3.12) \\ &= \boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,, \boldsymbol{u}_i[\kappa|k]\,, \boldsymbol{u}_{i-1}[\kappa|k]\,, \ldots, \boldsymbol{u}_0[\kappa|k]\right) \\ &= \boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,, \mathbf{U}_{0:i}[\kappa|k]\right) && (3.13) \\ y_i[\kappa|k] &= y_0[k] + \|\boldsymbol{x}_{i+1}[\kappa|k] - \boldsymbol{x}_{sp}\|^2_{\boldsymbol{Q}_i} + \|\widetilde{\boldsymbol{u}}_i[\kappa|k]\|^2_{\mathbf{S}} + \Upsilon\left(\boldsymbol{u}_i[\kappa|k]\right) && (3.14) \\ &= y_0[k] + \|\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,, \mathbf{U}_{0:i}[\kappa|k]\right) - \boldsymbol{x}_{sp}\|^2_{\boldsymbol{Q}_i} + \|\widetilde{\boldsymbol{u}}_i[\kappa|k]\|^2_{\mathbf{S}} + \Upsilon\left(\boldsymbol{u}_i[\kappa|k]\right) && (3.15) \end{aligned}$$

where $\widetilde{\boldsymbol{u}}_i[\kappa|k] = \boldsymbol{u}_i[\kappa|k] - \boldsymbol{u}_{i-1}[\kappa|k]$. For the first agent, this difference is $\widetilde{\boldsymbol{u}}_0[\kappa|k] = \boldsymbol{u}_0[\kappa|k] - \boldsymbol{u}_0^*[k-1]$ as there are no agents preceding it (where $\boldsymbol{u}_0^*[k-1]$ is the input implemented at the previous sampling instant).

To state the DESC's objective more formally with respect to the dynamics above,

agents aim to drive their assigned state-input pairs to equilibria $(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*)$ that collectively minimize the total cost $J[k]$. As the measured initial condition is held constant while the optimization is carried out, the predicted output trajectory will only be actively affected by local inputs and their discrete velocities. Placing (3.15) in general form, we have that

$$\begin{aligned} y_i[\kappa|k] &= y_0[k] + \|\widetilde{\boldsymbol{u}}_i[\kappa|k]\|_{\mathbf{S}}^2 + \|\boldsymbol{g}_i\left(\boldsymbol{x}_0[k], \mathbf{U}_{0:i}[\kappa|k]\right) - \boldsymbol{x}_{sp}\|_{\boldsymbol{Q}_i}^2 + \Upsilon\left(\boldsymbol{u}_i[\kappa|k]\right) \\ &= c + \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa|k]\right) + \ell_x\left(\boldsymbol{x}_{i+1}[\kappa|k]\right) + \ell_u\left(\boldsymbol{u}_i[\kappa|k]\right) \end{aligned}$$

Here, the stage costs are chosen to be $\ell_x\left(\boldsymbol{x}_{i+1}\right) \triangleq \|\boldsymbol{g}_i\left(\boldsymbol{x}_0[k], \mathbf{U}_{0:i}[\kappa|k]\right) - \boldsymbol{x}_{sp}\|_{\boldsymbol{Q}_i}^2$, $\ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i\right) \triangleq \|\widetilde{\boldsymbol{u}}_i[\kappa|k]\|_{\mathbf{S}}^2$, and $\ell_u\left(\boldsymbol{u}_i\right) \triangleq \Upsilon\left(\boldsymbol{u}_i[\kappa|k]\right)$ with the bias $c = y_0[k]$. Note that these selections for ℓ_x, ℓ_u and $\ell_{\widetilde{u}}$ fully comply with the requirements of Definition 3.3.

The control action that drives the predicted inputs to their optimal solutions is based on a gradient descent control law. A model of the local cost dynamics with respect to the input is therefore required. To begin developing a suitable input-output map, certain assumptions are required.

Assumption 3.4 – Stage Cost Convexity

By Definitions 3.2 and 3.3, we have that $\operatorname{argmin}\left\{\ell_x\left(\boldsymbol{x}\right)\right\} = \boldsymbol{x}^* = \boldsymbol{\pi}\left(\boldsymbol{u}^*\right)$. *Thus, for all* $\boldsymbol{u} \in \mathbb{U}$, *we have that for some* $\beta_1 \in \mathbb{R}_{>0}$

$$\left(\boldsymbol{u} - \boldsymbol{u}^*\right)^\top \left.\frac{\partial \ell_x\left(\boldsymbol{\pi}\left(\boldsymbol{u}\right)\right)}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}} \geq \beta_1 \left\|\boldsymbol{u} - \boldsymbol{u}^*\right\|^2$$

With that, we require that ℓ_u *be constructed such that* $\operatorname{argmin}\left\{\ell_u\left(\boldsymbol{u}\right)\right\} = \boldsymbol{u}^*$. *Therefore, for all* $\boldsymbol{u} \in \mathbb{U}$, *we have that for some* $\beta_2 \in \mathbb{R}_{>0}$

$$\left(\boldsymbol{u} - \boldsymbol{u}^*\right)^\top \left.\frac{\partial \ell_u\left(\boldsymbol{u}\right)}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}} \geq \beta_2 \left\|\boldsymbol{u} - \boldsymbol{u}^*\right\|^2$$

By the convexity of ℓ_x and ℓ_u, and the definition of $\ell_{\widetilde{u}}$, for all $\boldsymbol{u} \in \mathbb{U}$ and for some $\beta \in \mathbb{R}_{>0}$, the local steady-state cost is therefore such that

$$(\boldsymbol{u}-\boldsymbol{u}^*)^\top \left.\frac{\partial \ell\left(\boldsymbol{\pi}\left(\boldsymbol{u}\right),\boldsymbol{u},(\boldsymbol{u}-\boldsymbol{u}^*)\right)}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}} \geq \beta \left\|\boldsymbol{u}-\boldsymbol{u}^*\right\|^2$$

Assumption 3.5 – Existence of the Local Cost Minimum

Let $\widetilde{\mathbb{U}}$ be defined as in Definition 3.1 and let the local stage cost be defined as in Definition 3.3. Letting Assumption 3.4 hold with $\boldsymbol{x}^* = \boldsymbol{\pi}\left(\boldsymbol{u}^*\right) \in \mathbb{X}$, it is assumed that

i) $\left.\dfrac{\partial \ell\left(\boldsymbol{\pi}\left(\boldsymbol{u}\right),\boldsymbol{u},(\boldsymbol{u}-\boldsymbol{u}^*)\right)}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}^*} = \boldsymbol{0}$

ii) $\left.\dfrac{\partial^2 \ell\left(\boldsymbol{\pi}\left(\boldsymbol{u}\right),\boldsymbol{u},(\boldsymbol{u}-\boldsymbol{u}^*)\right)}{\partial \boldsymbol{u}\partial \boldsymbol{u}^\top}\right|_{\boldsymbol{u}} > \alpha I,\ \forall\ \boldsymbol{u} \in \mathbb{U},$ *where* $\alpha \in \mathbb{R}_{>0}$

Assumption 3.6 – Local Boundedness of the Gradient and Hessian

By the compactness of $\widetilde{\mathbb{U}}$, we have that there exists some $c \in \mathbb{R}_{\geqslant 0}$ such that

$$\sup_{\widetilde{\boldsymbol{u}} \in \widetilde{\mathbb{U}}} \{\|\widetilde{\boldsymbol{u}}\|\} = \max_{\widetilde{\boldsymbol{u}} \in \widetilde{\mathbb{U}}} \{\|\widetilde{\boldsymbol{u}}\|\} = c < \infty$$

It is thus assumed that there exists some $L_0,\ L_1,\ L_2 \in \mathbb{R}_{>0}$ for which

i) $\|y_i\| \leq \left\|\ell\left(\boldsymbol{x},\boldsymbol{u},\widetilde{\boldsymbol{u}}\right)\right\| \leq L_0$

ii) $\left\|\dfrac{\partial \ell\left(\boldsymbol{x},\boldsymbol{u},\widetilde{\boldsymbol{u}}\right)}{\partial \boldsymbol{u}}\right\| \leq L_1$

iii) $\left\|\dfrac{\partial^2 \ell\left(\boldsymbol{x},\boldsymbol{u},\widetilde{\boldsymbol{u}}\right)}{\partial \boldsymbol{u}\partial \boldsymbol{u}^\top}\right\| \leq L_2$

for all $\boldsymbol{u} \in \mathbb{U}$, for all $\widetilde{\boldsymbol{u}} \in \widetilde{\mathbb{U}}$, and for all $\boldsymbol{x} \in \mathbb{X}$.

Note that if ℓ is chosen to be smooth, that the boundedness of its gradient and Hessian follows directly from its definition and Assumption 3.6 is not required. However, the assumption is still included here for completeness in the event that the user

wishes to use a non-smooth selection for ℓ. Using the definitions above, the local cost dynamics can be written as

$$\begin{aligned}\Delta y_i[\kappa|k] &= y_i[\kappa+1|k] - y_i[\kappa|k] \\ &= \ell\big(\boldsymbol{x}_{i+1}[\kappa+1|k], \boldsymbol{u}_i[\kappa+1|k], \widetilde{\boldsymbol{u}}_i[\kappa+1|k]\big) - \ell\big(\boldsymbol{x}_{i+1}[\kappa|k], \boldsymbol{u}_i[\kappa|k], \widetilde{\boldsymbol{u}}_i[\kappa|k]\big)\end{aligned}$$

Using Definition 3.3 to break the local cost into its constituent stage costs yields

$$\begin{aligned}\Delta y_i[\kappa|k] = &\Big(y_0[k] + \ell_x\left(\boldsymbol{x}_{i+1}[\kappa+1|k]\right) + \ell_u\left(\boldsymbol{u}_i[\kappa+1|k]\right) + \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)\Big) \\ &- \Big(y_0[k] + \ell_x\left(\boldsymbol{x}_{i+1}[\kappa|k]\right) + \ell_u\left(\boldsymbol{u}_i[\kappa|k]\right) + \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa|k]\right)\Big)\end{aligned}$$

Next, substituting in (3.13) produces

$$\begin{aligned}\Delta y_i[\kappa|k] = &\ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k], \mathbf{U}_{0:i}[\kappa+1|k]\right)\big) - \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k], \mathbf{U}_{0:i}[\kappa|k]\right)\big) + \ell_u\left(\boldsymbol{u}_i[\kappa+1|k]\right) \\ &- \ell_u\left(\boldsymbol{u}_i[\kappa|k]\right) + \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right) - \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa|k]\right)\end{aligned}$$

If agents are to select solutions that collectively minimize the total cost, the optimization problem cannot be based on the direct minimization of local costs. In this case, the best possible outcome would have agents settle on a Nash Equilibrium. While this constitutes the best possible solution in certain restricted communication scenarios, it is not the most optimal result for this application. Since explicit knowledge of the total cost may not be available to all agents, they can gauge optimality based on local estimates of the *average* total cost (ATC).For the local cost definition provided above, the dynamics of the ATC as seen by one agent can be described as

$$\begin{aligned}\frac{1}{p}\Delta J[\kappa|k] &= \frac{1}{p}J[\kappa+1|k] - \frac{1}{p}J[\kappa|k] \\ &= \frac{1}{p}\sum_{i=1}^{p} y_i[\kappa+1|k] - y_i[\kappa|k] = \frac{1}{p}\sum_{i=1}^{p}\Delta y_i[\kappa|k]\end{aligned}$$

$$= \frac{1}{p}\sum_{j\neq i}\Delta y_j[\kappa|k] + \frac{1}{p}\Big(\ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\mathbf{U}_{0:i}[\kappa+1|k]\right)\big) + \ell_u\left(\boldsymbol{u}_i[\kappa+1|k]\right)$$
$$+ \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right) - \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\mathbf{U}_{0:i}[\kappa|k]\right)\big) - \ell_u\left(\boldsymbol{u}_i[\kappa|k]\right) - \ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i[\kappa|k]\right)\Big)$$

Recalling that $\widetilde{\boldsymbol{u}}_i = \boldsymbol{u}_i - \boldsymbol{u}_{i-1}$, we have that

$$\begin{aligned}\Delta\widetilde{\boldsymbol{u}}_i &= \left(\boldsymbol{u}_i[\kappa+1|k] - \boldsymbol{u}_{i-1}[\kappa+1|k]\right) - \left(\boldsymbol{u}_i[\kappa|k] - \boldsymbol{u}_{i-1}[\kappa|k]\right)\\ &= \left(\boldsymbol{u}_i[\kappa+1|k] - \boldsymbol{u}_i[\kappa|k]\right) - \left(\boldsymbol{u}_{i-1}[\kappa+1|k] - \boldsymbol{u}_{i-1}[\kappa|k]\right)\\ &= \Delta\boldsymbol{u}_i[\kappa|k] - \Delta\boldsymbol{u}_{i-1}[\kappa|k]\end{aligned}$$

Hence, the change in the input velocity stage cost can be expressed as

$$\begin{aligned}\Delta\ell_{\widetilde{u}}\left(\widetilde{\boldsymbol{u}}_i\right) &= \left(\Delta\widetilde{\boldsymbol{u}}_i[\kappa|k]\right)^\top\int_0^1 \frac{\partial\ell_{\widetilde{u}}\left(\left(1-\delta\right)\widetilde{\boldsymbol{u}}_i[\kappa|k] + \delta\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)}{\partial\widetilde{\boldsymbol{u}}_i}d\delta\\ &= \left(\Delta\boldsymbol{u}_i[\kappa|k] - \Delta\boldsymbol{u}_{i-1}[\kappa|k]\right)^\top\int_0^1 \frac{\partial\ell_{\widetilde{u}}\left(\left(1-\delta\right)\widetilde{\boldsymbol{u}}_i[\kappa|k] + \delta\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)}{\partial\widetilde{\boldsymbol{u}}_i}d\delta\end{aligned}$$

Next, consider the change in the state stage cost given by

$$\begin{aligned}\Delta\ell_x\left(\boldsymbol{x}_{i+1}\right) &= \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\mathbf{U}_{0:i}[\kappa+1|k]\right)\big) - \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\mathbf{U}_{0:i}[\kappa|k]\right)\big)\\ &= \Delta\boldsymbol{u}_0^\top[\kappa|k]\int_0^1 \frac{\partial\ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial\boldsymbol{u}_0}d\delta\\ &\quad + \cdots + \Delta\boldsymbol{u}_i^\top[\kappa|k]\int_0^1 \frac{\partial\ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial\boldsymbol{u}_i}d\delta\\ &= \sum_{j=0}^{i-1}\left(\Delta\boldsymbol{u}_j^\top[\kappa|k]\int_0^1 \frac{\partial\ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial\boldsymbol{u}_j}d\delta\right)\\ &\quad + \Delta\boldsymbol{u}_i^\top[\kappa|k]\int_0^1 \frac{\partial\ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial\boldsymbol{u}_i}d\delta\end{aligned}$$

Lastly, the change in the input stage cost can be expressed as

$$\Delta\ell_u\left(\boldsymbol{u}_i\right) = \Delta\boldsymbol{u}_i^\top[\kappa|k]\int_0^1 \frac{\partial\ell_u\left(\left(1-\delta\right)\boldsymbol{u}_i[\kappa|k] + \delta\boldsymbol{u}_i[\kappa+1|k]\right)}{\partial\boldsymbol{u}_i}d\delta$$

Combining all these expressions together allows the locally observed change in the ATC to be written as

$$\begin{aligned}\frac{1}{p}\Delta J[\kappa|k] = \frac{1}{p}\Bigg(&\sum_{j\neq i}\Delta y_j[\kappa|k] - \Delta\boldsymbol{u}_{i-1}^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_{\widetilde{u}}\left(\left(1-\delta\right)\widetilde{\boldsymbol{u}}_i[\kappa|k] + \delta\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)}{\partial \widetilde{\boldsymbol{u}}_i}d\delta \\ &+ \sum_{j=0}^{i-1}\Delta\boldsymbol{u}_j^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial \boldsymbol{u}_j}d\delta \\ &+ \Delta\boldsymbol{u}_i^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_{\widetilde{u}}\left(\left(1-\delta\right)\widetilde{\boldsymbol{u}}_i[\kappa|k] + \delta\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)}{\partial \widetilde{\boldsymbol{u}}_i}d\delta \\ &+ \Delta\boldsymbol{u}_i^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial \boldsymbol{u}_i}d\delta \\ &+ \Delta\boldsymbol{u}_i^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_u\left(\left(1-\delta\right)\boldsymbol{u}_i[\kappa|k] + \delta\boldsymbol{u}_i[\kappa+1|k]\right)}{\partial \boldsymbol{u}_i}d\delta\Bigg)\end{aligned}$$

As a result, a local drift parameter $\theta_i^0[\kappa|k]$ can be defined as

$$\begin{aligned}\theta_i^0[\kappa|k] = \frac{1}{p}\Bigg(&\sum_{j\neq i}\Delta y_j[\kappa|k] - \Delta\boldsymbol{u}_{i-1}^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_{\widetilde{u}}\left(\left(1-\delta\right)\widetilde{\boldsymbol{u}}_i[\kappa|k] + \delta\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)}{\partial \widetilde{\boldsymbol{u}}_i}d\delta \\ &+ \sum_{j=0}^{i-1}\Delta\boldsymbol{u}_j^{\top}[\kappa|k]\int_0^1 \frac{\partial \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial \boldsymbol{u}_j}d\delta\Bigg)\end{aligned} \tag{3.16}$$

while the gradient of the locally observed ATC with respect agent i's input is given by

$$\begin{aligned}\boldsymbol{\theta}_i^1[\kappa|k] = \frac{1}{p}\Bigg(&\int_0^1 \frac{\partial \ell_x\big(\boldsymbol{g}_i\left(\boldsymbol{x}_0[k]\,,\left(1-\delta\right)\mathbf{U}_{0:i}[\kappa|k] + \delta\mathbf{U}_{0:i}[\kappa+1|k]\right)\big)}{\partial \boldsymbol{u}_i}d\delta \\ &+ \int_0^1 \frac{\partial \ell_u\left(\left(1-\delta\right)\boldsymbol{u}_i[\kappa|k] + \delta\boldsymbol{u}_i[\kappa+1|k]\right)}{\partial \boldsymbol{u}_i}d\delta \\ &+ \int_0^1 \frac{\partial \ell_{\widetilde{u}}\left(\left(1-\delta\right)\widetilde{\boldsymbol{u}}_i[\kappa|k] + \delta\widetilde{\boldsymbol{u}}_i[\kappa+1|k]\right)}{\partial \widetilde{\boldsymbol{u}}_i}d\delta\Bigg)\end{aligned} \tag{3.17}$$

Substituting these parameters into the expression for the change in the ATC provides

$$\begin{aligned}\frac{1}{p}\Delta J[\kappa|k] &= \theta_i^0[\kappa|k] + \boldsymbol{\theta}_i^{1\top}[\kappa|k]\,\Delta\boldsymbol{u}_i[\kappa|k] \\ &= \boldsymbol{\theta}_i^{\top}[\kappa|k]\,\boldsymbol{\phi}_i[\kappa|k] \end{aligned} \tag{3.18}$$

where $\boldsymbol{\theta}_i[\kappa|k] = \left[\theta_i^0[\kappa|k]\,, \boldsymbol{\theta}_i^{1\top}[\kappa|k]\right]^{\top}$ and $\boldsymbol{\phi}_i[\kappa|k] = \left[1, \Delta\boldsymbol{u}_i^{\top}[\kappa|k]\right]^{\top}$.

The parameterization (3.18) is exact and expresses the ATC's dynamics with respect to agent i's input only. This parametrization is considered in the development of an estimation strategy that allows each agent to estimate $\boldsymbol{\theta}_i$ based on local estimates of the ATC. By the continuity of $\ell\left(\boldsymbol{x}, \boldsymbol{u}, \widetilde{\boldsymbol{u}}\right)$, the boundedness of its gradient and Hessian over its domain (Assumption 3.6), and the definition of $\boldsymbol{\theta}_i$, it follows that

$$\left\|\Delta\boldsymbol{\theta}_i^1[\kappa]\right\| \leq L_2\left\|\Delta\boldsymbol{u}_i[\kappa+1]\right\| + L_2\left\|\Delta\boldsymbol{u}_i[\kappa]\right\| \quad \forall\, i \in \{1,\ldots,p\}$$

This argument provides a basis for the following assumption, which is required for the control algorithm to be stable [59].

Assumption 3.7 – Boundedness of the Input Signal

The input signal $\boldsymbol{u}_i[\kappa|k]$ *is such that*

$$\boldsymbol{u}_i[\kappa|k] \in \mathbb{U} \qquad \forall\ \kappa \geqslant 0,\ \forall\ k \geqslant 0$$

for each agent $i \in \{1,\ldots,p\}$.

3.2 The CBMPC Algorithm

The general algorithm employed in this study is summarized in Figure 3.2. At time k, the current state of the system, $\boldsymbol{x}_0[k]$, is measured and its contribution to the total

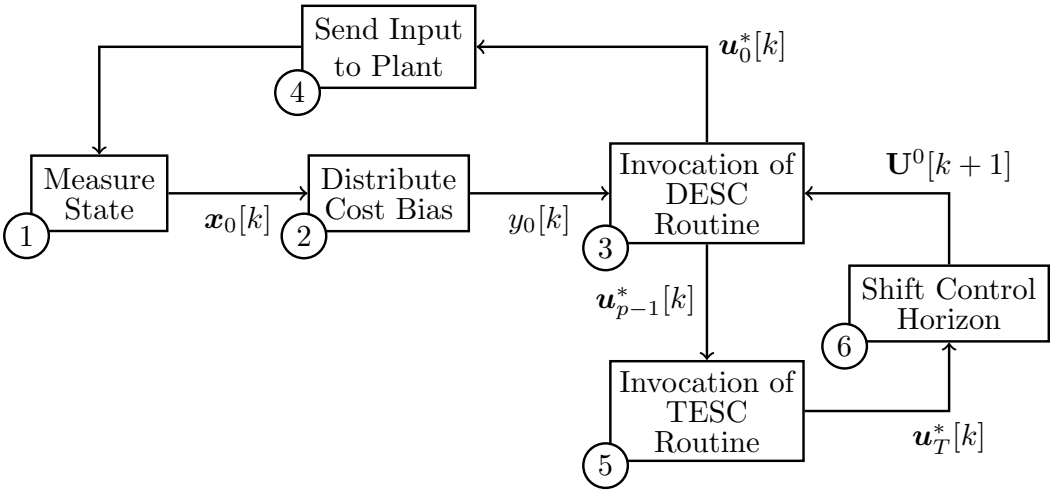


Figure 3.2: General flow diagram for the CBMPC architecture.

cost is determined. The local cost bias $y_0[k]$ that is communicated to each agent is determined by (3.10). Agents then invoke the DESC, outlined in Section 3.2.1, to drive the elements of an initial collection of inputs, denoted by $\mathbf{U}^0[k]$, towards one that further minimizes the agents' estimates of the ATC, $\widehat{\boldsymbol{J}} \in \mathbb{R}^p$. The routine runs for a total of κ_D iterations (or until a suitable termination condition is met), producing an adjusted collection of inputs, denoted by $\mathbf{U}^*[k]$, whose first element, $\boldsymbol{u}_0^*[k]$, is sent to the plant. Before beginning this cycle again, the initial input collection must be updated, providing $\mathbf{U}^0[k+1]$. This update is assisted by the TESC, which is tasked with selecting an input that minimizes the terminal cost at time $k+p+1$ (predicted at step k). The update process for $\mathbf{U}$ is described in detail in Section 3.2.2.

3.2.1 The DESC Optimization Routine

The DESC method used to select inputs is primarily based on the work of Vandermeulen et al. [149]. The routine has three components: a dynamic average consensus

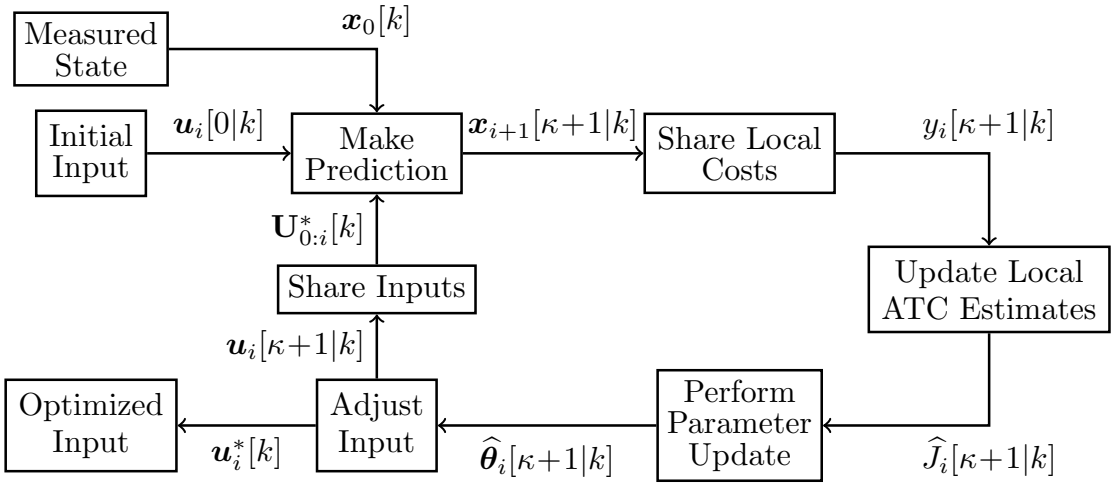


Figure 3.3: A block diagram of the DESC algorithm used to solve the MPC optimization problem.

algorithm that handles the agent's estimates of the ATC, a parameter estimation routine that allows agents to estimate the parameters $\boldsymbol{\theta}_i$ defined by (3.18), and a gradient-descent control law that tunes the inputs toward their optimal values. The block diagram in Figure 3.3 outlines the algorithm utilized by the DESC.

Dynamic Average Consensus

Since agents do not have an accurate knowledge of the total cost, a dynamic average consensus (DAC) algorithm provides agents with local estimates of the ATC to be minimized. As explained in Section 2.1.2, the DACA updates these estimates using local cost information shared over a suitable communication network and drives the estimates to a consensus that converges to the true ATC. The DACA employed by the CBMPC framework is provided by Kia et al. [81], where the ATC's dynamics are given

by

$$\begin{bmatrix} \Delta\widehat{\boldsymbol{J}}\left[\kappa|k\right] \\ \Delta\boldsymbol{\rho}\left[\kappa|k\right] \end{bmatrix} = \Delta t_{\mathrm{D}} \begin{bmatrix} -\beta_{\mathrm{P}}\mathbf{I} - \beta_{\mathrm{I}}\mathbf{L} & -\mathbf{I} \\ \beta_{\mathrm{P}}\beta_{\mathrm{I}}\mathbf{L} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{J}}\left[\kappa|k\right] \\ \boldsymbol{\rho}\left[\kappa|k\right] \end{bmatrix} + \Delta t_{\mathrm{D}} \begin{bmatrix} \beta_{\mathrm{P}}\mathbf{I} \\ \mathbf{0} \end{bmatrix} \boldsymbol{y}[\kappa|k] + \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \Delta\boldsymbol{y}\left[\kappa - 1|k\right] \tag{3.19}$$

where $\mathbf{L}$ is the selected network's Laplacian matrix, $\beta_{\mathrm{P}}, \beta_{\mathrm{I}} \in \mathbb{R}_{>0}$ are tuning parameters, Δt_{D} is the optimizer's update period, $\boldsymbol{\rho} \in \mathbb{R}^p$ is a vector of integrator variables and $\boldsymbol{y} \in \mathbb{R}^p$ is a vector of the local costs. Following the recommendation of Vandermeulen et al. [149] the tuning parameters are set to $\beta_{\mathrm{P}} = \Delta t_{\mathrm{D}}^{-1}$ and $\beta_{\mathrm{I}} = \left(\Delta t_{\mathrm{D}} \max\left(\deg(\mathbf{L})\right)\right)^{-1}$ as these values help improve the algorithm's performance in the distributed extremum-seeking controller. As explained in the work of Kia et al., the network's Laplacian must comply with Assumption 3.8 for (3.19) to converge.

Assumption 3.8 – Network Properties

The network is structured such that its graph is connected and undirected. Letting the eigenvalues of $\mathbf{L}$ *be denoted as* $0 = \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_p$*, as a consequence of connectedness we have that* $\lambda_2 \neq 0$*. In addition, by the definition of an undirected graph, the Laplacian will be such that* $\mathbf{L} = \mathbf{L}^\top$*.*

Further considerations are required to ensure suitable performance of the algorithm. For one, the first agent's input will have an impact on the local costs of every other agent. In addition, the terminal agent's state is weighted more heavily by $\mathbf{P}$ in its local cost. This could cause agent p's local cost to have a more significant impact on the ATC. This makes the ability to propagate local cost information through the network quickly desirable for agents 1 and p. Therefore, a network structured such

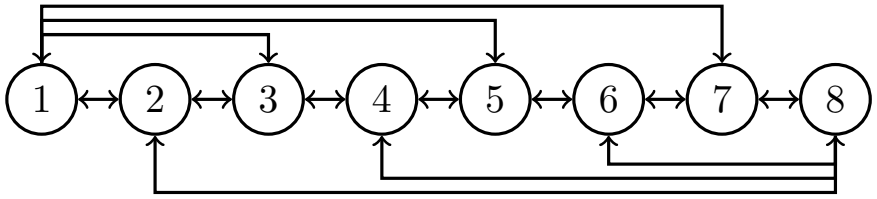


Figure 3.4: Structure of the communication network used to share local cost information for a system of 8 agents.

that each agent communicates with its immediate neighbor, with the first agent additionally communicating with all odd-numbered agents and the terminal agent also communicating with all even-numbered agents, is recommended. An example of this setup, valid for *even* numbers of total agents, is depicted in Figure 3.4.

Time-Varying Parameter Estimation

To estimate the gradient of the cost, agents will employ a RLS algorithm provided by Guay and Dochain [63] and Guay [57]. The estimation routine is defined by the following:

$$e_i[\kappa|k] = \Delta\widehat{J}_i[\kappa|k] - \Delta\widehat{y}_i[\kappa|k] \tag{3.20}$$

$$\Delta\widehat{y}_i[\kappa|k] = \widehat{\boldsymbol{\theta}}_i^{\top}[\kappa|k]\,\boldsymbol{\phi}_i[\kappa|k] \tag{3.21}$$

$$\boldsymbol{\Sigma}_i[\kappa+1|k] = \alpha\boldsymbol{\Sigma}_i[\kappa|k] + \boldsymbol{\phi}_i[\kappa|k]\,\boldsymbol{\phi}_i^{\top}[\kappa|k] + \alpha_s\mathbf{I} \qquad \boldsymbol{\Sigma}_i[0|k] = \sigma_0\mathbf{I} \tag{3.22}$$

$$\bar{\boldsymbol{\theta}}_i[\kappa|k] = \widehat{\boldsymbol{\theta}}_i[\kappa|k] + \frac{\boldsymbol{\Sigma}_i^{-1}[\kappa|k]\,\boldsymbol{\phi}_i[\kappa|k]\,e_i[\kappa|k]}{\alpha + \boldsymbol{\phi}_i^{\top}[\kappa|k]\,\boldsymbol{\Sigma}_i^{-1}[\kappa|k]\,\boldsymbol{\phi}_i[\kappa|k]} \tag{3.23}$$

$$\widehat{\boldsymbol{\theta}}_i[\kappa+1|k] = \mathrm{Proj}_{\gamma_\theta}\big\{\bar{\boldsymbol{\theta}}_i[\kappa|k]\big\} \qquad \widehat{\boldsymbol{\theta}}_i[0|k] = \theta_0\mathbf{1} \tag{3.24}$$

where e_i is the local output prediction error, $\alpha \in (0,1)$ is the forgetting factor for the local covariance matrix $\boldsymbol{\Sigma}_i$, $\alpha_s \in \mathbb{R}_{>0}$ is a filter parameter (typically < 1), and $\widehat{\boldsymbol{\theta}}_i$ is

the local estimate of $\boldsymbol{\theta}_i$. Note that the covariance matrix and vector of parameter estimates are initialized at $\boldsymbol{\Sigma}_i[0|k] = \sigma_0 \mathbf{I}$, where $\sigma_0 \in \mathbb{R}_{\geqslant 0}$, and $\widehat{\boldsymbol{\theta}}_i[0|k] = \theta_0 \mathbf{1}$, where $\theta_0 \in \mathbb{R}$, respectively. $\bar{\boldsymbol{\theta}}_i$ denotes the update of the estimate $\widehat{\boldsymbol{\theta}}_i$ before the latter is projected onto a ball of radius $\gamma_\theta \in \mathbb{R}_{>0}$ by the orthogonal projection operator Proj – which guarantees that $\widehat{\boldsymbol{\theta}}_i$ does not escape its uncertainty set by enforcing an upper bound on $\| \widehat{\boldsymbol{\theta}}_i \|$.

Any projection algorithm can be used so long as it meets the criteria outlined by Goodwin and Sin [53]. Namely, it must be constructed such that:

i) $\widehat{\boldsymbol{\theta}}_i[\kappa + 1] \in \Theta^0, \ \forall \ \kappa \geqslant 0$

ii) $\widetilde{\boldsymbol{\theta}}_i^\top[\kappa + 1] \, \boldsymbol{\Sigma}_i[\kappa + 1] \, \widetilde{\boldsymbol{\theta}}_i[\kappa + 1] \leq \bar{\widetilde{\boldsymbol{\theta}}}_i^\top[\kappa + 1] \, \boldsymbol{\Sigma}_i[\kappa + 1] \, \bar{\widetilde{\boldsymbol{\theta}}}_i[\kappa + 1]$

where $\Theta^0 = \mathbb{B}_{\gamma_\theta}$ is a ball of radius γ_θ, $\widetilde{\boldsymbol{\theta}}_i = \widehat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i$, and $\bar{\widetilde{\boldsymbol{\theta}}}_i = \bar{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i$. The recommended projection algorithm is supplied by the work of Guay et al. [59]:

Definition 3.9 – Orthogonal Lipschitz Projection

Let $\boldsymbol{\Sigma}_{Ch} = \text{Chol}\{\boldsymbol{\Sigma}_i[\kappa + 1]\}$ *denote the Cholesky factor of* $\boldsymbol{\Sigma}_i[\kappa + 1]$. *If* $\| \bar{\boldsymbol{\theta}}_i[\kappa + 1] \| \geq \gamma_\theta$, *perform the following:*

- *If* $\| \bar{\boldsymbol{\theta}}_i[\kappa + 1] \| \geq \gamma_\theta$, *perform the following:*

  i) Letting $\boldsymbol{\vartheta} = \dfrac{\gamma_\theta}{\| \bar{\boldsymbol{\theta}}_i \, [\kappa + 1] \|} \bar{\boldsymbol{\theta}}_i[\kappa + 1]$ *and* $z_\vartheta = \sqrt{\boldsymbol{\vartheta}^\top \boldsymbol{\Sigma}_i[\kappa + 1] \, \boldsymbol{\vartheta}}$

  ii) define $\bar{\boldsymbol{\varphi}} = \dfrac{z_\vartheta}{\| \boldsymbol{\varphi} \|} \boldsymbol{\varphi}$, *where* $\boldsymbol{\varphi} = \boldsymbol{\Sigma}_{Ch} \, \bar{\boldsymbol{\theta}}_i[\kappa + 1]$;

  iii) and then $\widehat{\boldsymbol{\theta}}_i[\kappa + 1] = \boldsymbol{\Sigma}_{Ch}^{-1} \, \bar{\boldsymbol{\varphi}}$.

- *Otherwise* $\widehat{\boldsymbol{\theta}}_i[\kappa + 1] = \bar{\boldsymbol{\theta}}_i[\kappa + 1]$.

For the parameter estimation routine to converge, Assumption 3.10 must be met. This requirement, known as a *persistence of excitation* (PE) condition, is prevalent in the fields of adaptive control theory and system identification. Detailed discussions of this assumption can be found in the works of Goodwin and Sin [53] and Ljung [92]. As stated in Section 2.3.1, a dither signal is used to meet this criterion in EBESC systems.

Assumption 3.10 – Persistence of Excitation

The input signals are such that there exist constants $T \in \mathbb{Z}_{>0}$ *and* $\gamma_\phi^-, \gamma_\phi^+ \in \mathbb{R}_{>0}$ *for which the following is true:*

$$\gamma_\phi^- \mathbf{I} < \frac{1}{T} \sum_{j=\kappa}^{\kappa+T-1} \boldsymbol{\phi}_i[j|k]\, \boldsymbol{\phi}_i^\top[j|k] < \gamma_\phi^+ \mathbf{I}$$

Control Law Design

The control law used for the DESC will be a general gradient-descent controller (see [63, 59]) given by

$$\boldsymbol{u}_i[\kappa+1|k] = \boldsymbol{u}_i[\kappa|k] - k_g \widehat{\boldsymbol{\theta}}_i^1[\kappa|k] + \boldsymbol{d}_i[\kappa|k] \tag{3.25}$$

where $k_g \in \mathbb{R}_{>0}$ is the proportional gain and $\boldsymbol{d}_i \in \mathbb{R}^m$ is a bounded dither signal that ensures the criteria of Assumption 3.10 are met. As explained by Ljung [92] and Goodwin and Sin [53], the dither signal must be such that $\|\boldsymbol{d}_i[\kappa]\| \leqslant \gamma_d$, where $\gamma_d \in \mathbb{R}_{>0}$, $\forall\ \kappa \geqslant 0$. When coupled with the upper bound on $\widehat{\boldsymbol{\theta}}_i$, this ensures that the discrete input velocity remains bounded, i.e. that $\|\Delta \boldsymbol{u}_i[\kappa]\| \leqslant k_g \gamma_\theta + \gamma_d$. These requirements are easily met by a sinusoidal that can take either of the following forms:

$$\boldsymbol{d}_i[\kappa|k] = D \sin(\boldsymbol{\omega}_i \kappa) \quad \text{or} \quad \boldsymbol{d}_i[\kappa|k] = D \sin(\boldsymbol{\nu}_i)$$

where $D = \gamma_d \in \mathbb{R}_{>0}$ is the amplitude of the dither signal, $\boldsymbol{\nu}_i \in \mathbb{R}^m$ is a vector of i.i.d. random variables with mean 0 and variance σ_d^2, and $\boldsymbol{\omega}_i \in \mathbb{R}^m$ is a vector of dither signal frequencies that meet the selection criteria outlined in Remark 3.11.

Remark 3.11 – Dither Signal Frequency Selection

If one chooses to use non-stochastic dithers, each $\boldsymbol{\omega}_i$ must be chosen with care. Letting $\boldsymbol{\Omega} = \left[\boldsymbol{\omega}_1^\top, \ldots, \boldsymbol{\omega}_p^\top\right]$, where p is the number of agents in the network ($p = 1$ in the centralized case), the frequencies must be such that:

i) $\Omega_i \neq \Omega_j + \Omega_k \ \forall \ i, j, k$

ii) $\Omega_i \neq \beta\Omega_j, \ \forall \ i, j, \ \forall \ \beta \in \mathbb{Q}$

The frequencies must also be in a range that avoids signal aliasing with respect to the optimizer's update period Δt_{D}.

3.2.2 Shifting the Prediction and Control Horizons

Once the DESC completes its task, producing $\mathbf{U}^*[k]$, the first agent's input ($\mathbf{U}_1^*[k]$) is sent to the plant. At the next sampling instant the state is measured to provide the DESC with its new initial condition.

To shift the control horizon each agent communicates its optimized input to the preceding agent, which is used as the initial input condition at the following time step. The terminal agent's initial input is supplied by the TESC, which selects an input vector that guarantees a reduction in the terminal cost. This process for shifting the control horizon (which corresponds to steps 4, 5 and 6 in Figure 3.2) is illustrated by the diagram in Figure 3.5 and formally defined by

$$\mathbf{U}^0[k+1] = \left\{\mathbf{U}_{2:p}^*[k]\,, \boldsymbol{u}_T^*[k]\right\} \tag{3.26}$$

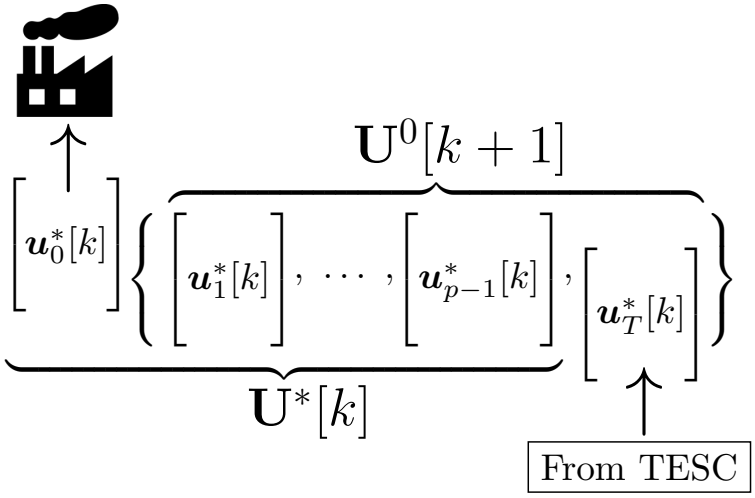


Figure 3.5: Graphical representation of the process by which the control horizon is shifted once the DESC completes its optimization task and how $\mathbf{U}^0[k+1]$ is generated by Equation (3.26). This process corresponds to steps 4, 5 and 6 in Figure 3.2. Factory image source:[Im5]

where $\boldsymbol{u}_T^*$ is the input vector generated by the TESC.

Lastly, the DAC algorithm's integrator variables must be reset at each sampling instant so that proper integral action is applied for the run. Thus, at each sampling instant the consensus algorithm is initialized as per

$$\begin{bmatrix} \widehat{J}[0|k] \\ \boldsymbol{\rho}[0|k] \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}[0|k] \\ \mathbf{0} \end{bmatrix} \tag{3.27}$$

The TESC Optimization Routine

The TESC operates under the same general guidelines as the DESC. It is an independent controller that is not required to communicate with other agents and its only purpose is to minimize its associated cost, y_T. To do so it uses the same RLS parameter estimation routine (3.20) to (3.24) and gradient-descent control law (3.25) as the DESC, but discards the DAC algorithm. The TESC's optimization dynamics are

similar to those of the DESC's agents in that it holds its initial state static at $\boldsymbol{x}_p^*[k]$.

Contrary to the local costs of the DESC's agents, however, the TESC is solely tasked with ensuring that the *terminal* cost decreases. Therefore, the TESC's cost is a function of the terminal state at time $k+p+1$ and does not penalize the input – unless a term is included to enforce input constraints. Excluding a direct input penalty allows the TESC to select inputs of larger magnitude if they are required to reduce the terminal cost. This produces the optimization dynamics and output given by

$$\boldsymbol{x}_T[\kappa+1|k] = \boldsymbol{x}_p^*[k] + \boldsymbol{f}\left(\boldsymbol{x}_p^*[k], \boldsymbol{u}_T[\kappa|k]\right) \tag{3.28}$$

$$y_T[\kappa|k] = \left\|\boldsymbol{x}_T^\top[\kappa+1|k] - \boldsymbol{x}_{sp}\right\|_{\mathbf{P}}^2 + \Upsilon_T\left(\boldsymbol{u}_T[\kappa|k]\right) \tag{3.29}$$

$$= \ell_T\left(\boldsymbol{x}_p^*[k], \boldsymbol{u}_T[\kappa|k]\right) \tag{3.30}$$

where $\Upsilon_T(\boldsymbol{u}_T)$ is an optional input penalty term for enforcing input constraints.

The predicted change in the TESC's cost can be parameterized similarly as with the DESC. There are no contributions from other agents or terms dependent on another agent's input that need to be considered, eliminating the need to consider drift. Hence, the parameterization of the TESC's output dynamics is given by

$$\begin{aligned}
\Delta y_T[\kappa|k] &= y_T[\kappa+1|k] - y_T[\kappa|k] \\
&= \ell_T\left(\boldsymbol{x}_p^*[k], \boldsymbol{u}_T[\kappa+1|k]\right) - \ell_T\left(\boldsymbol{x}_p^*[k], \boldsymbol{u}_T[\kappa|k]\right) \\
&= \int_0^1 \frac{\partial \ell_T\left(\boldsymbol{x}_p^*[k], \delta\boldsymbol{u}_T[\kappa+1|k] + (1-\delta)\,\boldsymbol{u}_T[\kappa+1|k]\right)}{\partial \boldsymbol{u}} d\delta \Delta\boldsymbol{u}_T[\kappa|k] \\
&= \boldsymbol{\theta}_T^\top[\kappa|k]\,\boldsymbol{\phi}_T[\kappa|k]
\end{aligned}$$

where $\boldsymbol{\phi}_T[\kappa|k] = \Delta\boldsymbol{u}_T[\kappa|k]$.

The parameter estimation routine remains unchanged, except the predicted output

error is with respect to the TESC's true cost and not an estimate of the ATC:

$$e_T[\kappa] = \Delta y_T[\kappa] - \Delta \widehat{y}_T[\kappa] \tag{3.31}$$

The TESC's control law is given by (3.32) and is subject to the same assumptions as (3.25) and its dither signal must also meet the conditions listed in Remark 3.11. The TESC runs for a total of κ_T iterations or until a suitably developed termination condition is met.

$$\boldsymbol{u}_T[\kappa+1] = \boldsymbol{u}_T[\kappa] - k_g^T \boldsymbol{\theta}_T[\kappa] + \boldsymbol{d}_T[\kappa] \tag{3.32}$$

3.2.3 A Note on Time-Scale Separation

As a result of the DESC and TESC holding their initial conditions static, they do not necessarily require time-scale separation to function correctly. Recall, however, that the DESC begins its optimization with initial estimates of the ATC equal to that agent's local cost and that, if the parameter estimation routine is to be successful, that the consensus algorithm must converge to the true ATC quickly. Introducing time-scale separation to the DESC's parameter estimation routine and control law allows the consensus algorithm to converge to the true ATC before any significant adjustments are made to the agents' inputs. Thus, even if it is not explicitly required for the parameter estimation routine to converge, the introduction of a time-scale separation can improve the overall performance of the CBMPC architecture.

Incorporating time-scale separation to the estimation routine and control law assumes that the DACA operates at a faster time-scale $\varepsilon \Delta t_{\text{D}}$, where $\varepsilon \in (0, 1)$. The optimizer and control law, however, operate at the slower time scale Δt_{D}. To accommodate this, the rates of change of $\boldsymbol{\Sigma}_i$, $\widehat{\boldsymbol{\theta}}_i$ and $\boldsymbol{u}_i$ must be scaled by ε. Thus, with

time-scale separation (3.22), (3.23) and (3.25) become:

$$\boldsymbol{\Sigma}_i[\kappa+1|k] = a\boldsymbol{\Sigma}_i[\kappa|k] + \varepsilon\boldsymbol{\phi}_i[\kappa|k]\,\boldsymbol{\phi}_i^{\top}[\kappa|k] + \varepsilon\alpha_s\mathbf{I} \tag{3.33}$$

$$\bar{\boldsymbol{\theta}}_i[\kappa+1|k] = \widehat{\boldsymbol{\theta}}_i[\kappa|k] + \frac{\varepsilon\boldsymbol{\Sigma}_i^{-1}[\kappa|k]\,\boldsymbol{\phi}_i[\kappa|k]\,e_i[\kappa|k]}{\alpha + \boldsymbol{\phi}_i^{\top}[\kappa|k]\,\boldsymbol{\Sigma}_i^{-1}[\kappa|k]\,\boldsymbol{\phi}_i[\kappa|k]} \tag{3.34}$$

$$\boldsymbol{u}_i[\kappa+1|k] = \boldsymbol{u}_i[\kappa|k] - \varepsilon k_g\widehat{\boldsymbol{\theta}}_i{}^{1}[\kappa|k] + \varepsilon\boldsymbol{d}_i[\kappa|k] \tag{3.35}$$

where $a = (1 + \varepsilon\,(\alpha - 1))$.

3.3 Constraint Handling

One of the strengths of MPC is the ability to incorporate process constraints in the closed-loop system [16, 129]. When considering CBMPC, the ability to consider constraints without making many changes to the ESC optimizers (if any) would be highly desirable. For this reason, interior point methods (also known as barrier methods) are used to enforce constraints in the overall architecture.

Interior point methods were popularized in the mid 1990s when renewed research developed the techniques further, leading to convergence times that were more competitive than alternative methods at the time (especially with large-scale problems) [156, 117]. To implement them in CBMPC, one adds terms known as *barrier functions* to the cost that penalize violations of the constraints.

It is for this reason that the quadratic term penalizing the input in (3.5) is generalized to the function $\Upsilon\,(\boldsymbol{u}_i)$ in (3.9). A quadratic input penalty dependent on the knowledge of $\boldsymbol{u}_{sp}$ can limit the performance of an MPC-based controller as any inaccuracies in the prediction model or any unknown disturbances affecting the system would

change the steady-state input values corresponding to the desired set-point. Consequently, such penalties would likely hinder the controller's performance. However, these scenarios have a considerably smaller impact on performance if barrier functions are used instead of a quadratic penalties (even if they were parameterized by $\boldsymbol{u}_{sp}$ as well), as their contributions are only significant when the constraint boundaries are approached.

3.3.1 Logarithmic Barrier Functions

Barrier functions are typically logarithmic in nature as logarithms inherently meet the general criteria for barrier function behavior by decaying quickly as the argument moves away from a "barrier" in the form of a vertical asymptote. While more recent approaches to barrier methods require that slackness be considered by the optimization algorithm via the dual problem [117], earlier approaches known as the *primal log-barrier method* are known to be more easily implementable [47, 117]. They do require some additional assumptions regarding the feasibility of the argument ([47]) that are discussed in more detail in Section 3.3.2. The general form of a barrier function used by this method is given by

$$\Phi\left(\boldsymbol{z}\right) = \sum_{j=1}^{m_c} -\mu_j \log\left(\xi_j\left(\boldsymbol{z}\right)\right) \tag{3.36}$$

where $\boldsymbol{z} \in \mathbb{R}^{m_c}$ is the vector of constrained variables, $\boldsymbol{\mu} \in \mathbb{R}^{m_c}_{>0}$ is a vector of weight parameters and the vector field $\boldsymbol{\xi} : \mathbb{R}^{m_c} \to \mathbb{R}^{m_c}_{\geqslant 0}$ is a constraint function satisfying $\boldsymbol{\xi}\left(\boldsymbol{z}\right) \geq \mathbf{0}$. The upper bound on a variable z_j is denoted by b_j^+ while the lower bound is denoted by b_j^-. Before selecting a definition for $\boldsymbol{\xi}$, we consider the following conditions:

i) it is desirable to keep the definition flexible by parameterizing the mapping with

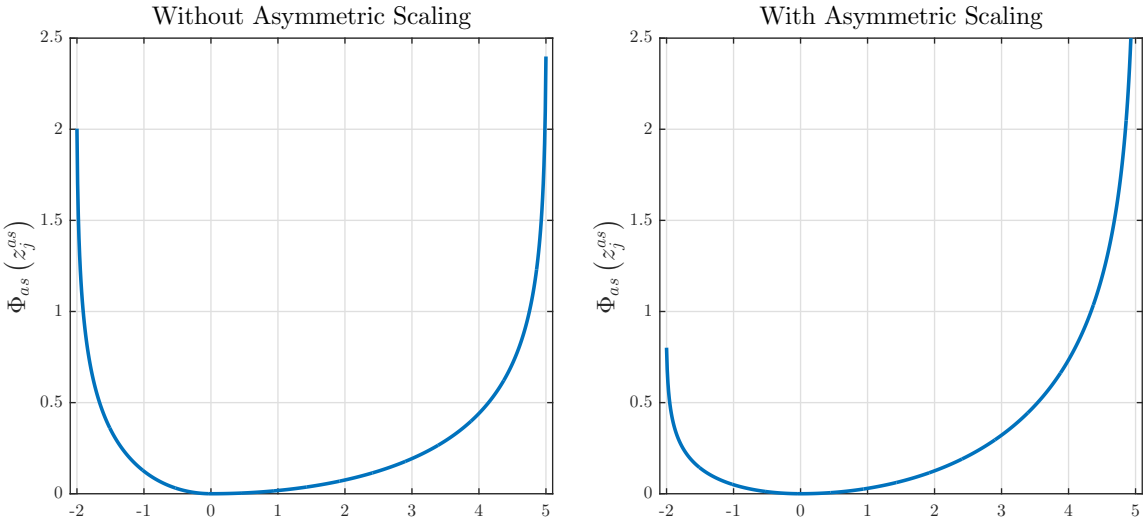


Figure 3.6: The plots above aim to demonstrate why care must be taken when selecting a barrier function for a variable that is asymmetrically bounded about its approximate set point. In this case, $z_{sp,j} = 0$, $b_j^+ = 5$ and $b_j^- = -2$. The plot on the left depicts how a lack asymmetric scaling can lead to a disproportionate response. If one wished to stabilize z_j at 0, a small deviation to the left would produce a much larger kick than an equal deviation to the right. The plot on the right demonstrates how growth of the barrier function is normalized in (3.38).

respect to the constrained variable z_j's approximate set point and desired upper and lower bounds

ii) the upper and lower bounds for a variable z_j may not be symmetrically centered about its approximate set point, $z_{sp,j}$. Care must be taken to ensure that deviations of similar size above or below $z_{sp,j}$ do not generate significantly disproportionate responses (see Figure 3.6 for an example of this phenomenon)

One possible option for $\boldsymbol{\xi}$ that accounts for *symmetrically* bounded (i.e. $\left|b_j^- - z_{sp,j}\right| = \left|b_j^+ - z_{sp,j}\right|$) and *asymmetrically* bounded (i.e. $\left|b_j^- - z_{sp,j}\right| \neq \left|b_j^+ - z_{sp,j}\right|$) variables is provided in Definition 3.12. For a much more flexible definition of mean-centered logarithmic barrier functions, the reader is directed to the work of DeHaan [30]. DeHaan's developments allow a significantly larger family of functions to be used and can provide

greater control of the enforcement of constraints for tailored applications.

Definition 3.12 – A General Barrier Function for Constrained Variables

Let the vector of constrained variables be given by $\boldsymbol{z} \in \mathbb{R}^{m_c}$ *where* m_c *is the total number of constrained variables. Then, the overall barrier function is given by*

$$\begin{aligned} \Phi(\boldsymbol{z}) &= \sum_{j=1}^{m_c} \Phi_j(z_j) \\ &= \sum_{j=1}^{m_c} -\mu_j \log(\xi_j(z_j)) \end{aligned} \tag{3.37}$$

where the map ξ_j *is defined as*

$$\xi_j(z_j) = \begin{cases} \left(1 - \left(\dfrac{z_j - z_{sp,j}}{\left|b_j^+ - z_{sp,j}\right|}\right)^2\right)^{\left|b_j^+ - z_{sp,j}\right| / \left|b_j^- - z_{sp,j}\right|} & \forall\ z_j \geq z_{sp,j} \\ \left(1 - \left(\dfrac{z_j - z_{sp,j}}{\left|b_j^- - z_{sp,j}\right|}\right)^2\right)^{\left|b_j^- - z_{sp,j}\right| / \left|b_j^+ - z_{sp,j}\right|} & \forall\ z_j < z_{sp,j} \end{cases} \tag{3.38}$$

Note that when z_j *is symmetrically bounded,* (3.38) *reduces to*

$$\xi_j^s\left(z_j^s\right) = 1 - \left(\frac{z_j^s - z_{sp,j}^s}{\left|b_j - z_{sp,j}^s\right|}\right)^2 \tag{3.39}$$

While barrier functions are introduced to consider input constraints and discussed as a substitute for input penalties, states can be constrained in the same manner without having to change the definition of the regressor vector $\boldsymbol{\phi}_i$ in the parameterization of the ATC provided by (3.18).

3.3.2 Switch Functions

The drawback to using logarithmic barrier functions is that outside the feasible range for a constrained variable z_j the value of $\Phi_j(z_j)$ is undefined. While more current

barrier methods are equipped to handle such situations [117], when using the primal log-barrier method it is required that the initial condition for the constrained variable is feasible. In the CBMPC framework, this is not guaranteed as a result of the TESC not penalizing the input.

In addition, it has been shown that meeting this requirement guarantees that the trajectory of z_j produced by the classic primal log-barrier method stays within the feasible set [47]. While the proposed barrier function structure is inspired from this method, the optimization algorithm being used is not the same. Here, a gradient descent with constant step size is being used which could cause the algorithm to diverge if the proportional gain k_g in (3.25) is too large.

These scenarios can be addressed by using suitable *switch functions* – second-order Taylor series approximations of the barrier functions made about a point near the boundary of the feasible set. If the trajectory of the constrained variable is accidentally driven outside of its feasible set, then one may evaluate the corresponding switch function instead of the barrier function. This would still result in a sudden, significant increase in the cost's magnitude without rendering the problem infeasible. The general definition for the switch functions, which does not change for symmetrically and asymmetrically bounded variables, is given by

$$\Psi_j(z_j) = \begin{cases} \Phi_j\left(\tilde{b_j}^+\right) + \Phi_j'\left(\tilde{b_j}^+\right)\left(z_j - \tilde{b_j}^+\right) + \Phi_j''\left(\tilde{b_j}^+\right)\left(z_j - \tilde{b_j}^+\right)^2 & \forall\ z_j \geq \tilde{b_j}^+ \\ \Phi_j\left(\tilde{b_j}^-\right) + \Phi_j'\left(\tilde{b_j}^-\right)\left(z_j - \tilde{b_j}^-\right) + \Phi_j''\left(\tilde{b_j}^-\right)\left(z_j - \tilde{b_j}^-\right)^2 & \forall\ z_j \leq \tilde{b_j}^- \end{cases} \tag{3.40}$$

where $\tilde{b_j}^+ = b_j^+ - \epsilon$ and $\tilde{b_j}^- = b_j^- + \epsilon$ are points very close to the feasibility boundary for z_j, with $\epsilon << 1$ being a strictly positive constant to be assigned. Thus, for the provided definitions of Φ and Ψ, the overall term to be added to the local costs defined

by (3.9) to enforce input constraints, $\Upsilon(\boldsymbol{u}_i)$, is defined as

$$\Upsilon(\boldsymbol{u}_i) = \sum_{j=1}^{m} \Upsilon_j(u_{i,j}) \tag{3.41}$$

where

$$\Upsilon_j(u_{i,j}) = \begin{cases} \Phi_j(u_{i,j}) & \forall\ u_{i,j} \in [\tilde{b_j}^-, \tilde{b_j}^+] \\ \Psi_j(u_{i,j}) & \forall\ u_{i,j} \notin [\tilde{b_j}^-, \tilde{b_j}^+] \end{cases} \tag{3.42}$$

3.4 Summary

To reiterate, the CBMPC architecture integrates a distributed ESC into the MPC algorithm as its optimizer. To do so, each point on the prediction horizon is treated as an agent; rather than adopting the conventional approach of using agents in a MAS to represent a plant's various subsystems, agents of the CBMPC are used to represent the same system at different moments in time.

At each sampling instant, the state is measured and the local cost bias is determined. Both the state and the bias are communicated to each of the agents and a distributed ESC routine is initiated. To drive the inputs to their respective optimums, each agent begins with an initial guess which it uses to predict the next state and compute the local cost. Local cost information is then shared between agents so that they may update their estimates of the system's average total cost via a discrete-time, dynamic average consensus algorithm. The agents track the change in their local estimates of the average total cost to update their estimates of its gradient with respect to the change in that agent's local input using a recursive least-squares algorithm. The estimated gradient is then used in a gradient descent to adjust the input. Each agent then communicates

their input to succeeding agents so that they may all correctly compute their respective predictive states at the next iteration. Once this cycle completes a given number of iterations, the routine returns the optimized inputs.

The first optimized input is then sent to the plant and implemented. Before moving to the next time step, the control horizon is shifted to provide the agents of the distributed ESC-based optimizer with updated initial conditions for the following sample time. This shift is performed by having each agent communicate their optimal input to the previous agent. However, this process leaves the terminal agent empty-handed, and so a terminal ESC is defined to provide the last agent with its new initial condition. Once an input that reduces the predicted terminal cost at the next time step is selected, it is passed to the terminal agent. At this point, the state is re-measured and the cycle is repeated.

{ Chapter 4 }

Analyzing the Performance of CBMPC Implementations

With the general machinery for CBMPC now outlined, the controller's performance can be discussed. As a complete proof for the overall architecture's stability is beyond the scope of this work, the main objective of this chapter is to test the performance of CBMPC. The impact of design variables, tuning parameters, and of a system's properties on the overall performance will be explored and examined in three case studies.

In the first investigation CBMPC is used to stabilize a first-order unstable MIMO system, both with and without input constraints. The simulation results are then compared to those produced by a conventional MPC controller[1] using sequential quadratic programming (SQP) to solve for the collection of optimal inputs at each time step. SQP was chosen as the optimization algorithm as these methods are widely accepted as some of the most accurate and efficient constrained optimization algorithms for nonlinear programming and they are widely applied in the design and implementation of MPC [136, 52, 25]. Lastly, a brief discussion of the impact of communication network structure on the DESC's performance is included.

[1]The MPC Toolbox provided by the software was *not* used to run the simulations with conventional MPC controllers to ensure that there were not any subroutines affecting the selection of inputs and ensure that the integrity of the comparisons wasn't compromised.

In the second case study, CBMPC is implemented to control the concentration of a reactant in a non-isothermal, jacket-cooled CSTR by manipulating the flow rate of coolant passing through the jacket. This problem consists of stabilizing a third-order unstable SISO system. The results of the simulation will be used to demonstrate that CBMPC can be used effectively to control systems with unstable dynamics of higher order without having to introduce any measure of state feedback to the control law and without having to make any changes to the architecture itself.

Finally, the third investigation is purposed with exploring the applicability of CBMPC to economic MPC scenarios. A first simulation demonstrates that CBMPC can effectively stabilize the system at a steady-state that maximizes a profit-based objective function. Then, a second simulation is included to show how including a time-averaged constraint on the input by making a slight modification to the objective function allows CBMPC to stabilize the system to a periodic orbit. This allows the user to avoid cases in which the optimum steady-state corresponds to the system's continuous operation at maximum control effort.

4.1 Example 1: A First-Order MIMO System

In this section, consider the discrete-time nonlinear system with dynamics given by

$$\boldsymbol{x}[k+1] = \boldsymbol{x}[k] + \Delta t \boldsymbol{f}\left(\boldsymbol{x}[k], \boldsymbol{u}[k]\right) \tag{4.1}$$

where Δt is the sampling interval and the vector field $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is defined as

$$f_1\left(\boldsymbol{x}, \boldsymbol{u}\right) = -x_1 + \frac{x_3 x_5}{5} + u_1\left(2 - \operatorname{atan}\left(\frac{x_5}{10}\right)\right) + \frac{\sin\left(u_6\right)}{2} \tag{4.2}$$

$$f_2\left(\boldsymbol{x}, \boldsymbol{u}\right) = -5x_2 + \frac{x_1 x_3}{1 + \sqrt{x_4^2 + x_5^2}} + u_2\left(5 + \cos\left(x_2\right)\right) \tag{4.3}$$

$$f_3(\boldsymbol{x},\boldsymbol{u}) = -\frac{x_3^3}{2} + \ln\left(0.01 + x_2^2\right) + x_7 + 6u_3\left(5 - \frac{3\sin(x_6)}{2}\right) \tag{4.4}$$

$$f_4(\boldsymbol{x},\boldsymbol{u}) = -\left(2 - 4\,\mathrm{atan}\left(\frac{x_4}{10}\right)\right)^2 + x_2\cos(x_1) + u_4 + \frac{u_2u_5}{50} \tag{4.5}$$

$$f_5(\boldsymbol{x},\boldsymbol{u}) = \frac{5x_5}{2} - \sin(x_3)\left(x_1 + \ln\left(\frac{x_7^2 + 0.01}{x_2^2 + 1}\right)\right) + 2u_5 \tag{4.6}$$

$$f_6(\boldsymbol{x},\boldsymbol{u}) = -x_6^2 + x_3e^{2-x_1^2} + u_6\left(1 - \frac{\cos^2(x_2)}{2 + \sin(x_4)}\right) \tag{4.7}$$

$$f_7(\boldsymbol{x},\boldsymbol{u}) = x_7^2 - \sin(x_1x_2) + \mathrm{atan}\Big(\frac{x_3x_4}{5}\Big) - \cos\Big(\frac{x_5x_6}{3}\Big) + u_7\Big(1 + \frac{1}{2}\,\mathrm{atan}^2\Big(\frac{u_3}{10}\Big)\Big) \tag{4.8}$$

For the simulations the system was initialized at $\boldsymbol{x}[0] = \mathbf{0}$ and stabilized to $\boldsymbol{x}_{sp} = [4, -1, 6, -2.5, -4, 1, 2.5]^\top$ with the corresponding steady-state inputs being $\boldsymbol{u}_{sp} \cong [3.50, -1.66, 4.70, 8.04, -5.72, 1.26, -3.66]^\top$. At each iteration, inputs are chosen by minimizing the conventional quadratic MPC cost function given by (3.5). The elements of the cost's diagonal weighting matrices are provided in Table 4.1 and the terminal cost's weighting matrix was selected by solving the discrete-time algebraic Riccati equation, as described in Section 3.1.1. For each case, the DESC's communication network was structured in the manner depicted in Figure 3.4.

Table 4.1: List of the MPC cost function design parameters used in the simulations to validate the performance of the CBMPC architecture.

Cost Function Weighting Matrices	
Q	$\mathrm{diag}\{[50, 50, 50, 250, 50, 50, 250]\}$
R	$\mathrm{diag}\{[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]\}$
S	$\mathrm{diag}\{[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]\}$

Table 4.2: A summary of the settings and tuning parameters used in the simulation of the system given by (4.1)-(4.8) to validate the performance of the CBMPC architecture with unconstrained inputs. See Figure 4.1 for the simulation results.

General Settings		DESC Settings		TESC Settings	
Parameter	Value	Parameter	Value	Parameter	Value
Δt	$0.02\ [s]$	$k_{g,\mathrm{D}}$	0.001	$k_{g,\mathrm{T}}$	0.005
p	22	D_{D}	0.02	D_{T}	0.05
Δt_{D}	$5 \cdot 10^{-4}$	α_{D}	0.001	α_{T}	0.01
κ_{D}	2000	$\alpha_{s,\mathrm{D}}$	0.01	$\alpha_{s,\mathrm{T}}$	0.01
κ_{T}	750	ε_{D}	0.5	ε_{T}	1
β_{P}	2000	$\sigma_{0,\mathrm{D}}$	$1 \cdot 10^5$	$\sigma_{0,\mathrm{T}}$	$1 \cdot 10^5$
β_{I}	181.8	$\hat{\boldsymbol{\theta}}_{i,\mathrm{D}}[0\|k]$	$-10 \cdot \mathbf{1}$	$\hat{\boldsymbol{\theta}}_{i,\mathrm{T}}[0\|k]$	$-10 \cdot \mathbf{1}$
$\sigma^2_{d,\mathrm{D}}, \sigma^2_{d,\mathrm{T}}$	1.75	$\gamma_{\theta_{\mathrm{D}}}$	75	$\gamma_{\theta_{\mathrm{T}}}$	50

4.1.1 Unconstrained Inputs

In this first scenario, the performance of CBMPC is compared to that of a conventional MPC when no input constraints are enforced. For the conventional MPC, the SQP algorithm (implemented via MATLAB's built-in `fmincon` function[2]) was used to solve the optimization problem at each iteration. For the CBMPC controller, a summary of the tuning parameters chosen for the DESC, for the TESC, and the remaining design variables is provided in Table 4.2. The results of each simulation are compared in Figure 4.1.

For this unconstrained case, each MPC framework clearly produces trajectories that are nearly identical. In contrast to the conventional controller, CBMPC produces inputs that are a bit smaller in magnitude early in the simulation. These more moderate inputs produced a smaller initial ATC value and lead to significantly less overshoot in

[2]While `fmincon` is typically used for constrained optimization, it was also used here to remain consistent when comparing the architectures in both the constrained and unconstrained scenarios.

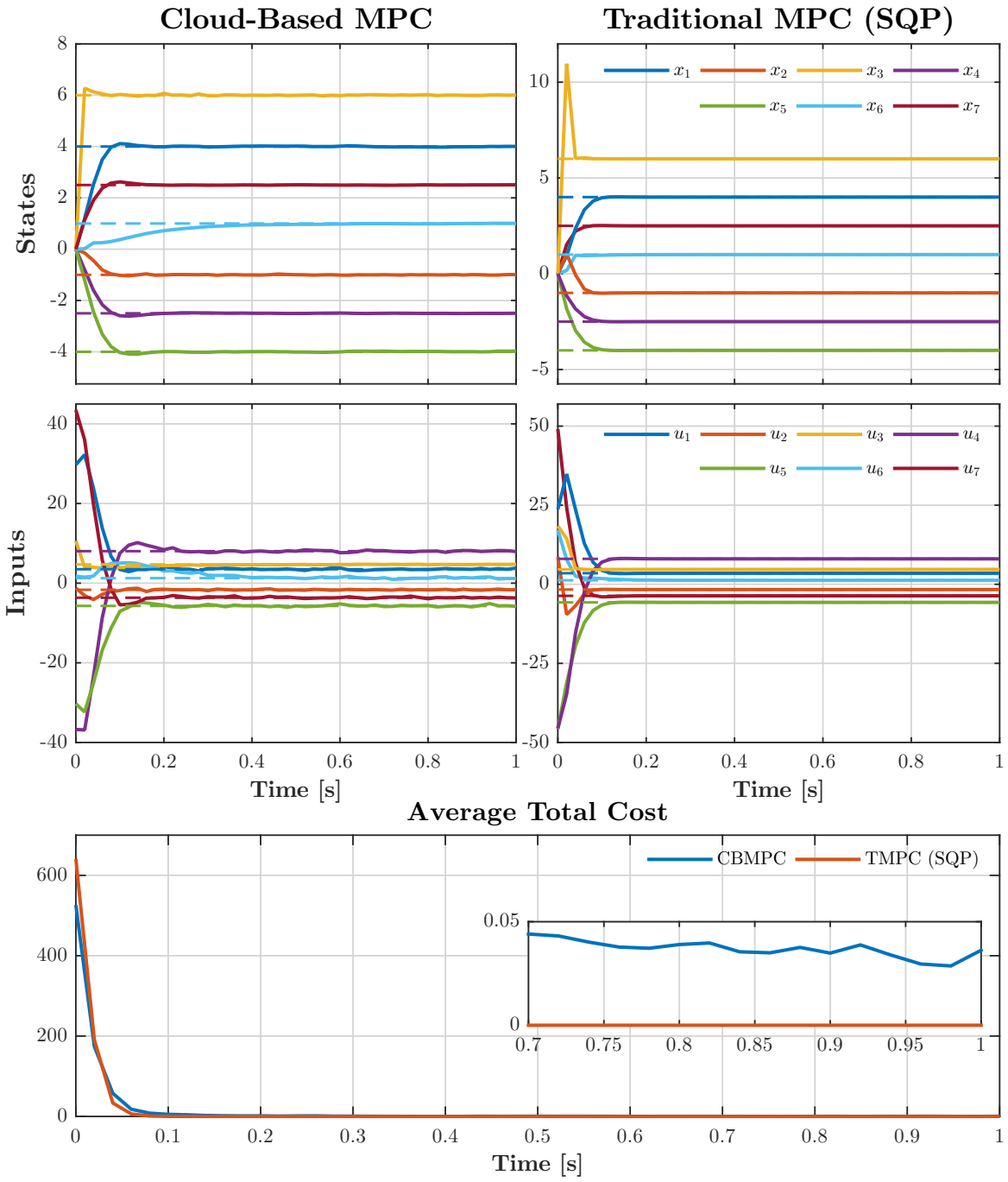


Figure 4.1: Performance comparison of CBMPC and traditional MPC using the SQP optimization algorithm *without* enforcing constraints on the inputs. The desired state set-points and corresponding steady-state inputs are designated by the dashed lines.

the third state's trajectory ($x_3[k]$). This occurrence is most closely related to the fact that the DESC is run for a finite, predefined number of iterations (κ_{D}), which means that the DESC can complete its run before the parameter estimates have had the chance to converge. Thus, the early input trajectories predicted by the CBMPC may not have been those that minimized the total cost to the maximum possible degree. While this implies that the first few control moves are not optimal, however, the implemented inputs have the desirable effect of almost eliminating the overshoot produced by the conventional controller in the third state's trajectory.

The aforementioned phenomenon highlights the forgiving nature of the CBMPC framework and demonstrates that full parameter convergence is not required for the control effort to be successful. Since the extremum-seeking control laws are based on a gradient descent, it is known that even if the inputs produced by the routines are sub-optimal, they still steer the system in a direction that reduces the cost. Thus, with each passing step the time required for full convergence decreases and the true minimizer for the cost at that sampling interval is reached and the closed-loop system operates as expected.

The simulation results demonstrate that the CBMPC yields a *practical* stability property in the closed-loop. This means that the controller doesn't drive the cost completely to 0 and that it can only guarantee its convergence to a neighborhood of the origin. This is due to the presence of dither signals in the input, which are required to provide the persistence of excitation necessary for closed-loop stability and convergence of the parameter estimation routine.

The size of the limit set to which the closed-loop system converges is proportional to the amplitude of the dither signal [1, 2], making smaller amplitudes preferable.

Table 4.3: List of the barrier function weights and input bounds used in the simulations to validate the performance of the CBMPC architecture.

Barrier Function Design Parameters	
Lower Bounds ($\boldsymbol{b}^-$)	$[-10, -10, -10, -15, -20, -10, -15]$
Upper Bounds ($\boldsymbol{b}^+$)	[20, 10, 20, 20, 10, 15, 20]
Barrier Weights ($\boldsymbol{\mu}$)	[1, 1, 1, 1, 1, 1, 1]

However, caution must be taken when reducing this value as the magnitude of the extremum's region of attraction (ROA) is determined by the ratio of the dither's amplitude to the controller's proportional gain (D/k_g) [58, 64]. As a result, if the dither's amplitude is reduced to diminish the size of the limit set and the gain is scaled down by the same amount the ROA's size will be unaffected. On the other hand, a reduction of the gain can slow down convergence and increase computation times. Tuning the ESC must therefore be done with careful consideration of the process dynamics' nature and the system sampling frequency.

4.1.2 Constrained Inputs

To validate the performance of the developed CBMPC architecture when input constraints are enforced, the simulation of the system given by (4.1)-(4.8) was repeated and compared to the results produced by a conventional MPC controller. The upper and lower bounds for the inputs are summarized in Table 4.3. For the CBMPC, constraints were enforced using the barrier function defined in (3.38) with the weights listed in Table 4.3.

Due to the inclusion of a barrier function in the local costs for the DESC's agents, the tuning parameters for the DESC were slightly modified from the values chosen

Table 4.4: A summary of the settings and tuning parameters used in the simulation of the system given by (4.1)-(4.8) to validate the performance of the CBMPC architecture with constrained inputs. See Figure 4.2 for the simulation results.

General Settings		DESC Settings		TESC Settings	
Parameter	Value	Parameter	Value	Parameter	Value
Δt	$0.02\ [s]$	$k_{g,\mathrm{D}}$	0.001	$k_{g,\mathrm{T}}$	0.005
p	22	D_{D}	0.01	D_{T}	0.05
Δt_{D}	$5 \cdot 10^{-4}$	α_{D}	0.001	α_{T}	0.01
κ_{D}	2000	$\alpha_{s,\mathrm{D}}$	0.01	$\alpha_{s,\mathrm{T}}$	0.01
κ_{T}	1000	ε_{D}	0.5	ε_{T}	1
β_{P}	2000	$\sigma_{0,\mathrm{D}}$	$1 \cdot 10^5$	$\sigma_{0,\mathrm{T}}$	$1 \cdot 10^5$
β_{I}	181.8	$\hat{\boldsymbol{\theta}}_{i,\mathrm{D}}[0\|k]$	$-5 \cdot \mathbf{1}$	$\hat{\boldsymbol{\theta}}_{i,\mathrm{T}}[0\|k]$	$-10 \cdot \mathbf{1}$
$\sigma^2_{d,\mathrm{D}}, \sigma^2_{d,\mathrm{T}}$	1.75	$\gamma_{\theta_{\mathrm{D}}}$	75	$\gamma_{\theta_{\mathrm{T}}}$	50

for the unconstrained scenario. A summary of the tuning parameters chosen for the DESC, for the TESC, and the remaining design variables used for the CBMPC in the simulation is provided in Table 4.4. A comparison of the results produced by both the CBMPC and conventional MPC is shown in Figure 4.2.

As with the unconstrained case, the CBMPC produced trajectories nearly identical to those generated by the conventional controller. With the enforcement of input constraints, inputs u_1, u_4, u_5, and u_7 remain saturated for the first 0.075 seconds of the simulation. If the function's weights are chosen carefully and the switch functions given by (3.40) are used to accommodate any input value that could potentially exceed their boundaries, the controller is capable of holding the inputs at their maximal values without having to saturate the signals directly. This highlights the effectiveness of barrier functions to enforce input constraints.

Due to the extra input penalty, the gradient with respect to the inputs increases more rapidly as they move away from their steady-state values. This increases the

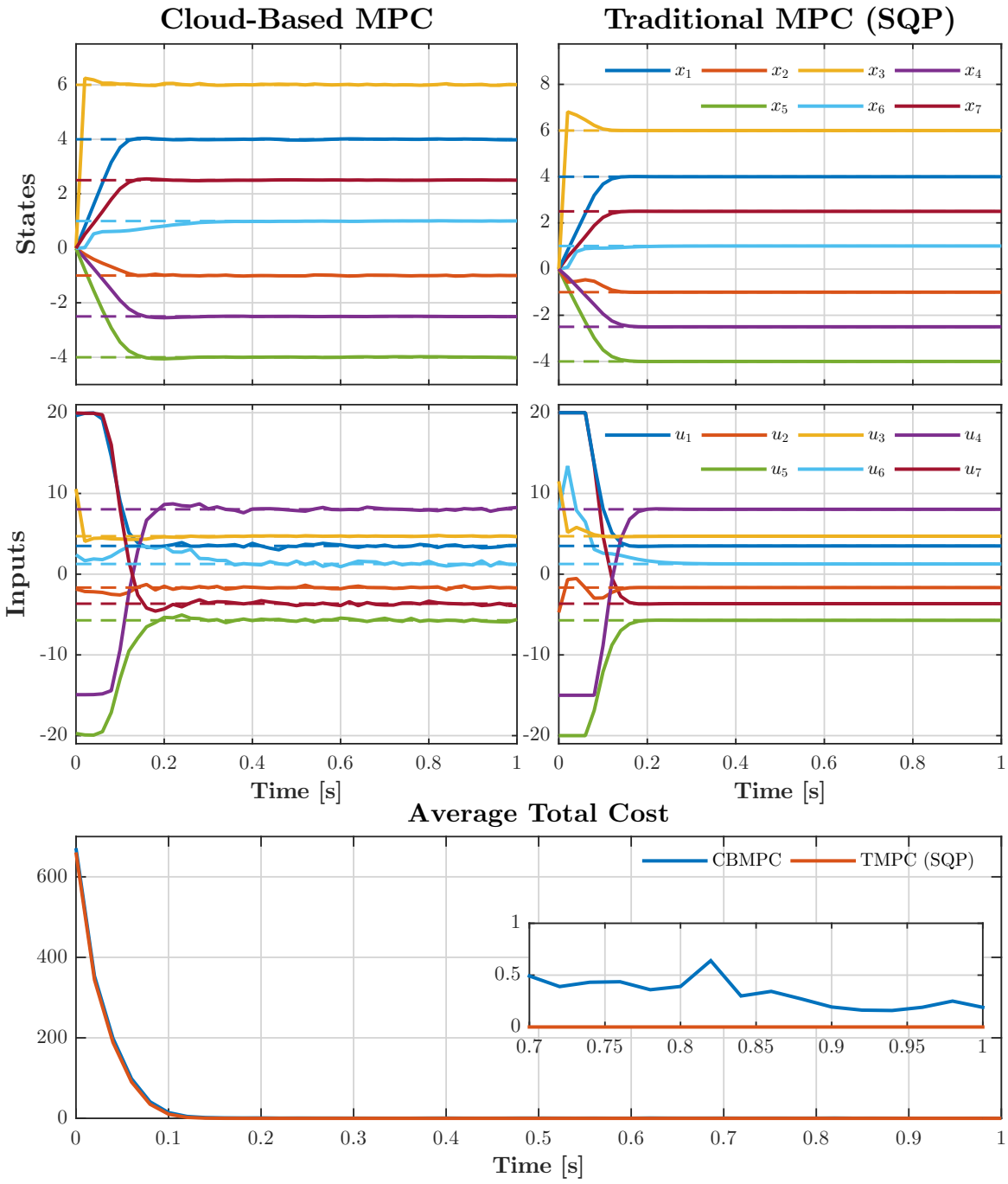


Figure 4.2: Performance comparison of CBMPC and traditional MPC using the SQP optimization algorithm *with* constraints on the inputs. The desired state set-points and corresponding steady-state inputs are designated by the dashed lines.

driving force of the controller and explains why the average total cost is not minimized to the degree as it was in the unconstrained scenario.

4.1.3 Computational Performance

While each control framework can produce near-identical results, the primary advantage in using CBMPC over conventional MPC controllers lies in the total computation time required by the solvers to determine the collection of optimal inputs at each sampling instant. The simulations were performed in MathWorks' MATLAB v.R2015a on a 2015 Dell Optiplex 9020 equipped with an Intel® Core™ i7-4790 (3.60 GHz) processor and 16 Gb installed RAM running Windows 8.1 Enterprise (64-bit). The computation times required by each controller to generate their optimal inputs at each time step are compared in Figure 4.3 for both the unconstrained and constrained scenarios. Note that for the CBMPC, the plot is of the average computation time per agent (as the framework allows the agents to run their routines in parallel).

As the extremum-seeking controllers are run for a predefined number of iterations at each sampling instant the total computation time taken by the CBMPC hardly varied from step to step. For the conventional controller, significantly more time was required earlier on in the simulation to arrive at a solution. However, even as time progressed, the conventional controller requires more time to perform the computations. These results clearly indicate that CBMPC is more than capable of performing to the same high degree of performance as conventional MPCs without requiring nearly as much time. This could allow CBMPC to be implemented for systems with faster dynamics.

Due to its distributed nature, CBMPC can perform the calculations for each input along the horizon in parallel. This also means that CBMPC can arbitrarily increase

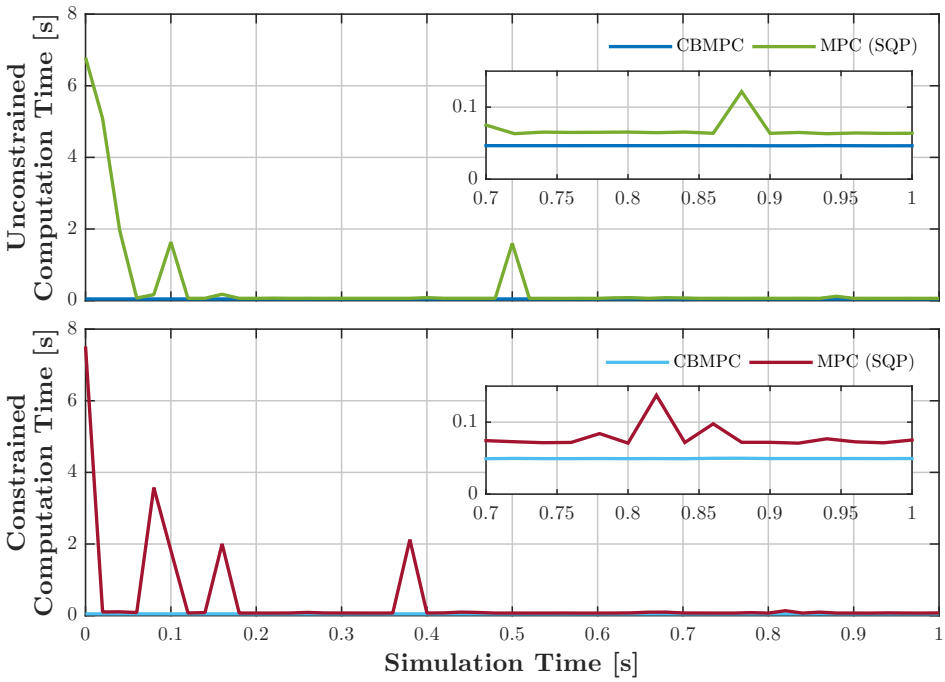


Figure 4.3: A comparison of the computation times required by the CBMPC and conventional MPC controllers to determine the collection of optimal inputs along the control horizon at each iteration.

the size of its prediction horizon without having *any* impact on total computation time. This offers another huge advantage over conventional controllers due to the stability requirements for nonlinear MPC. As systems increase in complexity, more attention must be paid to the selection of a suitable terminal cost matrix and terminal region. These are simply tools that allow the user to guarantee stability when considering a finite-horizon optimization problem by providing routes to approximate the infinite-horizon problem. With CBMPC, one could simply extend the prediction horizon to a point where a well tuned, specialized terminal cost penalty or suitable terminal constraint is no longer required (see [55]) – all without increase computation time and compromising its ability to control the system effectively.

4.2 Example 2: A Higher-Order SISO System

In this second case study, the CBMPC is used to stabilize a system with unstable, third-order dynamics. The system describes the dynamics of a non-isothermal, exothermic continuously-stirred tank reactor (CSTR). The reactor is cooled by a jacket and the rate of heat removal is controlled via the flow rate of coolant through the jacket. A diagram of the system can be found in Figure 4.4.

The CSTR is treated as a lumped-parameter system and is assumed to maintain a constant liquid-level. Assuming that density and heat capacity are constant, that the reaction has first-order kinetics, and that the reaction rate constant is described by an

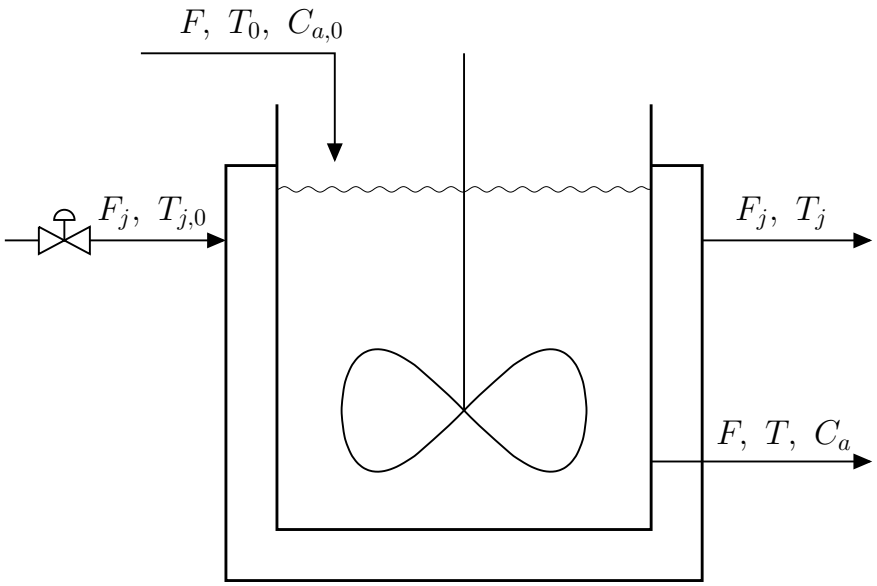


Figure 4.4: Process flow diagram for the simulated jacket-cooled CSTR.

Table 4.5: Summary of the parameters used in the simulation of a non-isothermal CSTR with a first-order reaction.

Parameter	Value	Units	Parameter	Value	Units
F	7.5	$m^3 \cdot min^{-1}$	ΔH_r	$-20 \cdot 10^3$	$kJ \cdot kmol^{-1}$
V	50	m^3	ρ	1000	$kg \cdot m^{-3}$
$C_{a,0}$	1	$kmol \cdot m^{-3}$	c_p	0.239	$kJ \cdot (kg \cdot K)^{-1}$
k_0	$8.2 \cdot 10^{10}$	min^{-1}	UA	7000	$kJ \cdot (min \cdot K)^{-1}$
$-E_a/R$	-8750	K	V_j	10	m^3
T_0	350	K	$T_{j,0}$	283	K

Arrhenius expression, we derive the following dynamic model:

$$\frac{dC_a}{dt} = \frac{F}{V}(C_{a,0} - C_a) - k_0 e^{-E_a/RT} C_a \tag{4.9}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) - \frac{\Delta H_r k_0 e^{-E_a/RT} C_a}{\rho c_p} + \frac{UA}{\rho c_p V}(T_j - T) \tag{4.10}$$

$$\frac{dT_j}{dt} = \frac{F_j}{V_j}(T_{j,0} - T_j) - \frac{UA}{\rho c_p V_j}(T_j - T) \tag{4.11}$$

where concentration of the reactant, C_a, is to be controlled by manipulating the flow rate of coolant through the jacket, F_j. A summary of the CSTR's model parameters is shown in Table 4.5.

The control objective is to stabilize the system to $C_{a,sp} = 0.20\ kmol/m^3$, $T_{sp} = 341.3\ K$ and $T_{j,sp} = 321.9\ K$. The system was initialized at $C_a[0] = 0.3\ kmol/m^3$, $T[0] = 334.2\ K$ and $T_j[0] = 315.2\ K$. The cost function weights and barrier function

Table 4.6: List of the MPC cost function weights and barrier function design parameters used in the simulation of a CSTR with third-order dynamics under CBMPC with input constraints.

Cost Function Weights		Barrier Function Parameters	
$\mathbf{Q}$	$\text{diag}\{[50, 30, 5]\}$	Lower Bound ($\boldsymbol{b}^-$)	3.6
$\mathbf{R}$	0.5	Upper Bound ($\boldsymbol{b}^+$)	80
$\mathbf{S}$	0.01	Barrier Weight ($\boldsymbol{\mu}$)	0.01

Table 4.7: A summary of the settings and tuning parameters used in the simulation of a CSTR with third-order dynamics under CBMPC with input constraints. See Figure 4.5 for the simulation results.

General Settings		DESC Settings		TESC Settings	
Parameter	Value	Parameter	Value	Parameter	Value
Δt	$0.075\ [s]$	$k_{g,\mathrm{D}}$	0.001	$k_{g,\mathrm{T}}$	0.005
p	26	D_{D}	0.025	D_{T}	0.005
Δt_{D}	$3.33 \cdot 10^{-4}$	α_{D}	$5 \cdot 10^{-4}$	α_{T}	0.01
κ_{D}	3000	$\alpha_{s,\mathrm{D}}$	0.01	$\alpha_{s,\mathrm{T}}$	0.01
κ_{T}	2000	ε_{D}	0.5	ε_{T}	0.5
β_{P}	3000	$\sigma_{0,\mathrm{D}}$	$1 \cdot 10^{6}$	$\sigma_{0,\mathrm{T}}$	$1 \cdot 10^{5}$
β_{I}	230.8	$\hat{\boldsymbol{\theta}}_{i,\mathrm{D}}[0\|k]$	$-75 \cdot \mathbf{1}$	$\hat{\boldsymbol{\theta}}_{i,\mathrm{T}}[0\|k]$	$-10 \cdot \mathbf{1}$
$\sigma^2_{d,\mathrm{D}}, \sigma^2_{d,\mathrm{T}}$	1.75	$\gamma_{\theta_{\mathrm{D}}}$	100	$\gamma_{\theta_{\mathrm{T}}}$	75

parameters used to upper and lower bound the coolant flow rate are given in Table 4.6 while the CBMPC tuning parameters are summarized in Table 4.7. The structure of the communication network used for the consensus algorithm is depicted in Figure 3.4. The simulation results, shown in Figure 4.5, indicate that CBMPC is capable of effectively stabilizing unstable systems with higher-order dynamics.

4.2.1 Including State Feedback in the Control Law

The above example used open-loop predictions to select the optimal inputs. It is generally recognized that MPC performs much better when closed-loop predictions are considered. One mechanism that is often used involves adding a linear stabilizing feedback in the implemented control effort. While no simulation example is provided for this case, including state feedback in the control law can also be done with CBMPC without making any changes to the framework – although it is not necessary for the

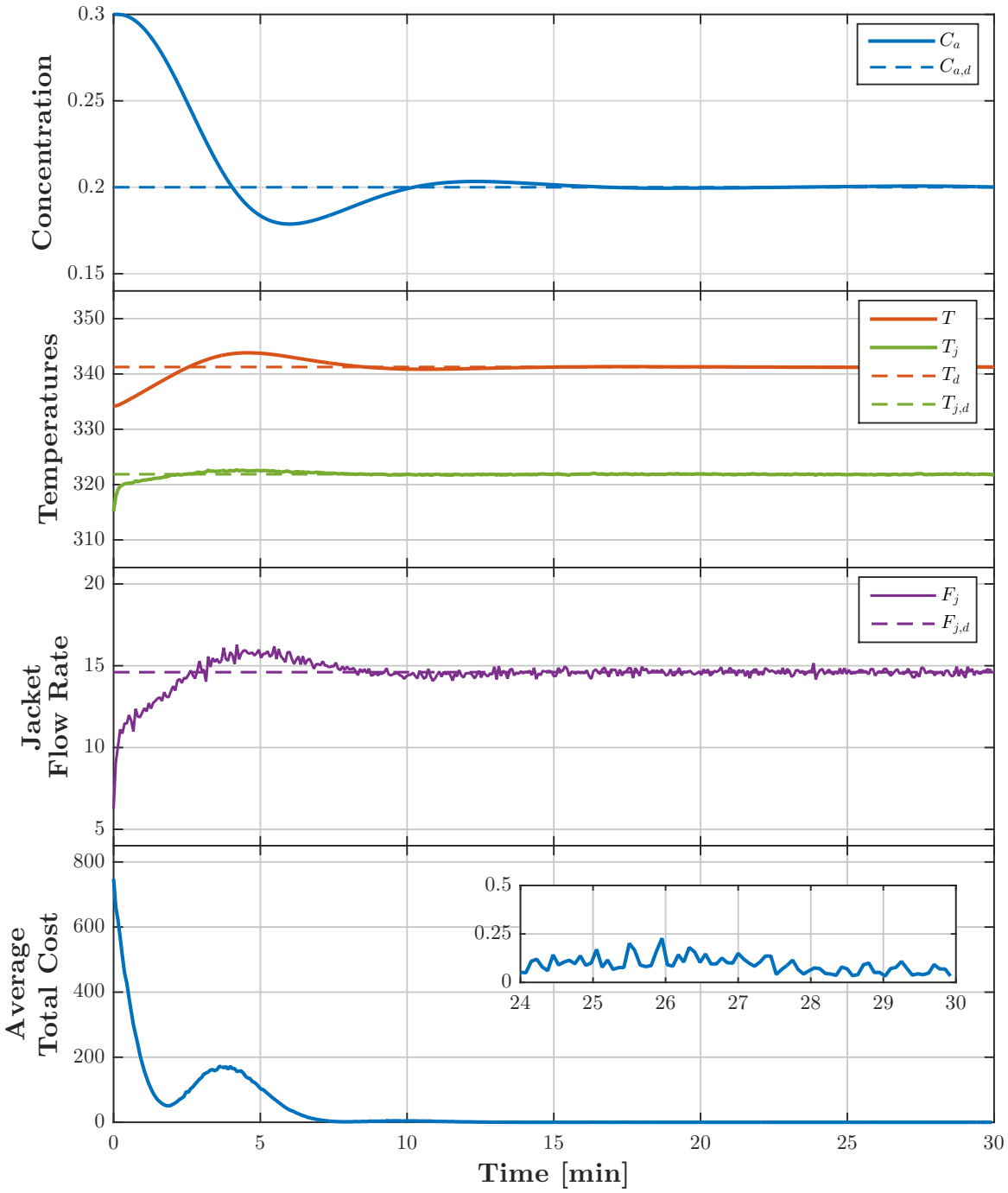


Figure 4.5: Results for the simulation of a jacket-cooled CSTR carrying out an exothermic reaction with first-order kinetics operating at constant volume. The reactant concentration, reactor temperature and jacket temperature are controlled using the CBMPC architecture by manipulating the flow rate of coolant through the jacket.

controller to be effective. Nonetheless, if incorporating state feedback to the control law was desired, one would simply need to define an auxiliary input $\boldsymbol{v}$, given here by

$$v[k] = -\mathbf{K}\left(\boldsymbol{x}[k] - \boldsymbol{x}_{sp}\right) + u[k] \tag{4.12}$$

where $\mathbf{K}$ is the gain matrix for proportional action on the state feedback and $\boldsymbol{u}$ is the input selected by the extremum-seeking controllers.

Once the extremum-seeking inputs are calculated as per the gradient descent (3.25) and the auxiliary input is determined, one can set this value equal to the controlled variable's rate of change and solve for the corresponding manipulated variable. For instance, for this case study taking the third derivative of the concentration's rate of change provides an expression for the system's dynamics with respect to the manipulated variable F_j. Setting the third derivative equal to the auxiliary input would then allow the user to solve for the coolant flow rate corresponding to that input, which would then be sent to the plant.

4.3 Example 3: Extensions to Economic MPC

MPC provides a systematic approach to select inputs to maximize efficiency and improve performance. This is achieved by optimizing metrics that aim to minimize the close-loop system's tracking error and control effort. In many practical situations, it may be advantageous to compute inputs that optimize an economics-based objective function. This premise provides the motivation for the design of economic-MPC systems. In this section, we propose an extension of CBMPC for the solution of large-scale economic MPC problems.

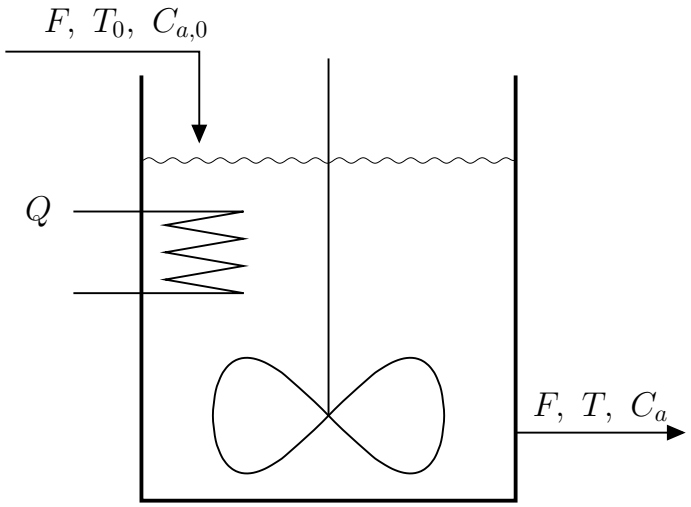


Figure 4.6: Diagram of the CSTR to be controlled under the economic MPC formulation.

For this case study, an example from the work of Ellis et al. [43] is considered. The system is similar to the exothermic non-isothermal CSTR in the previous example. In this study the reaction is taken to have second-order kinetics and can be cooled or heated by an internal coil. A schematic representation of the system is shown in Figure 4.6 and the model of the system's dynamics is given by

$$\frac{dC_a}{dt} = \frac{F}{V}(C_{a,0} - C_a) - k_0 e^{-E_a/RT} {C_a}^2 \tag{4.13}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_0 - T) - \frac{\Delta H_r k_0 e^{-E_a/RT} {C_a}^2}{\rho c_p} + \frac{Q}{\rho c_p V} \tag{4.14}$$

where the feedstock reactant concentration $C_{a,0}$ and heat flow Q are the manipulated variables. The model parameters are listed in Table 4.8.

The control objective is to maximize the production of the reaction product subject to the input constraints $C_{a,0} \in [0.5, 7.5]\ \ kmol/m^3$ and $Q \in [-20, 20]\ \ MJ/hr$. As this is a maximization problem, the expression for the produced product is negative in the cost. The barrier function (3.38) is used to enforce the input constraints, producing

Table 4.8: Summary of the parameters used in the simulation of a non-isothermal CSTR with a second-order reaction under economic CBMPC.

Parameter	Value	Units	Parameter	Value	Units
F	5	$m^3 \cdot hr^{-1}$	ΔH_r	$-1.16 \cdot 10^4$	$kJ \cdot kmol^{-1}$
V	1	m^3	ρ	1000	$kg \cdot m^{-3}$
T_0	300	K	c_p	0.239	$kJ \cdot (kg \cdot K)^{-1}$
k_0	$8.46 \cdot 10^6$	$m^3 \cdot (kmol \cdot hr)^{-1}$	$-E_a/R$	-6014	K

the cost function given by

$$y_i[k] = \Phi\left(\boldsymbol{u}_i[k]\right) - \beta k_0 e^{-E_a/RT} {C_a}^2 \Delta t \tag{4.15}$$

where β is a production weight parameter to scale the impact of production on the cost. For this scenario, the barrier and production weights were set to $\mu_{C_{a,0}} = 0.1$, $\mu_Q = 0.075$, and $\beta = 10$. The remaining design variables and tuning parameters are listed in Table 4.9. The simulation was initialized at $\boldsymbol{x}\left[0\right] = \left[2, 425\right]^\top$, and the results are shown in Figure 4.7.

The results demonstrate that CBMPC is effective for stabilizing the system at an

Table 4.9: A summary of the settings and tuning parameters used in the simulation of a CSTR under CBMPC with an economics-based objective function. See Figure 4.7 for the simulation results.

General Settings		DESC Settings		TESC Settings	
Parameter	Value	Parameter	Value	Parameter	Value
Δt	$0.01\,[hr]$	$k_{g,\mathrm{D}}$	$5 \cdot 10^{-4}$	$k_{g,\mathrm{T}}$	0.002
p	20	D_D	0.001	D_T	0.001
Δt_D	$2.5 \cdot 10^{-3}$	α_D	0.1	α_T	0.1
κ_D	400	$\alpha_{s,\mathrm{D}}$	0.015	$\alpha_{s,\mathrm{T}}$	0.015
κ_T	100	ε_D	0.75	ε_T	0.75
β_P	400	$\sigma_{0,\mathrm{D}}$	$1 \cdot 10^3$	$\sigma_{0,\mathrm{T}}$	$1 \cdot 10^3$
β_I	80	$\hat{\boldsymbol{\theta}}_{i,\mathrm{D}}[0\|k]$	$-1 \cdot \mathbf{1}$	$\hat{\boldsymbol{\theta}}_{i,\mathrm{T}}[0\|k]$	$-1 \cdot \mathbf{1}$
$\sigma^2_{d,\mathrm{D}}, \sigma^2_{d,\mathrm{T}}$	1.75	$\gamma_{\theta_\mathrm{D}}$	10	$\gamma_{\theta_\mathrm{T}}$	10

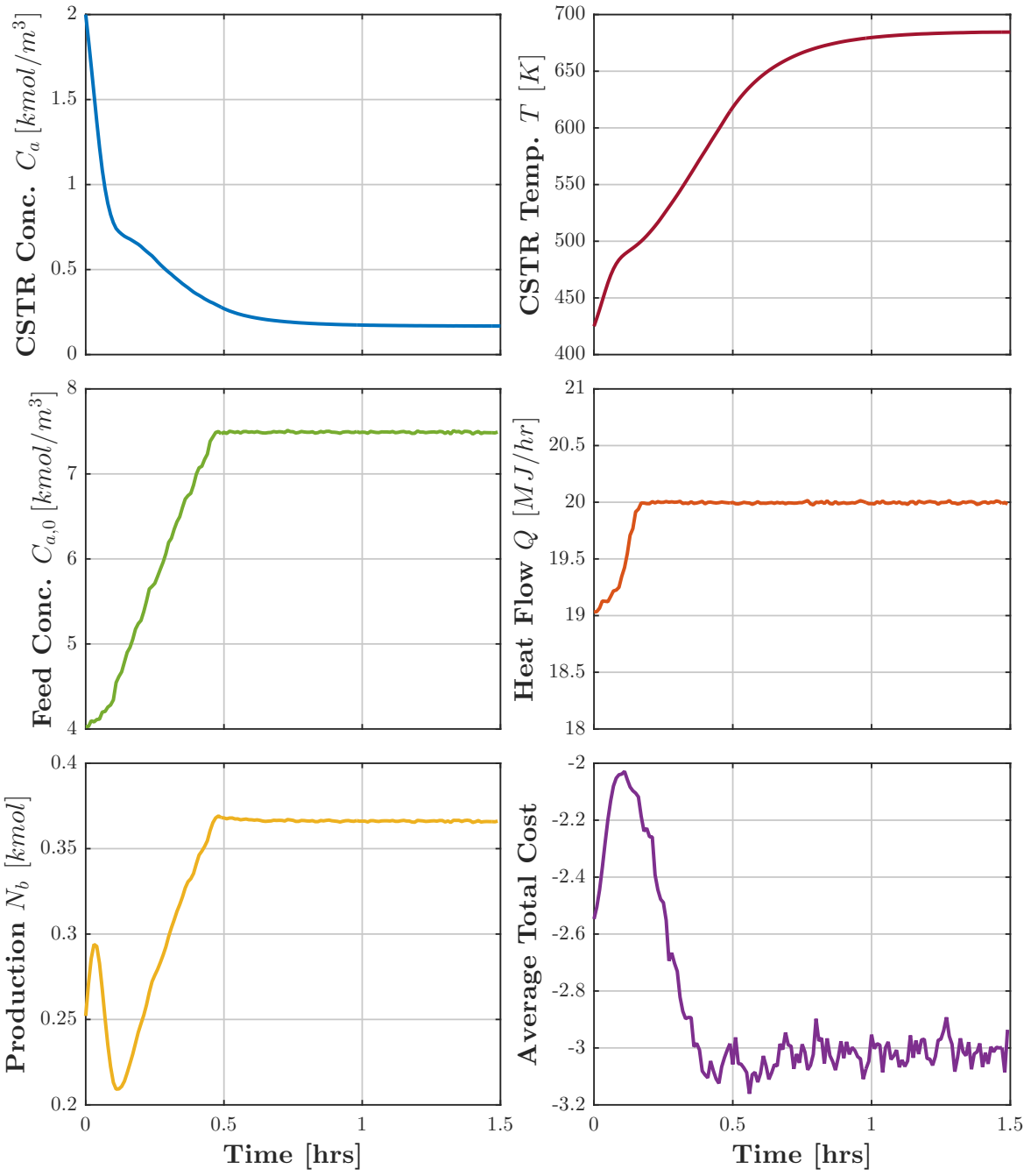


Figure 4.7: Results for the simulation of a non-isothermal CSTR carrying out a reaction with second order kinetics under CBMPC with an economic-based cost function.

operating condition that maximizes the stepwise unit production of the desired product. Since the inputs are stabilized to points very close to their upper bounds, the steady-state contribution of the barrier function to the cost is significantly larger than in the previous examples. This is what causes the large degree of variability in the average total cost once the system has reached steady-state. While operating the system at maximum input effort may maximize the production of the desired product, operating under such conditions may not be an appropriate choice in practice (due to increased operating or reactant material costs, for instance).

It has been argued in the literature that the most profitable solutions are achieved by plants operating around a stable, periodic orbit (see Section 2.2.3). To avoid continuously operating at maximum input effort and achieve the desired dynamic behavior an additional, time-average constraint is placed on the input. This is done by defining an *operating period* of length M sampling instants and selecting a desired average input value for the period. In this example, the maximum reaction rate can be achieved when the system is operated at the greatest possible temperature. As a result, the heat flow is fixed at its upper bound in this scenario, as is done in [43]. Following the approach of Angeli et al. [7], a time-average constraint is placed on the feedstock concentration:

$$\sum_{i=j}^{j+p} \boldsymbol{u}_i[k] + \sum_{i=0}^{j-1} \boldsymbol{u}_0^*[k-j+i] \geq pC_{a,0}{}^{-} + (j+p)\,\overline{C}_{a,0} \tag{4.16}$$

$$\sum_{i=j}^{j+p} \boldsymbol{u}_i[k] + \sum_{i=0}^{j-1} \boldsymbol{u}_0^*[k-j+i] \leq pC_{a,0}{}^{+} + (j+p)\,\overline{C}_{a,0} \tag{4.17}$$

where p is the prediction horizon length, $\overline{C}_{a,0}$ is the average feedstock concentration over the operating period M, $C_{a,0}{}^{-}$ and $C_{a,0}{}^{+}$ are the lower and upper bounds on the input, respectively, and j is an index that is reset at the beginning of each operating

period (i.e., $j = 0$ when k is a multiple of M). Note that the constraint given by (4.16) and (4.17) does not guarantee that the time-average input is achieved right away. However, it has been shown to drive the time-average input to its desired value asymptotically [7]. Rather than using the barrier function (3.38), the mean-centered barrier function described by DeHaan [30] is used in this example, given by

$$\overline{\Phi}\left(\boldsymbol{z}\right) = \sum_{i=1}^{m_z} \overline{\Phi}_i\left(z_i\right) - \overline{\Phi}_i\left(0\right) - \nabla\overline{\Phi}_i\left(0\right) z_i \tag{4.18}$$

$$\overline{\Phi}_i\left(z_i\right) = -\mu_i\Big(\ln\left(z_i + b_i^-\right) + \ln\left(-z_i + b_i^+\right)\Big) \tag{4.19}$$

where $\boldsymbol{z} \in \mathbb{R}^{n_z}$ is a vector of constrained variables, n_z is the total number of constrained variables, and b_i^+ and b_i^- are the upper and lower bounds on z_i. Since the barrier function enforcing the time-average input constraint is scalar, its contribution will be distributed between all of the agents' local costs (similarly to the bias y_0 in the previous examples). Therefore, the local costs to be minimized by the DESC in this example are defined as

$$y_i[k] = \frac{\overline{\Phi}(\mathbf{U}[k])}{p} + \overline{\Phi}_u(u_i[k]) - \beta k_0 e^{-E_a/RT} {C_a}^2 \Delta t \tag{4.20}$$

where $\overline{\Phi}(\mathbf{U})$ is the barrier function enforcing the time-average input constraint, $\overline{\Phi}_u$ enforces the upper and lower bounds on the feedstock concentration, and β is the same product production weight as in (4.15).

To help stabilize the system to the desired orbit, an equality constraint is placed on the terminal state. This is set to $\boldsymbol{x}_p[k] = \boldsymbol{x}_s^*$, where $\boldsymbol{x}_s^*$ is the optimal steady state corresponding to the time-average input. Note that this method is not viable when the system is subjected to disturbances, noise, or model inaccuracies. In such cases, a suitable terminal region must also be defined for the closed loop to be stable [5, 6].

To enforce the equality constraint, the terminal agent's local cost is augmented by a quadratic penalty to obtain

$$y_p[k] = \frac{\overline{\Phi}(\mathbf{U}[k])}{p} + \overline{\Phi}_u(u_i[k]) - \beta k_0 e^{-E_a/RT} {C_a}^2 \Delta t + (\boldsymbol{x}_p[k] - \boldsymbol{x}_s^*)^\top \mathbf{P} (\boldsymbol{x}_p[k] - \boldsymbol{x}_s^*)$$

where $\mathbf{P} \in \mathcal{M}_{n\times n}(\mathbb{R})$ is a symmetrical, positive definite weighting matrix.

For the simulation, the desired time-average feedstock concentration is chosen to be 4.0 $kmol/m^3$ for an operating period of $M = 100$ time steps and the heat flow fixed at 20 MJ/hr, as in the example found in [43]. The optimal steady-states corresponding to the desired inputs are $\boldsymbol{x}_s^* = [0.719, 481.4]^\top$ in $kmol/m^3$ and K, respectively, and the selected terminal penalty matrix is given by

$$\mathbf{P} = \begin{bmatrix} 25 & 0.5 \\ 0.5 & 0.02 \end{bmatrix}$$

The production weight is set to $\beta = 200$, the barrier function parameters used in the simulation are listed in Table 4.10, and the remaining design variables and tuning parameters are given in Table 4.11. The communication network used in the simulation was structured such that each agent communicates with its neighbors and with every second agent afterwards. The corresponding Laplacian matrix for a network with an

Table 4.10: List of the MPC cost function weights and barrier function design parameters used in the simulation of a CSTR under economic CBMPC with a time-average input constraint.

Barrier Function Parameters			
${C_{a,0}}^{min}$	0.5	Input Barrier Weight	0.5
${C_{a,0}}^{max}$	7.5	Time-Average Barrier Weight	0.5

even number of agents is given by

$$\mathbf{L} = \begin{bmatrix} p/2 & -1 & 0 & -1 & \cdots & -1 & 0 \\ -1 & p/2 & -1 & 0 & \ddots & & -1 \\ 0 & -1 & p/2 & \ddots & \ddots & \ddots & \vdots \\ -1 & 0 & \ddots & \ddots & \ddots & 0 & -1 \\ \vdots & \ddots & \ddots & \ddots & p/2 & -1 & 0 \\ -1 & & \ddots & 0 & -1 & p/2 & -1 \\ 0 & -1 & \cdots & -1 & 0 & -1 & p/2 \end{bmatrix}$$

The system is simulated over 25 hours of operation. Since the system converged to a stable orbit quickly, the results for the last 5 hours of simulation time are shown in Figure 4.8 to show the closed-loop behavior more clearly.

The simulation demonstrates that CBMPC can successfully stabilize the system to a stable orbit. However, in this case it does not perform as well as the conventional controller implemented in the work of Ellis et al. [43]. The literature suggests that

Table 4.11: A summary of the settings and tuning parameters used in the simulation of a CSTR under CBMPC with an economics-based objective function. See Figure 4.7 for the simulation results.

General Settings		DESC Settings		TESC Settings	
Parameter	Value	Parameter	Value	Parameter	Value
Δt	$0.01\,[hr]$	$k_{g,\mathrm{D}}$	0.01	$k_{g,\mathrm{T}}$	0.0025
p	20	D_{D}	0.1	D_{T}	0.0025
Δt_{D}	$2.5{\cdot}10^{-4}$	α_{D}	0.01	α_{T}	0.01
κ_{D}	4000	$\alpha_{s,\mathrm{D}}$	0.01	$\alpha_{s,\mathrm{T}}$	$5 \cdot 10^{-3}$
κ_{T}	1000	ε_{D}	0.01	ε_{T}	0.5
β_{P}	400	$\sigma_{0,\mathrm{D}}$	$1 \cdot 10^4$	$\sigma_{0,\mathrm{T}}$	$1 \cdot 10^5$
β_{I}	40	$\hat{\boldsymbol{\theta}}_{i,\mathrm{D}}[0\|k]$	$-10 \cdot \mathbf{1}$	$\hat{\boldsymbol{\theta}}_{i,\mathrm{T}}[0\|k]$	$-10 \cdot \mathbf{1}$
$\sigma^2_{d,\mathrm{D}}, \sigma^2_{d,\mathrm{T}}$	2	$\gamma_{\theta_{\mathrm{D}}}$	100	$\gamma_{\theta_{\mathrm{T}}}$	50

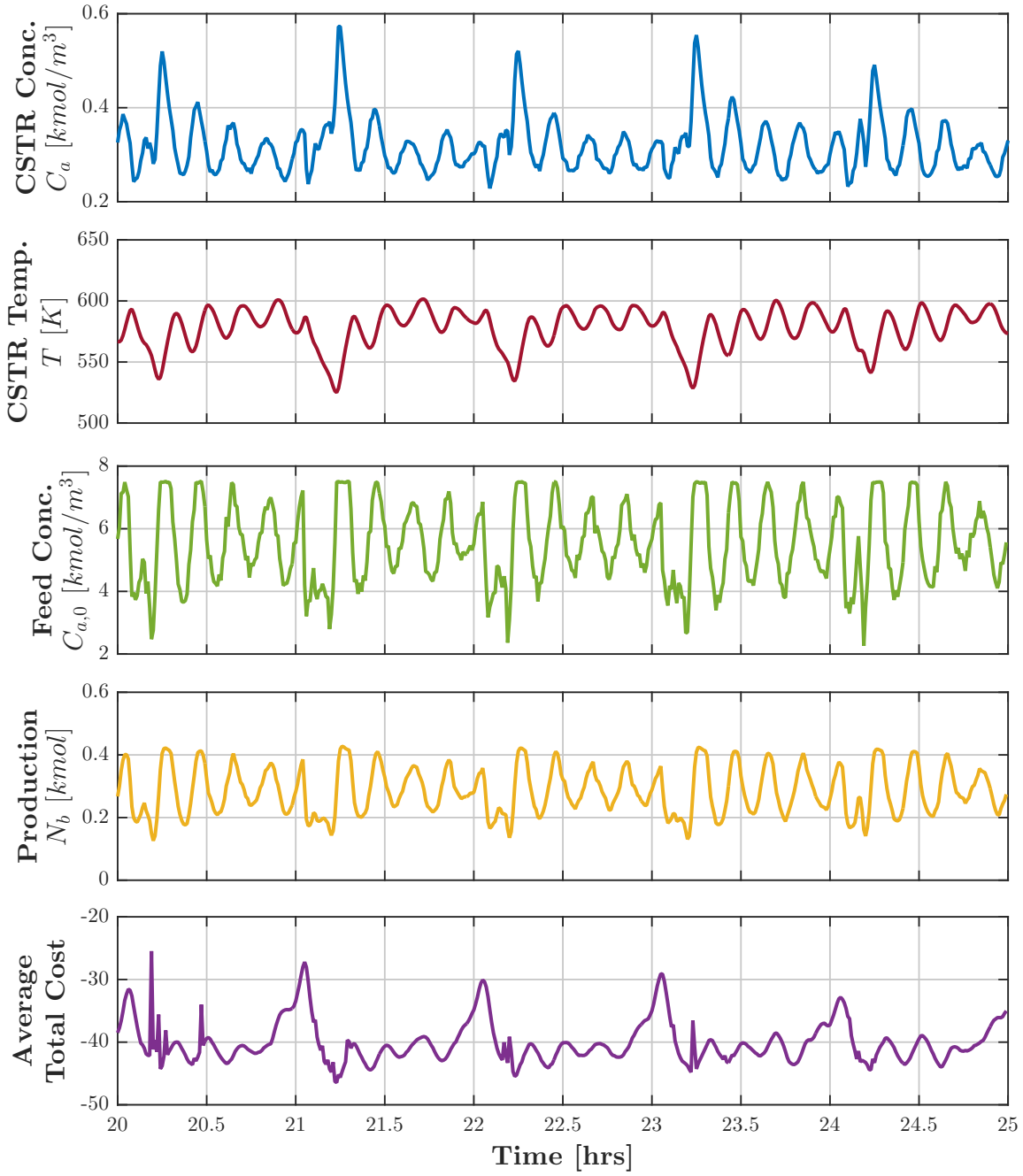


Figure 4.8: Results for the simulation of a non-isothermal CSTR carrying out a reaction with second order kinetics under CBMPC with an economic-based cost function with the enforcement of a time-average input constraint. The heat flow plot is not included as the value was held fixed at 20 MJ/hr.

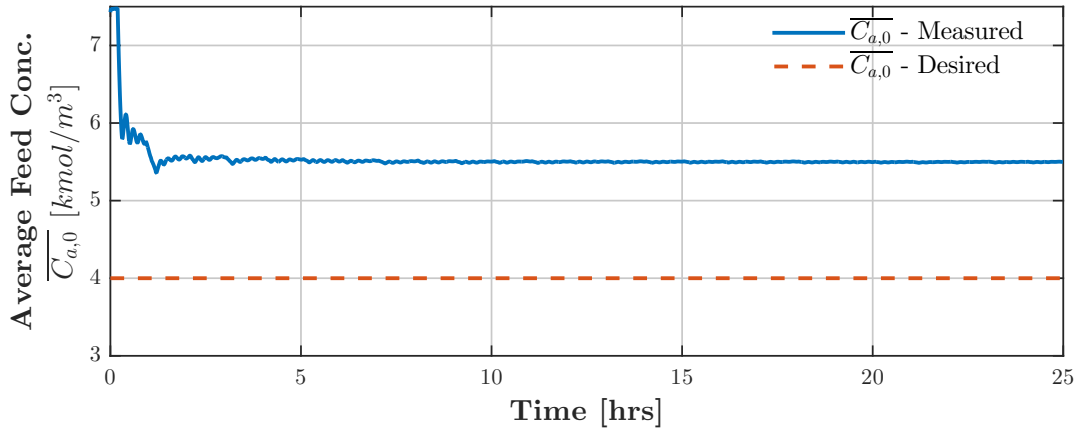


Figure 4.9: A plot of the average feedstock concentration, $\overline{C}_{a,0}$, over the course of the whole simulation. The solid line designates the actual average feed concentration achieved by the CBMPC while the desired average is indicated by the dotted line.

the optimal orbit sees the feedstock concentration alternate between its maximum and minimum allowed values every 10 time steps. While the fluctuations in feedstock concentration produced by the CBMPC occur at the desired frequency, the controller does not drive their minimum value as low as the literature suggests is optimal. This is most likely due to the fact that the desired average feedstock concentration ($\overline{C}_{a,0} = 4\ kmol/m^3$) is not achieved, as is shown in Figure 4.9.

This phenomenon could suggest that the weight of the barrier function enforcing the time-average input constraint is too low relative to the penalty on the terminal state. However, it is found that increasing the weight of this term reduced the frequency at which the input attempted to alternate between higher and lower values, reducing the controller's overall performance. This indicates that including a heavily-weighted barrier function to enforce the time-average input constraint prevents the penalty on the terminal state from being fully minimized. The effected violations of the terminal constraint – which was enforced as a strict equality in the literature example – are most likely the primary factor that led to the lower bound on feedstock concentration

being higher than expected.

Increasing the weight on the terminal state's penalty also reduced the closed-loop performance of the system under CBMPC. Increasing the terminal cost's weight caused the CBMPC to discourage the selection of inputs closer to their upper and lower bounds. This caused the system to become more focused on meeting the terminal constraint at the cost of reduced product production. In the end, it was found that increasing the degree of time-scale separation had the most impact on improving the resolution of CBMPC's input trajectory.

In summary, the results demonstrate that CBMPC can effectively stabilize the closed-loop to a periodic orbit and, while it does not achieve the same performance as a conventional controller in this example, the architecture shows promise in its application to economic process optimization. By investigating the manner in which the local costs are constructed, the relationships between the various constraints placed on the optimization problem, and the impact of the extremum-seeking controllers' tuning parameters on the optimization trajectories further, CBMPC could bring all the benefits of distributed control to economic MPC formulations and achieve closed-loop performance comparable to conventional economic MPC.

{ Chapter 5 }

Conclusion

5.1 Summary

In this thesis, a fully distributed, cloud-based MPC architecture is developed. It used a distributed optimization approach for the design of MPC controllers for nonlinear systems in discrete-time. By integrating a dynamic average consensus algorithm and time-varying extremum-seeking control with MPC, the inputs along prediction horizon can be assigned to agents that share local cost information to collaboratively minimize the total cost at each iteration. This allows the optimization problem to be solved more quickly and for the prediction horizon to be extended to arbitrarily large lengths without having a major impact on the total computation time.

In Chapter 3, the mathematical foundations for the architecture are presented. First, general definitions that describe the criteria for viable objective functions are provided. An input-output model describing the dynamics of the average total cost as a function of an agent's local input is derived to form the basis of a gradient descent control law. Using this model, the machinery for the distributed ESC tasked with computing the optimal inputs is outlined. The recommended dynamic average consensus algorithm allows agents to estimate the average total cost based on local cost information shared over a suitable communication network. Then, a parameter estimation

routine that allows agents to estimate the unknown parameters in the derived input-output model. The estimates are used by agents in a gradient-descent-based control law, permitting them to converge on inputs that practically minimize the total cost at that iteration. The remainder of the chapter outlines a mathematical basis for taking process constraints into account when selecting inputs, built on logarithmic barrier functions.

Chapter 4 presents three case studies in which the performance of the proposed CBMPC architecture is investigated. The results of the first study demonstrate that the architecture converges to the same solutions produced by its centralized MPC counterpart, both in constrained and unconstrained scenarios. In the second case study, it is shown that CBMPC can be implemented to control systems with dynamics of higher order effectively. In the third investigation, CBMPC is implemented in an economic MPC scheme and the results indicate that the architecture is also capable of driving a system to a stable periodic orbit.

The CBMPC architecture proposed in this thesis is shown to perform as well as its centralized counterpart for a very broad class of nonlinear systems. By using a distributed optimization approach – and without requiring a convoluted decomposition of the optimization problem – the framework is shown to be computationally efficient and very easily scalable. While the examples examined in the case studies only considered systems distributed by their prediction horizons, this approach could also be used for a subsystem-based distribution of a plant's processes, as shown by the communication network depicted in Figure 5.1. The developed technique could be used for a wide variety of real-world applications and assist with improving operational efficiency in many industrial processes by providing a highly flexible foundation for cooperative DMPC.

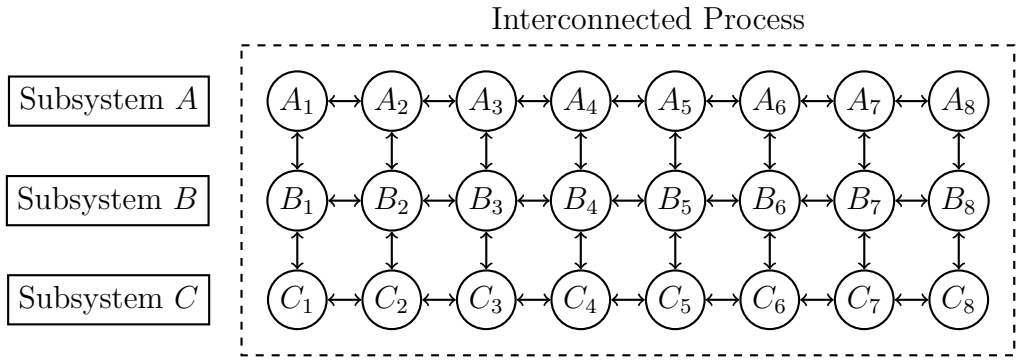


Figure 5.1: An example demonstrating how agents could be used to represent both the points along the prediction horizon in addition to different subsystems.

5.2 Recommendations for Future Work

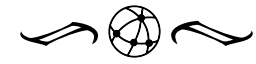
While the initial results provided in Chapter 4 demonstrate the promise shown by CBMPC as a machinery for distributed MPC, there are many aspects of the architecture that remain to be explored and developed. The most immediate next step would be to complete a proof of the construct's stability and convergence to the solution produced by the corresponding centralized controller.

In each case study, the DESC computes a predefined number of iterations at each sampling instant. As a result, the time taken by the DESC to make its computations remained constant throughout the simulation. To improve computational efficiency further, an exit condition for the algorithm that detects convergence to the optimum could de developed. Coupling this with an investigation of time-varying tuning parameters, such as decaying dithers and adjusted initial conditions for the parameter estimates, could also improve the overall architecture's performance and reduce the likelihood of optimization trajectory divergence.

Another avenue for research would be concerned with accounting for asynchronous

measurements and investigating the impact of local controller failures and communication delays. This would be especially useful if CBMPC is used to distribute the optimization problem between agents representing different subsystems in addition to the points along the prediction horizon, as shown in Figure 5.1. If CBMPC is installed on a cloud server, developing protocols to handle lost or time-delayed communications could also improve the reliability of the approach. A detailed investigation of these issues would also bring out one of CBMPC's greatest strengths: a resilience to external, intelligent attack. Due to the architecture's distributed nature, a malicious party wishing to disrupt a facility's operations by causing controller malfunctions would be faced with a significantly more difficult task. Rather than only needing to bring down one, centralized controller, the malicious party would need to decipher the roles of each agent and the communications passing between them – considerably complicating the endeavorss.

Additionally, investigating CBMPC's robustness and its ability to accommodate process and measurement noise or erroneous models would allow the framework to be used effectively in a much broader range of applications. Since the architecture's distributed nature is embedded in the controller's optimization algorithm, it could be possible to formulate the problem to incorporate adaptive MPC techniques to the construct.



Bibliography

[1] Adetola, V. and Guay, M. "Parameter Convergence in Adaptive Extremum-Seeking Control". In: *Automatica* 43.1 (Jan. 2007), pp. 105–110. DOI: 10.1016/j.automatica.2006.07.021.

[2] Adetola, V. and Guay, M. "Finite-Time Parameter Estimation in Adaptive Control of Nonlinear Systems". In: *IEEE Transactions on Automatic Control* 53.3 (Apr. 2008), pp. 807–811. DOI: 10.1109/TAC.2008.919568.

[3] Alanqar, A., Ellis, M., and Christofides, P. "Economic Model Predictive Control of Nonlinear Process Systems Using Multiple Empirical Models". In: *Proceedings of the 2015 American Control Conference*. Chicago, IL, July 2015, pp. 4953–4958. DOI: 10.1109/ACC.2015.7172110.

[4] Allgöwer, F., Badgwell, T., Qin, J., Rawlings, J., and Wright, S. "Nonlinear Predictive Control and Moving Horizon Estimation - An Introductory Overview". In: *Advances in Control: Highlights of ECC '99*. Ed. by Frank, P. M. Springer, 1999, pp. 391–449. DOI: 10.1007/978-1-4471-0853-5_19.

[5] Amrit, R., Rawlings, J. B., and Angeli, D. "Economic Optimization Using Model Predictive Control With a Terminal Cost". In: *Annual Reviews in Control* 35.2 (Dec. 2011), pp. 178–186. DOI: 10.1016/j.arcontrol.2011.10.011.

[6] Amrit, R., Rawlings, J. B., and Biegler, L. T. "Optimizing Process Economic Performance Using Model Predictive Control". In: *Computers & Chemical Engineering* 58 (Nov. 2013), pp. 334–343. DOI: 10.1016/j.compchemeng.2013.07.015.

[7] Angeli, D., Amrit, R., and Rawlings, J. B. "On Average Performance and Stability of Economic Model Predictive Control". In: *IEEE Transactions on Automatic Control* 57.7 (July 2012), pp. 1615–1626. DOI: 10.1109/TAC.2011.2179349.

[8] Ariyur, K. and Krstić, M. "Analysis and Design of Multivariable Extremum Seeking". In: *Proceedings of the 2002 American Control Conference*. IEEE. Anchorage, AK, May 2002, pp. 2903–2908. DOI: 10.1109/ACC.2002.1025231.

[9] Ariyur, K. and Krstić, M. "Slope-Seeking: A Generalization of Extremum Seeking". In: *International Journal of Adaptive Control and Signal Processing* 18.1 (2004), pp. 1–22. DOI: 10.1002/acs.777.

[10] Barshad, Y. and Gulari, E. "A Dynamic Study of CO Oxidation on Supported Platinum". In: *AIChE Journal* 31.4 (Apr. 1985), pp. 649–658. DOI: 10.1002/aic.690310415.

[11] Başar, T. and Olsder, G. J. *Dynamic Noncooperative Game Theory*. 2nd Ed. Vol. 23. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1999. ISBN: 978-0-89871-429-6. DOI: 10.1137/1.9781611971132.

[12] Bitmead, R. R., Gevers, M., and Wertz, V. *Adaptive Optimal Control: The Thinking Man's GPC*. Englewood Cliffs, NJ: Prentice-Hall, 1990. ISBN: 978-0-13013-277-2. DOI: 10.1002/acs.4480050507.

[13] Blackman, P. "Extremum-Seeking Regulators". In: Westcott, J. *An Exposition of Adaptive Control. Proceedings of a Symposium Held at the Imperial College of Science and Technology, London*. New York: The Macmillan Company, 1962. ISBN: 978-0-08009-659-9.

[14] Bondy, J. and Murty, U. *Graph Theory*. Vol. 244. Graduate Texts in Mathematics. Springer, 2008. ISBN: 978-1-84628-969-9. DOI: 10.1007/978-1-84628-970-5.

[15] Budman, H., Kzyonsek, M., and Silveston, P. "Control of a Nonadiabatic Packed Bed Reactor Under Periodic Flow Reversal". In: *Canadian Journal of Chemical Engineering* 74.5 (Oct. 1996), pp. 751–759. DOI: 10.1002/cjce.5450740527.

[16] Camacho, E. F. and Bordens, C. *Model Predictive Control*. 2nd Ed. London: Springer, 2007. ISBN: 978-0-85729-398-5. DOI: 10.1007/978-0-85729-398-5.

[17] Camponogara, E., Jia, D., Krogh, B., and Talukdar, S. "Distributed Model Predictive Control". In: *IEEE Control Systems* 22.1 (Feb. 2002), pp. 44–52. DOI: 10.1109/37.980246.

[18] Chen, C. and Shaw, L. "On Receding Horizon Feedback Control". In: *Automatica* 18.3 (1982), pp. 349–352. DOI: 10.1016/0005-1098(82)90096-6.

[19] Chen, H. and Allgöwer, F. "A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability". In: *Automatica* 34.10 (Oct. 1998), pp. 1205–1217. DOI: 10.1016/S0005-1098(98)00073-9.

[20] Chen, X., Heidarinejad, M., Liu, J., and D.Christofides, P. "Distributed Economic MPC: Application to a Nonlinear Chemical Process Network". In: *Journal of Process Control* 22.4 (Apr. 2012), pp. 689–699. DOI: 10.1016/j.jprocont.2012.01.016.

[21] Chioua, M., Srinivasan, B., Guay, M., and Perrier, M. "Dependence of the Error in the Optimal Solution of Perturbation-Based Extremum Seeking Methods on the Excitation Frequency". In: *The Canadian Journal of Chemical Engineering* 85.4 (Aug. 2007), pp. 447–453. DOI: 10.1002/cjce.5450850407.

[22] Chisci, L., Lombardi, A., and Mosca, E. "Dual-Receding Horizon Control of Constrained Discrete Time Systems". In: *European Journal of Control* 2.4 (1996), pp. 278–285. DOI: 10.1016/S0947-3580(96)70052-3.

[23] Choi, J., Krstić, M., Ariyur, K., and Lee, J. "Extremum Seeking Control for Discrete-Time Systems". In: *IEEE Transactions on Automatic Control* 47.2 (Feb. 2002), pp. 319–323. DOI: 10.1109/9.983370.

[24] Christofides, P. D., Scattolini, R., Muñoz de la Peña, D., and Liu, J. "Distributed Model Predictive Control: A Tutorial Review and Future Research Directions". In: *Computers & Chemical Engineering* 51.5 (Apr. 2013), pp. 21–41. DOI: 10.1016/j.compchemeng.2012.05.011.

[25] *Constrained Nonlinear Optimization Algorithms*. Software Documention. Version 8.5.0 R2015a. The MathWorks, Inc. Natick, Massachusetts, 2017. URL: https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html.

[26] Crichlow, J. *An Introduction to Distributed and Parallel Computing*. 1st Ed. Prentice Hall, 1988. ISBN: 978-0-13481-086-7.

[27] Cutler, C. and Ramaker, B. "Dynamic Matrix Control - A Computer Control Algorithm". In: *Proceedings of the 86th National Meeting of the American Institute of Chemical Engineers*. Houston, TX: Dynamic Matrix Control Corporation, Apr. 1979.

[28] Davis, J., Edgar, T., Porter, J., Bernaden, J., and Sarli, M. "Smart Manufacturing, Manufacturing Intelligence and Demand-Dynamic Performance". In: *Computers & Chemical Engineering* 47 (Dec. 2012), pp. 145–156. DOI: 10.1016/j.compchemeng.2012.06.037.

[29] De Nicolao, G., Magni, L., and Scattolini, R. "Stabilizing Nonlinear Receding Horizon Control via a Nonquadratic Penalty". In: *Proceedings of th IMACS Multiconference on Computational Engineering in Systems Applications*. Vol. 1. Lille, France, 1996, pp. 185–187.

[30] DeHaan, D. S. "Realtime Adaptive Methods for Model Predictive Control of Nonlinear Systems". Ph.D. Thesis. Kingston, Canada: Queen's University, July 2006.

[31] DeHaan, D. and Guay, M. "Extremum-Seeking Control of State-Constrained Nonlinear Systems". In: *Automatica* 41.9 (Sept. 2005), pp. 1567–1574. DOI: 10.1016/j.automatica.2005.03.030.

[32] Dockner, E. and Neck, R. "Cooperative and Non-Cooperative Solutions for a Linear-Quadratic Differential Game Model of Stabilization Policies". In: *Analysis and Optimization of Systems*. Ed. by Bensoussan, A. and Lions, J. Vol. 83. Lecture Notes in Control and Information Sciences. 2006, pp. 807–819. ISBN: 978-3-540-16729-7. DOI: 10.1007/BFb0007608.

[33] Dougherty, S. and Guay, M. "An Extremum-Seeking Controller for Distributed Optimization Over Sensor Networks". In: *IEEE Transactions on Automatic Control* 62.2 (Feb. 2017), pp. 928–933. DOI: 10.1109/TAC.2016.2566806.

[34] Douglas, J. "Periodic Reactor Operation". In: *Industrial and Engineering Chemistry Process Design and Development* 6.1 (Jan. 1967), pp. 43–48. DOI: 10.1021/i260021a008.

[35] Du, H., Chen, M., and Wen, G. "Leader–Following Attitude Consensus for Spacecraft Formation with Rigid and Flexible Spacecraft". In: *Journal of Guidance, Control, and Dynamics* 39.4 (2016), pp. 944–951. DOI: 10.2514/1.G001273.

[36] Dunbar, W. "Distributed Receding Horizon Control of Dynamically Coupled Nonlinear Systems". In: *IEEE Transactions on Automatic Control* 52.7 (July 2007), pp. 1249–1263. DOI: 10.1109/TAC.2007.900828.

[37] Dunbar, W. and Murray, R. "Distributed Receding Horizon Control for Multi-Vehicle Formation Stabilization". In: *Automatica* 42.4 (Apr. 2006), pp. 549–558. DOI: 10.1016/j.automatica.2005.12.008.

[38] Durlauf, S. N. and Blume, L. E., eds. *Game Theory*. The New Palgrave Economics Collection. Palgrave Macmillan UK, 2010. ISBN: 978-0-230-28084-7. DOI: 10.1057/9780230280847.

[39] Dutta, A., Ionescu, C., and Keyser, R. D. "A Pragmatic Approach to Distributed Nonlinear Model Predictive Control: Application to a Hydrostatic Drivetrain". In: *Optimal Control Applications and Methods* 36 (2015), pp. 369–380. DOI: 10.1002/oca.2141.

[40] Ebegbulem, J. "Distributed Control of Multi-Agent Systems Using Extremum-Seeking". Masters Thesis. Kingston, Canada: Queen's University, Mar. 2016.

[41] Ebegbulem, J. "Distributed Control of Multi-Agent Systems Over Unknown Communication Networks Using Extremum Seeking". In: *Journal of Process Control* In Press (2017). DOI: 10.1016/j.jprocont.2017.09.002.

[42] Ebegbulem, J. "Distributed Extremum Seeking Control for Wind Farm Power Maximization". In: *Proceedings of the 2017 IFAC World Congress*. Vol. PP. International Federation of Automatic Control. July 2017.

[43] Ellis, M., Durand, H., and Christofides, P. D. "A Tutorial Review of Economic Model Predictive Control Methods". In: *Journal of Process Control* 24.8 (Aug. 2014), pp. 1156–1178. DOI: 10.1016/j.jprocont.2014.03.010.

[44] Ellis, M., Zhang, J., Liu, J., and Christofides, P. D. "Robust Moving Horizon Estimation Based Output Feedback Economic Model Predictive Control". In: *Systems & Control Letters* 68 (June 2014), pp. 101–109. DOI: 10.1016/j.sysconle.2014.03.003.

[45] Farina, M. and Scattolini, R. "Distributed Non-Cooperative MPC with Neighbor-to-Neighbor Communication". In: *IFAC Proceedings Volumes* 44.1 (18th IFAC World Congress Jan. 2011), pp. 404–409. DOI: 10.3182/20110828-6-IT-1002.01092.

[46] Ferramosca, A., Rawlings, J. B., Limon, D., and Camacho, E. F. "Economic MPC for a Changing Economic Criterion". In: *49th IEEE Conference on Decision and Control (CDC)*. Atlanta, GA, Dec. 2010, pp. 6131–6136. DOI: 10.1109/CDC.2010.5717482.

[47] Fiacco, A. V. and McCormick, G. P. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM Publications, 1990. ISBN: 978-0-89871-254-4. DOI: 10.1137/1.9781611971316.

[48] Findeisen, R., Imsland, L., Allgöwer, F., and Foss, B. A. "State and Output Feedback Nonlinear Model Predictive Control: An Overview". In: *European Journal of Control* 9.2 (2003), pp. 190–206. DOI: 10.3166/ejc.9.190-206.

[49] Freeman, R. A., Yang, P., and Lynch, K. M. "Stability and Convergence Properties of Dynamic Average Consensus Estimators". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA, 2006, pp. 398–403. DOI: 10.1109/CDC.2006.377078.

[50] Gao, Y., Dai, L., Xia, Y., and Liu, Y. "Distributed Model Predictive Control for Consensus of Nonlinear Second-Order Multi-Agent Systems". In: *International Journal of Robust and Nonlinear Control* 27.5 (Mar. 2017), pp. 830–842. DOI: 10.1002/rnc.3603.

[51] Ghaffari, A., Krstić, M., and Nešić, D. "Multivariable Newton-Based Extremum Seeking". In: *Automatica* 48.8 (Aug. 2012), pp. 1756–1767. DOI: 10.1016/j.automatica.2012.05.059.

[52] Gill, P. E., Saunders, M. A., and Wong, E. "On the Performance of SQP Methods for Nonlinear Programming". In: *Modeling and Optimization: Theory and Applications*. Vol. 147. Springer Proceedings in Mathematics and Statistics. Springer Verlag, 2015, pp. 95–123. ISBN: 978-3-319-23699-5. DOI: 10.1007/978-3-319-23699-5_5.

[53] Goodwin, G. C. and Sin, K. S. *Adaptive Filtering Prediction and Control*. New Jersey: Prentice Hall, 1984. ISBN: 978-0-486-46932-4.

[54] Grüne, L. "Economic Receding Horizon Control Without Terminal Constraints". In: *Automatica* 49.3 (Mar. 2013), pp. 725–734. DOI: 10.1016/j.automatica.2012.12.003.

[55] Grüne, L. and Panneck, J. *Nonlinear Model Predictive Control. Theory and Algorithms*. Communications and Control Engineering. New York: Springer, 2011. ISBN: 978-0-85729-501-9. DOI: 10.1007/978-0-85729-501-9.

[56] Guay, M., Vandermeulen, I., Dougherty, S., and McLellan, P. "Distributed Extremum-Seeking Control over Networks of Dynamic Agents". In: *Proceedings of the 2015 American Control Conference*. Chicago, IL: IEEE, July 2015, pp. 159–164. DOI: 10.1109/ACC.2015.7170728.

[57] Guay, M. "A Time-Varying Extremum-Seeking Control Approach for Discrete-Time Systems". In: *Journal of Process Control* 24.3 (Mar. 2014), pp. 98–112. DOI: 10.1016/j.jprocont.2013.11.014.

[58] Guay, M. "A Perturbation-Based Proportional Integral Extremum-Seeking Control Approach". In: *IEEE Transactions on Automatic Control* 61.11 (Nov. 2016), pp. 3370–3381. DOI: 10.1109/TAC.2016.2519840.

[59] Guay, M., Beerens, R., and Nijmeijer, H. "A Time-Varying Extremum-Seeking Approach for Discrete-Time Systems with Application to Model Predictive Control". In: *IFAC Proceedings Volumes*. Vol. 47. 3. 2014, pp. 1023–1028. DOI: 10.3182/20140824-6-ZA-1003.02598.

[60] Guay, M. and Burns, D. J. "A Comparison of Extremum Seeking Algorithms Applied to Vapor Compression System Optimization". In: *Proceedings of the 2014 American Control Conference*. Portland, OR, June 2014, pp. 1076–1081. DOI: 10.1109/ACC.2014.6859288.

[61] Guay, M. and Burns, D. J. "A Proportional Integral Extremum-Seeking Control Approach for Discrete-Time Nonlinear Systems". In: *International Journal of Control* 90.8 (2017), pp. 1543–1554. DOI: 10.1109/CDC.2015.7403314.

[62] Guay, M. and Dochain, D. "A Proportional-Integral Extremum Seeking Controller Design Technique". In: *IFAC Proceedings Volumes* 47.3 (19^{th} IFAC World Congress 2014), pp. 377–382. DOI: 10.3182/20140824-6-ZA-1003.02492.

[63] Guay, M. and Dochain, D. "A Time-Varying Extremum-Seeking Control Approach". In: *Automatica* 51 (Jan. 2015), pp. 356–363. DOI: 10.1016/j.automatica.2014.10.078.

[64] Guay, M. and Dochain, D. "A Proportional-Integral Extremum Seeking Controller Design Technique". In: *Automatica* 77 (Mar. 2017), pp. 61–67. DOI: 10.1016/j.automatica.2016.11.018.

[65] Guay, M. and Zhang, T. "Adaptive Extremum Seeking Control of Nonlinear Dynamic Systems with Parametric Uncertainties". In: *Automatica* 39.7 (July 2003), pp. 1283–1293. DOI: 10.1016/S0005-1098(03)00105-5.

[66] Gupta, S. K., Kar, K., Mishra, S., and Wen, J. T. "Collaborative Energy and Thermal Comfort Management Through Distributed Consensus Algorithms". In: *IEEE Transactions on Automation Science and Engineering* 12.4 (Oct. 2015), pp. 1285–1296. DOI: 10.1109/TASE.2015.2468730.

[67] Heidarinejad, M., Liu, J., and Christofides, P. D. "State-Estimation-Based Economic Model Predictive Control of Nonlinear Systems". In: *Systems & Control Letters* 61.9 (Sept. 2012), pp. 926–935. DOI: 10.1016/j.sysconle.2012.06.007.

[68] Huang, R., Biegler, L. T., and Harinath, E. "Robust Stability of Economically Oriented Infinite Horizon NMPC that Includes Cyclic Processes". In: *Journal of Process Control* 22.1 (Jan. 2012), pp. 51–59. DOI: 10.1016/j.jprocont.2011.10.010.

[69] Jadbabaie, A., Yu, J., and Hauser, J. "Unconstrained Receding Horizon Control of Nonlinear Systems". In: *IEEE Transactions on Automatic Control* 46.5 (May 2001). DOI: 10.1109/9.920800.

[70] Jia, D. and Krogh, B. "Distributed Model Predictive Control". In: *Proceedings of the 2001 American Control Conference*. Vol. 4. Arlington, VA, 2001, pp. 2767–2772. DOI: 10.1109/ACC.2001.946306.

[71] Kadam, J. and Marquardt, W. "Integration of Economical Optimization and Control for Intentionally Transient Process Operation". In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Ed. by Findeisen, R., Allgöwer, F., and Biegler, L. T. Lecture Notes in Control and Information Sciences. New York: Springer, 2007, pp. 419–434. ISBN: 978-3-540-72699-9. DOI: 10.1007/978-3-540-72699-9_34.

[72] Kalman, R. E. "Contributions to the Theory of Optimal Control". In: *Boletin de la Sociedad Matematica Mexicana (Bulletin of the Mexican Mathematical Society)* 5 (1960), pp. 102–119.

[73] Kaplan, S. M. *Wiley Electrical and Electronics Engineering Dictionary*. 1st Ed. Wiley - IEEE Press, 2004. ISBN: 978-0-47140-224-4.

[74] Kazakevich, V. (K. "On Extremum Regulation (Об Экстремальном Регулировании)". Ph.D. Thesis. Moscow, Russia: Moscow State Technical University, 1944.

[75] Kazakevich, V. (K. "Extremum Control of Objects with Inertia and of Unstable Objects (Об Экстремальном Регулировании Инерционных и Неустойчивых Объектов)". в: *Reports of the USSR Academy of Sciences (Доклады Академии Наук СССР)* 133.4 (1960), с. 756—759. URL: http://mi.mathnet.ru/eng/dan23856.

[76] Kazakevich, V. (K. "Theory for an Ideal Extremum Regulator Model (Теория Идеальной Модели Экстремального Регулятора)". в: *Automation and Telemechanics (Автоматика И Телемеханика)* 21.4 (1960), с. 489—505. URL: http://mi.mathnet.ru/eng/at12528.

[77] Kazakevich, V. (K. "On Improving the Quality of Extreme Regulation of Inertial Objects in the Presence of Perturbations (Об Улуьшении Качества Зкстремального Регулирования Инерционных Объектов При Наличии Возмущений)". в: *Reports of the USSR Academy of Sciences (Доклады Академии Наук СССР)* 136.4 (1961), с. 783—786. URL: http://mi.mathnet.ru/eng/dan24579.

[78] Kazakevich, V. (K. "On the Algorithm of the Fastest Search in Impulse Systems of Extreme Regulation (Об Алгоритме Наибыстрейшего Поиска в Импульсных Системах Зкстремального Регулирования)". в: *Automation and Telemechanics (Автоматика И Телемеханика)* 12 (1966), с. 71—80. URL: http://mi.mathnet.ru/eng/at11245.

[79] Keerthi, S. and Gilbert, E. "Optimal Infinite Horizon Feedback Laws for a General Class of Constrained Discrete Time Systems: Stability and Moving-Horizon Approximations". In: *Journal of Optimization Theory and Application* 57.2 (1988), pp. 265–293. DOI: 10.1007/BF00938540.

[80] Keviczky, T., Borrelli, F., and Balas, G. J. "Decentralized Receding Horizon Control for Large Scale Dynamically Decoupled Systems". In: *Automatica* 42.12 (Dec. 2006), pp. 2105–2115. DOI: 10.1016/j.automatica.2006.07.008.

[81] Kia, S., Cortés, J., and Martinez, S. "Dynamic Average Consensus Under Limited Control Authority and Privacy Requirements". In: *International Journal of Robust and Nonlinear Control* 25.13 (Sept. 2015), pp. 1941–1966. DOI: 10.1002/rnc.3178.

[82] Koehler, S., Danielson, C., and Borrelli, F. "A Primal-Dual Active-Set Method for Distributed Model Predictive Control". In: *Optimal Control Applications and Methods* 38.3 (June 2017), pp. 399–419. DOI: 10.1002/oca.2262.

[83] Kool, W., Cushman, F. A., and Gershman, S. J. "When Does Model-Based Control Pay Off?" In: *PLOS Computational Biology* 12.8 (Aug. 2016), pp. 1–34. DOI: 10.1371/journal.pcbi.1005090.

[84] Krstić, M. and Wang, H.-H. "Stability of Extremum Seeking Feedback for General Nonlinear Dynamic Systems". In: *Automatica* 36.4 (2000), pp. 595–601. DOI: 10.1016/S0005-1098(99)00183-1.

[85] Ławryńczuk, M. *Computationally Efficient Model Predictive Control Algorithms. A Neural Network Approach*. Vol. 3. Studies in Systems, Decision and Control. New York: Springer, 2014. ISBN: 978-3-319-04229-9. DOI: 10.1007/978-3-319-04229-9.

[86] Leblanc, M. "Sur l'Électrification des Chemins de Fer au Moyen de Courants Alternatifs de Fréquence Élevée". In: *Revue Générale de l'Électricité* 12.8 (Aug. 1922), pp. 275–277.

[87] Lee, C. and Bailey, J. E. "Modification of Consecutive-Competitive Reaction Selectivity by Periodic Operation". In: *Industrial and Engineering Chemistry Process Design and Development* 19.1 (Jan. 1980), pp. 160–166. DOI: 10.1021/i260073a028.

[88] Lee, J. H., Natarajana, S., and Lee, K. S. "A Model-Based Predictive Control Approach to Repetitive Control of Continuous Processes with Periodic Operations". In: *Journal of Process Control* 11.2 (Apr. 2001), pp. 195–207. DOI: 10.1016/S0959-1524(00)00047-0.

[89] Li, C., Qu, Z., and Ingram, M. "Distributed Extremum Seeking and Cooperative Control for Mobile Communication". In: *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*. Orlando, FL, Dec. 2011, pp. 4510–4515. DOI: 10.1109/CDC.2011.6161402.

[90] Lia, C., Qub, Z., and Weitnauer, M. A. "Distributed Extremum Seeking and Formation Control for Nonholonomic Mobile Network". In: *Systems & Control Letters* 75 (Jan. 2015), pp. 27–34. DOI: 10.1016/j.sysconle.2014.11.005.

[91] Liu, J., Chen, X., Muñoz de la Peña, D., and Christofides, P. D. "Sequential and Iterative Architectures for Distributed Model Predictive Control of Nonlinear Process Systems". In: *AIChE Journal* 56.8 (2010), pp. 2137–2149. DOI: 10.1002/aic.12155.

[92] Ljung, L. *System Identification: Theory for the User. Theory for the User*. New-Jersey: Prentice-Hall, Inc., 1998. ISBN: 978-0-13656-695-3.

[93] Lynch, N. A. *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. ISBN: 978-0-08050-470-4.

[94] Maestre, J., Muñoz de la Peña, D., and Camacho, E. "Distributed Model Predictive Control Based on a Cooperative Game". In: *Optimal Control Applications and Methods* 32.2 (Apr. 2011), pp. 153–176. DOI: 10.1002/oca.940.

[95] Maestre, J., Muñoz de la Peña, D., Camacho, E., and Alamo, T. "Distributed Model Predictive Control Based on Agent Negotiation". In: *Journal of Process Control* 21.5 (June 2011), pp. 685–697. DOI: 10.1016/j.jprocont.2010.12.006.

[96] Mancusi, E., Altimari, P., Russo, L., and Crescitelli, S. "Multiplicities of Temperature Wave Trains in Periodically Forced Networks of Catalytic Reactors for Reversible Exothermic Reactions". In: *Chemical Engineering Journal* 171.2 (July 2011), pp. 655–668. DOI: 10.1016/j.cej.2011.04.026.

[97] Marlin, T. and Hrymak, A. "Real-Time Operations Optimization of Continuous Processes". In: *Proceedings of the Fifth International Conference on Chemical Process Control*. Ed. by Kantor, J., Garcia, C., and Carnahan, B. Tahoe City, CA: American Institute of Chemical Engineers, 1996, pp. 156–164. ISBN: 978-0-81690-741-0.

[98] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. "Constrained Model Predictive Control: Stability and Optimality". In: *Automatica* 36.6 (June 2000), pp. 789–814. DOI: 10.1016/S0005-1098(99)00214-9.

[99] Mayne, D. Q., Seron, M. M., and Raković, S. V. "Robust Model Predictive Control of Constrained Linear Systems with Bounded Disturbances". In: *Automatica* 41.2 (Feb. 2005), pp. 219–224. DOI: 10.1016/j.automatica.2004.08.019.

[100] Meerkov, S. (M. "Асимптотические Методы Исследования Одного Класса Форсированных Режимов в Экстремальных Системах (Asymptotic Methods for Investigating a Class of Forced States in Extremal Systems)". в: *Автоматика И Телемеханика (Automation and Telemechanics)* 12 (1967), с. 114—118. URL: http://mi.mathnet.ru/eng/at10980.

[101] Mendoza-Serrano, D. and Chmielewski, D. "HVAC Control Using Infinite-Horizon Economic MPC". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. Maui, HI, Dec. 2012, pp. 6963–6968. DOI: 10.1109/CDC.2012.6426071.

[102] Menon, A. and Baras, J. S. "Collaborative Extremum Seeking for Welfare Optimization". In: *53rd IEEE Conference on Decision and Control*. Los Angeles, CA, Dec. 2014, pp. 346–351. DOI: 10.1109/CDC.2014.7039405.

[103] Michalska, H. and Mayne, D. Q. "Robust Receding Horizon Control of Constrained Nonlinear Systems". In: *IEEE Transactions on Automatic Control* 38.11 (1993), pp. 1623–1632. DOI: 10.1109/9.262032.

[104] Moase, W., Manzie, C., and Brear, M. "Newton-Like Extremum Seeking for the Control of Thermoacoustic Instability". In: *IEEE Transactions on Automatic Control* 55.9 (2010), pp. 2094–2105. DOI: 10.1109/TAC.2010.2042981.

[105] Morari, M. and Lee, J. H. "Model Predictive Control: Past, Present and Future". In: *Computers & Chemical Engineering* 23.4 (1999), pp. 667–682. DOI: 10.1016/S0098-1354(98)00301-9.

[106] Moreau, L. "Stability of Multiagent Systems with Time-Dependent Communication Links". In: *IEEE Transactions on Automatic Control* 50.2 (Feb. 2005), pp. 169–182. DOI: 10.1109/TAC.2004.841888.

[107] Morosanov, I. (M. "Method of Extremum Regulation (Методы Зкстремального Регулирования)". в: *Automation and Telemechanics (Автоматика И Телемеханика)* 18.11 (1957), с. 1029—1044. URL: http://mi.mathnet.ru/eng/at13238.

[108] Moshksar, E. and Guay, M. "Invariant Manifold Approach for Time-Varying Extremum-Seeking Control Problem". In: *Proceedings of the 19th IFAC World Congress*. IFAC. Capetown, South Africa, Aug. 2014, pp. 9129–9134. DOI: 10.3182/20140824-6-ZA-1003.02631.

[109] Müller, M. A., Angeli, D., and Allgöwer, F. "Economic Model Predictive Control with a Self-Tuning Terminal Cost". In: *European Journal of Control* 19.5 (Sept. 2013), pp. 408–416. DOI: 10.1016/j.ejcon.2013.05.019.

[110] Müller, M. A., Angeli, D., Allgöwer, F., Amrit, R., and Rawlings, J. B. "Convergence in Economic Model Predictive Control with Average Constraints". In: *Automatica* 50.12 (Dec. 2014), pp. 3100–3111. DOI: 10.1016/j.automatica.2014.10.059.

[111] Natarajan, S. and Lee, J. H. "Repetitive Model Predictive Control Applied to a Simulated Moving Bed Chromatography System". In: *Computers & Chemical Engineering* 24.2-7 (July 2000), pp. 1127–1133. DOI: 10.1016/S0098-1354(00)00493-2.

[112] Nathan, D., Buse, J., Davidson, M., Ferrannini, E., Holman, R., Sherwin, R., and Zinman, B. "Medical Management of Hyperglycaemia in Type 2 Diabetes Mellitus: A Consensus Algorithm for the Initiation and Adjustment of Therapy". In: *Diabetologia* 52.17 (Jan. 2009), pp. 17–30. DOI: 10.1007/s00125-008-1157-y.

[113] Necoara, I., Nedelcu, V., and Dumitrache, I. "Parallel and Distributed Optimization Methods for Estimation and Control in Networks". In: *Journal of Process Control* 21.5 (June 2011), pp. 756–766. DOI: 10.1016/j.jprocont.2010.12.010.

[114] Nedic, A. and Ozdaglar, A. "Distributed Subgradient Methods for Multi-Agent Optimization". In: *IEEE Transactions on Automatic Control* 54.1 (Jan. 2009), pp. 48–61. DOI: 10.1109/TAC.2008.2009515.

[115] Negenborn, R. "Multi-Agent Model Predictive Control with Applications to Power Networks". Ph.D. Thesis. Delft, Netherlands: Delft University of Technology, 2007.

[116] Nešić, D., Mohammadi, A., and Manzie, C. "A Systematic Approach to Extremum-Seeking Based on Parameter Estimation". In: *Proceedings of the 49th IEEE Conference on Decision and Control*. IEEE. Atlanta, GA, Dec. 2010. DOI: 10.1109/CDC.2010.5716937.

[117] Nocedal, J. and Wright, S. J. *Numerical Optimization*. 2nd Ed. New York: Springer, 2006. Chap. 19. ISBN: 978-0-387-40065-5. DOI: 10.1007/978-0-387-40065-5.

[118] Olfati-Saber, R., Fax, J., and Murray, R. "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE* 95.1 (Jan. 2007), pp. 215–233. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.887293.

[119] Olfati-Saber, R. and Murray, R. M. "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays". In: *IEEE Transactions on Automatic Control* 49.9 (2004), pp. 1520–1533. DOI: 10.1109/TAC.2004.834113.

[120] Omell, B. P. and Chmielewski, D. J. "IGCC Power Plant Dispatch Using Infinite-Horizon Economic Model Predictive Control". In: *Industrial and Engineering Chemistry Research* 52.9 (2013), pp. 3151–3164. DOI: 10.1021/ie3008665.

[121] Özgülşen, F., Kendra, S. J., and Çinar, A. "Nonlinear Predictive Control of Periodically Forced Chemical Reactors". In: *AIChE Journal* 39.4 (Apr. 1993), pp. 589–598. DOI: 10.1002/aic.690390407.

[122] Parisini, T. and Zoppoli, R. "A Receding-Horizon Regulator for Nonlinear Systems and a Neural Approximation". In: *Automatica* 31.10 (Oct. 1995), pp. 1443–1451. DOI: 10.1016/0005-1098(95)00044-W.

[123] Perry, R. and Green, D. *Perry's Chemical Engineers' Handbook*. 8th Ed. McGraw-Hill Education, 2007. ISBN: 978-0-07142-294-9.

[124] Poveda, J., Benosman, M., and Teel, A. "Distributed Extremum Seeking in Multi-Agent Systems with Arbitrary Switching Graphs". In: *Proceedings of the 2017 IFAC World Congress*. Vol. PP. International Federation of Automatic Control. July 2017. URL: https://pdfs.semanticscholar.org/0cdf/dc6be8f01640b296eaac91e80e65753a3bdd.pdf.

[125] Poveda, J. and Quijano, N. "Distributed Extremum Seeking for Real-Time Resource Allocation". In: *Proceedings of the 2013 American Control Conference*. Washington DC, DOC, June 2013, pp. 2772–2777. DOI: 10.1109/ACC.2013.6580254.

[126] Qin, S. J. and Badgwell, T. A. "A Survey of Industrial Model Predictive Control Technology". In: *Control Engineering Practice* 11.7 (2003), pp. 733–764. DOI: 10.1016/S0967-0661(02)00186-7.

[127] Rantzer, A. "Dynamic Dual Decomposition for Distributed Control". In: *Proceedings of the 2009 American Control Conference*. St. Louis, MO, June 2009, pp. 884–888. DOI: 10.1109/ACC.2009.5160224.

[128] Rawlings, J. B., Angeli, D., and Bates, C. N. "Fundamentals of Economic Model Predictive Control". In: *51st IEEE Conference on Decision and Control*. Maui, HI: IEEE, Dec. 2012, pp. 3851–3861. DOI: 10.1109/CDC.2012.6425822.

[129] Rawlings, J. B. and Mayne, D. Q. *Model Predictive Control: Theory and Design*. Madison: Nob Hill Publishing, 2015. ISBN: 978-0-975-93770-9.

[130] Rawlings, J. B. and Stewart, B. T. "Coordinating Multiple Optimization-Based Controllers: New Opportunities and Challenges". In: *Journal of Process Control* 18.9 (Oct. 2008), pp. 839–845. DOI: 10.1016/j.jprocont.2008.06.005.

[131] Richalet, J., Rault, A., Testud, J., and Papon, J. "Model Predictive Heuristic Control: Applications to Industrial Processes". In: *Automatica* 14.5 (Sept. 1978), pp. 413–428. DOI: 10.1016/0005-1098(78)90001-8.

[132] Rotea, M. "Analysis of Multivariable Extremum Seeking Algorithms". In: *Proceedings of the 2000 American Control Conference*. IEEE. Chicago, IL, June 2000, pp. 433–437. DOI: 10.1109/ACC.2000.878937.

[133] Sánchez-Peña, R. S. and Sznaier, M. *Robust Systems Theory and Applications*. Vol. 12. Adaptive and Learning Systems for Signal Processing, Communications and Control. John Wiley, 1998. ISBN: 978-0-471-17627-5.

[134] Scheu, H., Busch, J., and Marquardt, W. "Nonlinear Distributed Dynamic Optimization Based on First Order Sensitivities". In: *Proceedings of the 2010 American Control Conference*. Baltimore, MD, June 2010, pp. 1574–1579. DOI: 10.1109/ACC.2010.5531587.

[135] Scheu, H. and Marquardt, W. "Sensitivity-Based Coordination in Distributed Model Predictive Control". In: *Journal of Process Control* 21.5 (June 2011), pp. 715–728. DOI: 10.1016/j.jprocont.2011.01.013.

[136] Schittkowski, K. "NLQPL: A FORTRAN-Subroutine Solving Constrained Nonlinear Programming Problems". In: *Annals of Operations Research* 5 (1985), pp. 485–500.

[137] Shu, X., Rigopoulos, K., and Cinar, A. "Vibrational Control of an Exothermic CSTR: Productivity Improvement by Multiple Input Oscillations". In: *IEEE Transactions on Automatic Control* 34.2 (Feb. 1989), pp. 193–196. DOI: 10.1109/9.21097.

[138] Siirola, J. and Edgar, T. "Process Energy Systems: Control, Economic, and Sustainability Objectives". In: *Computers & Chemical Engineering* 47 (Dec. 2012), pp. 134–144. DOI: 10.1016/j.compchemeng.2012.06.019.

[139] Silveston, P. L. and Hudgins, R. R. *Periodic Operation of Chemical Reactors*. 1st Ed. Butterworth-Heinemann, 2013. ISBN: 978-0-12-391854-3.

[140] Sterman, L. E. and Ydstie, B. E. "Periodic Forcing of the CSTR: An Application of the Generalized π-Criterion". In: *AIChE Journal* 37.7 (July 1991), pp. 986–996. DOI: 10.1002/aic.690370704.

[141] Stewart, B. T., Venkat, A. N., Rawlings, J. B., Wright, S. J., and Pannocchia, G. "Cooperative Distributed Model Predictive Control". In: *Systems & Control Letters* 59.8 (Aug. 2010), pp. 460–469. DOI: 10.1016/j.sysconle.2010.06.005.

[142] Stewart, B. T., Wright, S. J., and Rawlings, J. B. "Cooperative Distributed Model Predictive Control for Nonlinear Systems". In: *Journal of Process Control* 21.5 (June 2011), pp. 698–704. DOI: 10.1016/j.jprocont.2010.11.004.

[143] Stone, P. and Veloso, M. "Multiagent Systems: A Survey from a Machine Learning Perspective". In: *Autonomous Robots* 8.3 (June 2000), pp. 345–383. DOI: 10.1023/A:1008942012299.

[144] Talbi, E.-G., ed. *Metaheuristics for Bi-level Optimization*. Vol. 482. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-37838-6. DOI: 10.1007/978-3-642-37838-6.

[145] Tan, Y., Moase, W., Manzie, C., Nešić, D., and Mareels, I. "Extremum Seeking from 1922 to 2010". In: *Proceedings of the 29th Chinese Control Conference*. Beijing, China, July 2010, pp. 14–26. URL: http://ieeexplore.ieee.org/document/5572972/.

[146] Tan, Y., Nešić, D., and Mareels, I. "On Non-Local Stability Properties of Extremum Seeking Control". In: *Automatica* 42.6 (2006), pp. 889–903. DOI: 10.1016/j.automatica.2006.01.014.

[147] Van den Hof, P., Scherer, C., and Heuberger, P., eds. *Model-Based Control. Bridging Rigorous Theory and Advanced Technology*. Springer, 2009. ISBN: 978-1-4419-0894-0.

[148] Vandermeulen, I. "Distributed Extremum-Seeking Control with Applications to High Altitude Balloons". Masters Thesis. Kingston, Canada: Queen's University, Aug. 2016.

[149] Vandermeulen, I., Guay, M., and McLellan, P. "Discrete-Time Distributed Extremum-Seeking Control over Networks with Unstable Dynamics". In: *IEEE Transactions on Control of Network Systems* PP (Apr. 2017). DOI: 10.1109/TCNS.2017.2691464.

[150] Vandermeulen, I., Guay, M., and McLellan, P. "Distributed Control of High-Altitude Balloon Formation by Extremum-Seeking Control". In: *IEEE Transactions on Control Systems Technology* PP (May 2017). DOI: 10.1109/TCST.2017.2692742.

[151] Vandermeulen, I., Guay, M., and McLellan, P. "Formation Control of High-Altitude Balloons Experiencing Real Wind Currents by Discrete-Time Distributed Extremum Seeking Control". In: *Proceedings of the 2017 American Control Conference*. Seattle, WA: IEEE, May 2017, pp. 991–996. DOI: 10.23919/ACC.2017.7963082.

[152] Venkat, A. N., Rawlings, J. B., and Wright, S. J. "Stability and Optimality of Distributed Model Predictive Control". In: *Proceedings of the 44^{th} IEEE Conference on Decision and Control and the European Control Conference*. Barcelona, Spain, 2005, pp. 6680–6685. DOI: 10.1109/CDC.2005.1583235.

[153] Venkat, A. N., Rawlings, J. B., and Wright, S. J. "Distributed Model Predictive Control of Large-Scale Systems". In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Ed. by Findeisen, R., Allgöwer, F., and Biegler, L. T. Lecture Notes in Control and Information Sciences. New York: Springer, 2007, pp. 591–605. ISBN: 978-3-540-72698-2.

[154] Wang, Z. and Chong, J. O. "Distributed Model Predictive Control of Linear Discrete-Time Systems with Local and Global Constraints". In: *Automatica* 81 (July 2017), pp. 184–195. DOI: 10.1016/j.automatica.2017.03.027.

[155] Wooldridge, M. *An Introduction to Multi Agent Systems*. 2nd Ed. Wiley Publishing, 2009. ISBN: 978-0-47051-946-2.

[156] Wright, M. H. "The Interior-Point Revolution in Optimization: History, Recent Developments, and Lasting Consequences". In: *Bulletin of the American Mathematical Society* 42 (2005), pp. 39–56. DOI: 10.1090/S0273-0979-04-01040-7.

[157] Xu, J. M. and Soh, Y. C. "Distributed Extremum Seeking Control of Networked Large-Scale Systems Under Constraints". In: *Proceedings of the 52^{nd} IEEE Conference on Decision and Control*. Firenze, Italy, Dec. 2013, pp. 2187–2192. DOI: 10.1109/CDC.2013.6760206.

[158] Ye, M. and Hu, G. "Distributed Extremum Seeking for Constrained Networked Optimization and Its Application to Energy Consumption Control in Smart Grid". In: *IEEE Transactions on Control Systems Technology* 24.6 (Nov. 2016), pp. 2048–2058. DOI: 10.1109/TCST.2016.2517574.

[159] Zhang, C. and Ordóñez, R. "Robust and Adaptive Design of Numerical Optimization-Based Extremum Seeking Control". In: *Automatica* 45.3 (Mar. 2009), pp. 634–646. DOI: 10.1016/j.automatica.2008.09.025.

[160] Zhang, J., Sheng, V. S., Li, Q., Wu, J., and Wu, X. "Consensus Algorithms for Biased Labeling in Crowdsourcing". In: *Information Sciences* 282-283 (Mar. 2017), pp. 254–273. DOI: 10.1016/j.ins.2016.12.026.

[161] Zhu, M. and Martinez, S. "Discrete-time Dynamic Average Consensus". In: *Automatica* 46.2 (Feb. 2010), pp. 322–329. DOI: 10.1016/j.automatica.2009.10.021.

Image Sources

[Im1] C. Design. *Factory*. The Noun Project. July 2017. URL: https://thenounproject.com/term/factory/642644/.

[Im2] Freepik. *Estructura de una Fábrica*. FlatIcon. July 2017. URL: http://www.flaticon.es/icono-gratis/estructura-de-una-fabrica_18222.

[Im3] L. Squid. *Factory*. The Noun Project. July 2017. URL: https://thenounproject.com/term/factory/26209/.

[Im4] C. Stall. *Factory*. The Noun Project. July 2017. URL: https://thenounproject.com/term/mill/175832/.

[Im5] A. Wattenbergeror. *Factory*. The Noun Project. July 2017. URL: https://thenounproject.com/term/factory/10948/.