Applied Distributed Model Predictive Control for Energy Efficient Buildings
and Ramp Metering

by

Sarah Muraoka Koehler

A dissertation submitted in partial satisfaction of the requirements for the degree of
Doctor of Philosophy
in
Engineering - Mechanical Engineering
in the
Graduate Division
of the
University of California, Berkeley

Committee in charge:
Professor Francesco Borrelli, Chair
Professor Roberto Horowitz
Professor Laurent El Ghaoui

Fall 2016
Applied Distributed Model Predictive Control for Energy Efficient Buildings and Ramp Metering

Copyright 2016 by Sarah Muraoka Koehler
Abstract

Applied Distributed Model Predictive Control for Energy Efficient Buildings and Ramp Metering

by

Sarah Muraoka Koehler

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

Industrial large-scale control problems present an interesting algorithmic design challenge. A number of controllers must cooperate in real-time on a network of embedded hardware with limited computing power in order to maximize system efficiency while respecting constraints and despite communication delays. Model predictive control (MPC) can automatically synthesize a centralized controller which optimizes an objective function subject to a system model, constraints, and predictions of disturbance. Unfortunately, the computations required by model predictive controllers for large-scale systems often limit its industrial implementation only to medium-scale slow processes.

Distributed model predictive control (DMPC) enters the picture as a way to decentralize a large-scale model predictive control problem. The main idea of DMPC is to split the computations required by the MPC problem amongst distributed processors that can compute in parallel and communicate iteratively to find a solution. Some popularly proposed solutions are distributed optimization algorithms such as dual decomposition and the alternating direction method of multipliers (ADMM). However, these algorithms ignore two practical challenges: substantial communication delays present in control systems and also problem non-convexity.

This thesis presents two novel and practically effective DMPC algorithms. The first DMPC algorithm is based on a primal-dual active-set method which achieves fast convergence, making it suitable for large-scale control applications which have a large communication delay across its communication network. In particular, this algorithm is suited for MPC problems with a quadratic cost, linear dynamics, forecasted demand, and box constraints. We measure the performance of this algorithm and show that it significantly outperforms both dual decomposition and ADMM in the presence of communication delay. The second DMPC algorithm is based on an inexact interior point method which is suited for nonlinear optimization problems. The parallel computation of the algorithm exploits iterative linear algebra methods for the main linear algebra computations in the algorithm. We show
that the splitting of the algorithm is flexible and can thus be applied to various distributed platform configurations.

The two proposed algorithms are applied to two main energy and transportation control problems. The first application is energy efficient building control. Buildings represent 40% of energy consumption in the United States. Thus, it is significant to improve the energy efficiency of buildings. The goal is to minimize energy consumption subject to the physics of the building (e.g. heat transfer laws), the constraints of the actuators as well as the desired operating constraints (thermal comfort of the occupants), and heat load on the system. In this thesis, we describe the control systems of forced air building systems in practice. We discuss the “Trim and Respond” algorithm which is a distributed control algorithm that is used in practice, and show that it performs similarly to a one-step explicit DMPC algorithm. Then, we apply the novel distributed primal-dual active-set method and provide extensive numerical results for the building MPC problem.

The second main application is the control of ramp metering signals to optimize traffic flow through a freeway system. This application is particularly important since urban congestion has more than doubled in the past few decades. The ramp metering problem is to maximize freeway throughput subject to freeway dynamics (derived from mass conservation), actuation constraints, freeway capacity constraints, and predicted traffic demand. In this thesis, we develop a hybrid model predictive controller for ramp metering that is guaranteed to be persistently feasible and stable. This contrasts to previous work on MPC for ramp metering where such guarantees are absent. We apply a smoothing method to the hybrid model predictive controller and apply the inexact interior point method to this nonlinear non-convex ramp metering problem.
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Model Predictive Control</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Distributed Model Predictive Control</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>Building Thermal Control</td>
<td>10</td>
</tr>
<tr>
<td>1.4</td>
<td>Ramp Metering Control</td>
<td>11</td>
</tr>
<tr>
<td>1.5</td>
<td>Thesis Layout</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>Building Forced Air Systems</td>
<td>15</td>
</tr>
<tr>
<td>2.1</td>
<td>Forced Air Systems</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Building Automation Systems</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Current Building Control Practices</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>Model Predictive Control</td>
<td>22</td>
</tr>
<tr>
<td>2.5</td>
<td>Distributed Model Predictive Control for Forced Air Systems</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>Building Temperature Distributed Control via Explicit MPC and “Trim and Respond” Methods</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Model Predictive Control</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Explicit Distributed MPC Solution</td>
<td>28</td>
</tr>
<tr>
<td>3.4</td>
<td>Trim and Respond</td>
<td>33</td>
</tr>
<tr>
<td>3.5</td>
<td>Comparison of Methods</td>
<td>35</td>
</tr>
<tr>
<td>3.6</td>
<td>Results: Simulations</td>
<td>36</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusion</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>A Primal-Dual Active-Set Method for Distributed Model Predictive Control</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>Primal-Dual Active-Set Method</td>
<td>40</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Pages</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>4.2</td>
<td>A Distributed Implementation on Star Communication Networks</td>
<td>44</td>
</tr>
<tr>
<td>4.3</td>
<td>Theoretical Analysis</td>
<td>51</td>
</tr>
<tr>
<td>4.4</td>
<td>Example: Thermal Building Control</td>
<td>53</td>
</tr>
<tr>
<td>4.5</td>
<td>Results</td>
<td>54</td>
</tr>
<tr>
<td>4.6</td>
<td>Conclusions</td>
<td>59</td>
</tr>
<tr>
<td>5</td>
<td>Freeway Ramp Metering Systems</td>
<td>60</td>
</tr>
<tr>
<td>5.1</td>
<td>The Asymmetric Cell Transmission Model</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Ramp Metering Systems</td>
<td>63</td>
</tr>
<tr>
<td>5.3</td>
<td>Current Ramp Metering Practices</td>
<td>64</td>
</tr>
<tr>
<td>5.4</td>
<td>Model Predictive Control</td>
<td>65</td>
</tr>
<tr>
<td>5.5</td>
<td>Distributed Model Predictive Control for Ramp Metering Systems</td>
<td>67</td>
</tr>
<tr>
<td>6</td>
<td>Stable Hybrid Model Predictive Control for Ramp Metering</td>
<td>68</td>
</tr>
<tr>
<td>6.1</td>
<td>Piecewise Affine Model</td>
<td>69</td>
</tr>
<tr>
<td>6.2</td>
<td>Hybrid Model Predictive Control</td>
<td>71</td>
</tr>
<tr>
<td>6.3</td>
<td>Case Studies</td>
<td>77</td>
</tr>
<tr>
<td>6.4</td>
<td>Conclusions and Future Work</td>
<td>79</td>
</tr>
<tr>
<td>7</td>
<td>An Inexact Interior Point Algorithm for DMPC with Application to Ramp Metering</td>
<td>82</td>
</tr>
<tr>
<td>7.1</td>
<td>Nonlinear Model Predictive Control</td>
<td>83</td>
</tr>
<tr>
<td>7.2</td>
<td>Inexact Interior Point Method</td>
<td>85</td>
</tr>
<tr>
<td>7.3</td>
<td>Iterative Linear Algebra Methods</td>
<td>90</td>
</tr>
<tr>
<td>7.4</td>
<td>Preconditioners</td>
<td>92</td>
</tr>
<tr>
<td>7.5</td>
<td>Distributed Computation</td>
<td>94</td>
</tr>
<tr>
<td>7.6</td>
<td>Application: Freeway Ramp Metering</td>
<td>95</td>
</tr>
<tr>
<td>7.7</td>
<td>Results</td>
<td>98</td>
</tr>
<tr>
<td>7.8</td>
<td>Conclusions and Future Work</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>104</td>
</tr>
</tbody>
</table>
List of Figures

1.1 MPC Scheme [1] .......................................................... 2

2.1 System Scheme .......................................................... 16
2.2 Typical layout of building embedded platforms and routers. Physical components and connections are denoted by solid lines; control platforms and wired connections are denoted by dashed lines; routers and communication backbone are denoted by purple dotted lines. .............................................. 19
2.3 Star Network Communication Graph ...................................... 22

3.1 Top: disturbance profile over one day. Left column: MPC simulation results. Right column: “Trim and Respond” simulation results ........................................... 37

4.1 Network Flow Dynamic Coupling Example ................................. 45
4.2 Star Network Communication Graph ......................................... 45
4.3 Communication and Execution Times vs Problem Size ..................... 56
4.4 Closed Loop Profiles ...................................................... 58

5.1 Schematic of Freeway Segmentation ........................................ 60
5.2 Demand and Supply for Segment $i$. The yellow line shows the maximum capacity of flow, and the green and red lines show the demand and supply respectively. 62
5.3 Ramp Metering Schematic .................................................. 63

6.1 Trajectory of Hybrid MPC controller. The blue and red lines represent trajectories of first and second segments, respectively. ................................. 78
6.2 Trajectory of LP controller ................................................. 78
6.3 Trajectory of a short-sighted LP controller ................................. 80
6.4 Trajectory of a short horizon Hybrid MPC controller ..................... 80

7.1 Hybrid MPC solution for 2 link example ................................ 99
7.2 Inexact interior point solution for 2 link example ........................ 99
7.3 Trajectory using Inexact Interior Point .................................... 100
7.4 Trajectory using JuMP with Ipopt ........................................ 102
List of Tables

4.1 Theoretical Trade-offs ........................................... 53
4.2 Experimental Problem Parameters .............................. 55
4.3 Numerical Performance for HVAC problem (times values are in units of seconds) 55
4.4 Numerical Performance for Random problem (times values are in units of seconds) 56
4.5 Numerical Performance of qpOASES (times values are in units of seconds) ........ 57
4.6 Numerical Comparison of Active-Set Methods in Closed Loop .......................... 58
Acknowledgments

First and foremost, I would like to thank my advisor, Francesco Borrelli, for guiding me through the PhD process. I am extremely appreciative of his patience and thoughtfulness over the years. I would also like to thank my thesis committee members, Roberto Horowitz and Laurent El Ghaoui, for their feedback on my research.

In my research in energy efficient buildings, I am incredibly grateful to both Yudong Ma and Allan Daly for their help. I must thank Yudong for being the friendly building MPC expert when I started out in the lab, and Allan for helping me understand building control practices (especially “Trim and Respond”). Frank Chuang, Tony Kelman, and Sergey Vichik were also very helpful in discussing MPC for buildings, as well as other topics related to my thesis. I would also like to thank Claus Danielson for the guidance he provided to me during the middle years of my PhD. I have learned a lot from these people through many conversations of research and stories about life.

On the ramp metering project, I am incredibly thankful to have met and worked with Negar Mehr. She has shown incredible kindness and patience toward me. I thank my advisor, Francesco Borrelli, for enabling this particular collaboration. Additionally, I would like to thank Gabriel Gomes for his helpful feedback on our work on ramp metering.

I would like to thank all of the people who were/are a part of the MPC lab. This includes PhD students, masters students, visiting scholars, and more. Graduate school would not have been nearly as enjoyable if I had not been able to interact with such a friendly and intelligent group of people. In particular, I must thank Theresa Lin for organizing many of the social activities we did together as a lab. I would also like to thank Ashwin Carvalho and Jason Kong who were both good friends and peers to have.

I am very thankful for all of the friends I have made while living in Berkeley. This especially includes the women of GWE (Graduate Women of Engineering), the Berkeley Wine Club, SECO (Science and Engineering Community Outreach), the Jazz Dance Company of Berkeley (as well as other dancers I met through the Physical Education department), other graduate students from the Mechanical Engineering department, and other friends I have met outside the department. They have all helped balance out my life in many ways.

Going back further into the history of my life, I would like to thank Eric Kulczycki, who has been a mentor ever since I was an intern at the NASA Jet Propulsion Laboratory in 2010. He has been incredibly positive and supportive of me in my endeavors in engineering graduate school. Similarly, I must thank the NSF for its Research Experiences for Undergraduates (REUs) for getting me interested in pursuing research in graduate school. Professors Mark Yim and Daniel Koditschek from the University of Pennsylvania particularly made a positive influence on me in my REU experience at UPenn.

I am also thankful to my parents for helping me get to this stage in life. I am particularly thankful that they have always encouraged me to be a well-rounded person, embracing both my engineering and artistic sides throughout life. I would also like to thank my best friend Amanda Liebrecht and her parents for supporting me throughout almost my entire life and for especially being so excited about my pursuit of engineering!
I am incredibly thankful to my boyfriend Mike Wilson for his support and encouragement throughout my journey from college through graduate school. He has been the biggest personal cheerleader throughout the PhD process and I cannot imagine what graduate school would have been like without his support.

I am incredibly grateful to have received 5 years of fellowship funding in pursuit of my PhD. First, I must thank the Japanese American Women Alumnae of UC Berkeley for providing me with the Chancellor’s Fellowship which funded my first two years of graduate school. It inspires me that these women made something so financially significant despite historical hardships. I would also like to thank the National Science Foundation for awarding me their graduate research fellowship. I also have to thank the Lawrence Berkeley National Laboratory for funding me during the last semester of my PhD.

This material is based upon work partially supported by the National Science Foundation under Grant DGE 1106400 and No. 0844456. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
Chapter 1

Introduction

Today there exists an abundant amount of large-scale control problems: buildings [2], freeways [3], the power grid [4], unmanned aerial vehicle coordination [5], traffic platooning [6] are just a couple of examples of large-scale systems that require smart control for optimal performance. These large-scale problems share the following features:

1. The large-scale system is made up of many dynamical subsystems; each subsystem is directly connected to only a few other subsystems.

2. Each subsystem is subject to constraints on its states and inputs.

3. There is a demand or load on each subsystem which can be forecasted.

4. The subsystems are physically spread across space with distributed computing platforms and a communication network.

5. The full scale system has a global control objective, such as maximum efficiency or minimum cost. It is highly desirable to steer the system toward the global optimal objective, if a global optimal solution exists.

Model Predictive Control [7] is the only known control strategy that systematically addresses all of these control problem features. The main idea of predictive control is to use a model of the plant to predict the future evolution of the system [8], [9], [10]. At each sampling time, starting at the current state, an open-loop optimal control problem is solved over a finite horizon. The optimal command signal is applied to the process only during the following sampling interval. At the next time step, a new optimal control problem based on new measurements of the state is solved over a shifted horizon. The resultant control algorithm is referred to as Model Predictive Control (MPC), or also Receding Horizon Control (RHC). Figure 1.1 illustrates this concept.

It is well known that model predictive control is a computationally intensive strategy, and is especially cumbersome for large-scale systems and embedded systems with limited computational power. Thus, several strategies have been proposed and studied to promote
CHAPTER 1. INTRODUCTION

The main idea of decentralized MPC is that each subsystem solves a local open-loop optimal control problem in a receding horizon fashion. The subsystems either do not communicate any information with neighbors [11] or can communicate very limited information, usually once per time step before the optimal control problem is solved (i.e. the subsystems do not negotiate while solving their respective optimal control problems) [12]. Decentralized MPC has been shown to work well for systems that are dynamically uncoupled but are either coupled by cost function [13] or constraints [14]. In some cases, systems that are dynamically coupled can be decoupled by using a robust approach [15]. In general, the decentralized MPC solution will be suboptimal relative to a centralized MPC approach. However, suboptimality may not be of high importance if the cost function is a weighted tracking error since its evaluated amount may not have too much physical meaning. The weighted tracking error is a commonly used MPC cost function in these approaches.

Distributed MPC differs from decentralized MPC in that communication between subsystems may occur while the control strategy is being computed at each time step [12]. In many cases, a distributed MPC solves the centralized problem by means of a parallelizable optimization algorithm such as dual decomposition [16], the alternating direction method of multipliers [17], or the alternating minimization algorithm [18]. Such a distributed MPC algorithm consists of subsystems computing steps of an optimization algorithm and communicating information in an iterative fashion over the network in order to reach consensus for the centralized problem. Thus, this distributed MPC approach can usually avoid suboptimality with respect to the centralized solution, provided the algorithm is allowed to iterate enough to converge to the optimal solution. This approach is particularly useful when the centralized cost function is physically very meaningful; examples of energy cost and traffic efficiency are two instances in which suboptimality should be avoided if possible.

This thesis studies distributed model predictive control for large-scale systems. In particular, two novel distributed model predictive control algorithms are proposed. In this introduction, MPC design is formalized and a review of distributed MPC methods is included. In
particular, we highlight the challenges in implementing distributed MPC solutions in practice, thus motivating the research of novel distributed MPC algorithms done in this thesis. Then, two prominent large-scale control applications of energy efficient buildings and ramp metering are introduced. The end of this introduction summarizes the outline and main contributions of this thesis.

1.1 Model Predictive Control

In this section, we formalize model predictive control, discuss stability design for MPC, and describe the form of MPC for large-scale systems.

A model predictive controller is designed to minimize an objective and to satisfy constraints given a model of the system and predictions of demand. A model predictive controller solves a constrained finite time optimal control (CFTOC) problem at each time step in a receding horizon fashion. Consider the following CFTOC problem at time step $t$:

$$
\min_{X,U} \ p(x(N)) + \sum_{k=0}^{N-1} (q(x(k), u(k))) 
$$

s.t. $x(k+1) = f(x(k), u(k), \hat{d}(k))$

$x(k) \in \mathcal{X}$, $k = 1, \ldots, N$

$u(k) \in \mathcal{U}$, $k = 0, \ldots, N - 1$

$x(N) \in \mathcal{X}_f$

$x(0) = x|_t$

where the state at time step $k$ is $x(k) \in \mathcal{R}^n$ with $X = [x(0)^T, \ldots, x(N)^T]^T$, the input at time step $k$ is $u(k) \in \mathcal{R}^m$ with $U = [u(0)^T, \ldots, u(N-1)^T]^T$, $\hat{d}(k)$ is the predicted demand at time step $k$, the prediction horizon length is $N$, the dynamics are encoded by $f(\cdot)$, $\mathcal{X}$ is a polyhedron encoding state constraints, $\mathcal{U}$ is a polyhedron encoding actuator constraints, and the current state feedback $x|_t$ is used. The objective of the control problem is encoded in the cost function as a summation of stage costs $q(x(k), u(k))$ and a terminal cost $p(x(N))$. In MPC design, the terminal cost function is used in conjunction with the terminal constraint $x(N) \in \mathcal{X}_f$ to guarantee persistent feasibility and stability of the MPC problem.

Let the optimal solution to problem (1.1) at time step $t$ be

$$
U^* = [u^*(0)^T, u^*(1)^T, \ldots, u^*(n)^T]^T.
$$

At time step $t$, the solution $u^*(0)$ is implemented. The optimization (4.8) is repeated at time $t + 1$, with the updated state estimation $x|_{t+1}$ and estimated load, yielding a receding horizon control strategy.

The MPC controller (1.1)-(1.2) in general can perform poorly if not designed well. The first main issue is feasibility. The MPC problem is feasible if the following set is not empty:

$$
\mathcal{S}_{MPC} = \{ X \in \mathcal{R}^{Nn}, U \in \mathcal{R}^{Nm} \mid x(k+1) = f(x(k), u(k), \hat{d}(k)) \ \forall k = 1, \ldots, N, \}
$$
CHAPTER 1. INTRODUCTION

(1.3)

The challenge is that even if the MPC problem (1.1) is feasible at the initial time step \( t = 0 \), there may be a time \( t = T \) at which the set \( S_{\text{MPC}} \) is empty. An MPC problem is called persistently feasible if the set \( S_{\text{MPC}} \) can be guaranteed to never be empty over all time \( t \geq 0 \).

The second performance challenge is stability. As is true of any controller, it is desirable for the closed loop system to be stable. Fortunately, there are design methods for guaranteeing the persistent feasibility and stability of MPC controllers. We review here the design of stable and persistently feasible MPC controllers with quadratic cost and linear systems as described in [7]. Define the cost functions to be quadratic with penalty terms \( P, Q, R \succ 0 \):

\[
p(x(N)) = x(N)^T P x(N) \quad (1.4a)
\]

\[
q(x(k), u(k)) = x(k)^T Q x(k) + u(k)^T R u(k). \quad (1.4b)
\]

Also, consider the following linear dynamics in place of (1.1b):

\[
x(k+1) = A x(k) + B u(k) \quad \forall k \in \mathbb{N}. \quad (1.5)
\]

where \( \mathbb{N} \) is the set of all natural numbers, including 0. Then, the following steps can be taken to design a persistently feasible and stable MPC controller (1.1)-(1.2) by design of terminal cost penalty \( P \) and terminal set \( \mathcal{X}_f \):

1. Determine the infinite horizon discrete time linear quadratic regulator for (1.5) and given quadratic penalties \( Q, R \). This gives the feedback law \( u(k) = -K_{\text{LQR}} x(k) \).

2. Compute the solution of the discrete algebraic riccati equation for (1.5) and given quadratic penalties \( Q, R \). This gives the solution \( P_\infty \). This is used as the terminal cost penalty \( P \) in (1.4).

3. Compute the maximal positive invariant set of the closed loop system \( x(k+1) = (A - BK_{\text{LQR}}) x(k) \). This set can be computed by using the multi-parametric toolbox [19]. Use this set as the terminal set in (1.1e).

Using this design, the model predictive controller is persistently feasible and stable by design. For more details on the design steps as well as the proof of stability and persistent feasibility, please refer to [7].

The MPC design method described here is recommended for the stability guarantees it provides, but it is commonly not used in most implementations. This is usually because of the difficulty in computing the terminal set and even the terminal cost for general MPC problems (1.1), especially if there are nonlinear dynamics, nonlinear costs, and/or nonlinear constraints. One large contribution of this thesis is that we address the design of a persistently feasible and stable hybrid model predictive controller for ramp metering systems (see Chapter 6).
1.2 Distributed Model Predictive Control

In this thesis, we consider distributed model predictive control (DMPC) solutions that are in the form of distributed optimization algorithms applied to large-scale model predictive control problems. It is therefore useful to review the most popular distributed optimization algorithms which have been proposed in the literature. In this section, two algorithms are reviewed: dual decomposition and the alternating direction method of multipliers (ADMM). In particular, the main idea of each algorithm is summarized with a discussion of some of the relevant theoretical results as well as some practical strategies and challenges.

This thesis studies the solution of model predictive control problems for large-scale systems. A large-scale system is composed of \( r \) subsystems that are sparsely coupled. Each subsystem \( i \) is said to be connected to its neighbors \( \mathcal{N}(i) \) by coupling in the states and/or inputs of the dynamics, where \( |\mathcal{N}(i)| \ll r \). We will define \( i \) as being in its own neighbor set, i.e. \( i \in \mathcal{N}(i) \). Each subsystem has its own local performance in its states \( x_i \) and inputs \( u_i \). Thus, we introduce the following form of model predictive control for large-scale linear systems:

\[
\min_{X,U} \sum_{k=0}^{N} \sum_{i=1}^{r} l_i(x_i(k), u_i(k)) \quad (1.6a)
\]

\[
\text{s.t. } x_i(k+1) = \sum_{j \in \mathcal{N}(i)} A_{ij} x_j(k) + \sum_{j \in \mathcal{N}(i)} B_{ij} u_j(k) + d_i(k) \quad (1.6b)
\]

\[
x_i(k) \in \mathcal{X}_i, \quad i = 1, \ldots, r, \text{ and } k = 0, \ldots, N - 1 \quad (1.6c)
\]

\[
u_i(k) \in \mathcal{U}_i, \quad i = 1, \ldots, r, \text{ and } k = 0, \ldots, N - 1 \quad (1.6d)
\]

\[
x_i(N) \in \mathcal{X}_{f,i}, \quad i = 1, \ldots, r \quad (1.6e)
\]

\[
x_i(0) = x_i|_t \quad i = 1, \ldots, r \quad (1.6f)
\]

We note that the design methodology reviewed in the previous section does not necessarily provide a terminal constraint \( \mathcal{X}_f \) that is divisible into separable terminal constraints \( \mathcal{X}_{f,i} \). There is however a method available in the literature for developing separable \( \mathcal{X}_{f,i} \) for linear systems in [20].

In order to explain the distributed optimization algorithms clearly, we cast this problem into a simpler form. In particular, define \( z_i = [x_i^T(0) \ldots x_i^T(N) \ u_i^T(0) \ldots u_i^T(N-1)]^T \) and the following cost function:

\[
J_i(z_i) = \sum_{k=0}^{N} l_i(x_i(k), u_i(k)). \quad (1.7)
\]
In addition, write the state feedback (1.6f) and the dynamics (1.6b) as an equality constraint:
\[ g_{i}(z_{N(i)}) = \left[ \begin{array}{c} x_{i}(0) - x_{i}\left|_{t} \right. \\ x_{i}(1) - \sum_{j \in N(i)} A_{ij}x_{j}(0) - \sum_{j \in N(i)} B_{ij}u_{j}(0) - d_{i}(0) \\ \vdots \\ x_{i}(N) - \sum_{j \in N(i)} A_{ij}x_{j}(N-1) - \sum_{j \in N(i)} B_{ij}u_{j}(N-1) - d_{i}(N-1) \end{array} \right] = C_{i}z_{N(i)} - d_{i} = 0 \] (1.8)
where \( C_{i} \) is a large sparse matrix with instances of \( I, A_{ij}, \) and \( B_{ij} \) matrices and \( d_{i} = [d_{i}(0)^{T} \ldots d_{i}(N-1)^{T}]^{T}. \) A detailed example of \( C_{i} \) is developed in Chapter 4. Finally, lump together the polyhedron constraints \( x_{i}(k) \in X_{i}, u_{i}(k) \in U_{i}, x_{i}(N) \in X_{f,i} \) for all \( k = 0, \ldots, N \) as:
\[ A_{i}z_{i} \leq b_{i}. \] (1.9)

Then, the optimization problem can be written as:
\[
\begin{align*}
\min_{z_{1}, z_{2}, \ldots, z_{r}} & \sum_{i=1}^{r} J_{i}(z_{i}) \\
\text{s.t.} & \quad A_{i}z_{i} \leq b_{i} \quad \forall i \in \{1, \ldots, N\} \\
& \quad C_{i}z_{N(i)} = d_{i} \quad \forall i \in \{1, \ldots, N\}
\end{align*}
\] (1.10)

We will use this optimization problem form in the next section to review two popular methods for distributed model predictive control.

**Dual Decomposition**

Dual decomposition has been studied extensively for large-scale optimization problems [21, 22] and has somewhat recently become popular for application to control [23, 16, 24]. The main idea of dual decomposition is to take advantage of the separability of the cost function, despite the presence of non-separable constraints. This separability enables simple optimization algorithms such as the projected gradient, subgradient, or cutting-plane methods to be encoded as a distributed algorithm.

Consider the optimization problem presented in (1.10). The cost function (1.10a) is *separable* in \( z_{i} \) because it is equal to the sum of functions \( J_{i} \) which are each only a function of \( z_{i} \). Thus, if the problem (1.10) were unconstrained, each cost function \( J_{i}(z_{i}) \) could be optimized *separately* as no variable \( z_{i} \) appears in more than one function. In fact, the cost function subject to (1.10b) is a separable problem, but the constraints (1.10c) cause coupling thus breaking the separability of (1.10).

The Lagrangian of (1.10) introduces dual variables \( \lambda \geq 0 \) and \( \mu \):
\[
\mathcal{L}(z, \lambda, \mu) = \sum_{i=1}^{r} [J_{i}(z_{i}) + \lambda_{i}^{T}(A_{i}z_{i} - b_{i}) + \mu_{i}^{T}(C_{i}z_{N(i)} - d_{i})]
\] (1.11)
Note that the Lagrangian is separable over $z_i$. The projected gradient method is then applied to the dual problem, resulting in the following iterative computations, beginning with an initial guess for the dual variables $\lambda^0$ and $\mu^0$:

\[
\begin{align*}
    z_{i}^{t+1} &= \arg \min_{z_i} \mathcal{L}(z_i, \lambda^t, \mu^t) \quad (1.12a) \\
    \lambda_i^{t+1} &= ((\lambda_i^t + \alpha(A_i z_{i}^{t+1} - b_i))_+) \quad (1.12b) \\
    \mu_i^{t+1} &= \mu_i^t + \alpha(C_i z_{N(i)}^{t+1} - d_i)) \quad (1.12c)
\end{align*}
\]

where the $t$ superscript is the iteration number, $(\cdot)_+$ is the projection onto nonnegative real numbers $\mathbb{R}_+$ and $\alpha$ is a step size parameter. Note that the $z$ update (1.12a) can be computed in parallel; that is $z_i$ can be updated independently of $z_{j \neq i}$ due to the separability of the Lagrangian (1.11). Additionally, each element of each dual variable $\lambda$ and $\mu$ (1.12b), (1.12c) can be updated in parallel as it only relies on its previous iterate. Thus, the steps of the projected gradient method are highly parallelizable.

Although the steps of dual decomposition are trivially parallelizable, there is non-negligible communication required between steps if the algorithm is implemented on distributed platforms. The $z$ updates (1.12a) are parallelizable but the update of any element of the dual variables $\lambda$ and $\mu$ generally depends on the full vector $z$. Thus, the $z$ update can occur in parallel, but $z_i$ must be gathered into the full vector $z$ before the dual update steps. Additionally, the $z$ vector must be scattered to the processors that update the dual vectors. Similarly, the dual variable steps are trivially parallelizable, but between the dual variable update steps (1.12b), (1.12c) and the $z$ update (1.12a), the dual variables $\lambda$ and $\mu$ must be similarly gathered and scattered. In some cases, the $z$ update may be centralized so that one set of gather and scatter communications may be avoided; similarly, the $\lambda$ and $\mu$ updates may alternatively be centralized. This decision to centralize some steps of the algorithm is highly dependent on the optimization problem’s physical context.

The clear advantages of dual decomposition are that the algorithm is simple to implement and is trivially parallelizable. In many cases, the dual variables $\lambda$ and $\mu$ are described as prices for the constraints (1.10b), (1.10c) and thus there is an intuitive price based interpretation (see [25, 17] for nice explanations of this idea). The convergence of the dual decomposition algorithm is considerably important in studying this problem. In general, the convergence of the projected gradient method is $O(1/k)$ for a constant step size $\alpha$ [22] but can be accelerated to $O(1/k^2)$ by choice of step size originally developed by Nesterov [24]. The constant step size rule requires that the first derivative of the cost function (1.10a) is Lipschitz continuous in order for convergence to be guaranteed [22], and the linear rate of convergence requires strict convexity of the cost function local to the optimizer [26]. Similar assumptions must be made to achieve $O(1/k^2)$ convergence.

### Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers is similar to dual decomposition except that it does not require as many assumptions in order to converge [17]. Most notably, the
main assumption for ADMM to converge is that the cost function must be convex and the Lagrangian must have a saddle point [17]. This is already significantly looser than the assumptions required to prove the convergence of dual decomposition.

The steps of ADMM can be formulated as follows. First, problem (1.10) is modified to the following form:

\[
\min_{z,y} \sum_{i=1}^{r} J_i(z_i) + I_C(z) + I_A(y) \quad (1.13a)
\]

subject to

\[
z = y \quad (1.13b)
\]

where the constraints have been encoded in indicator functions \( I_A(z) \) and \( I_C(x) \), defined as follows:

\[
I_A(y) = \begin{cases} 
0 & \text{if } A_i z_i \leq b_i \forall i \in \{1, \ldots, r\} \\
\infty & \text{otherwise,}
\end{cases} \quad (1.14)
\]

\[
I_C(z) = \begin{cases} 
0 & \text{if } C_i z_N(i) - d_i \forall i \in \{1, \ldots, r\} \\
\infty & \text{otherwise.}
\end{cases} \quad (1.15)
\]

ADMM is then similar to dual decomposition, except that it is based on the augmented Lagrangian of (1.13):

\[
\mathcal{L}_\rho(z, y, \lambda) = \sum_{i=1}^{r} J_i(z_i) + I_C(z) + I_A(y) + \lambda^T(z - y) + \frac{\rho}{2} \| z - y \|^2. \quad (1.16)
\]

Note that the \( \lambda \) is a dual variable for the consistency constraint (1.13b), and is different than the \( \lambda \) defined for dual decomposition. Instead of one primal update step (1.12a), ADMM has an \( z \) update and a \( y \) update before updating the dual variable \( \lambda \):

\[
z^{t+1} = \arg\min_{z} \mathcal{L}_\rho(z^{t}, y^{t}, \lambda^t) \quad (1.17a)
\]

\[
y^{t+1} = \arg\min_{z} \mathcal{L}_\rho(z^{t+1}, y^{t}, \lambda^t) \quad (1.17b)
\]

\[
\lambda^{t+1} = \lambda^t + \rho(z^{t+1} - y^{t+1}) \quad (1.17c)
\]

The last step (1.17c) is also known as a Gauss-Seidel pass. This particular choice of splitting \( z \) and \( y \) becomes quite clear when applying ADMM to a quadratic program. Define the cost function to be a sum of quadratic functions:

\[
J(z) = \sum_{i=1}^{r} J_i(z_i) = \sum_{i=1}^{r} z_i^T Q_i z_i \quad (1.18)
\]

Then the \( z \) update step (1.17a) reduces to a linear equation:

\[
\begin{bmatrix} \nabla^2 f + \rho I & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} z^{t+1} \\ \nu^{t+1} \end{bmatrix} = \begin{bmatrix} -\lambda^t + \rho y^t \\ d \end{bmatrix}. \quad (1.19)
\]
where \( d = [d_1^T \ldots d_r^T]^T \) and \( C \) is made up of \( C_i \) such that \( C_i z_N(i) = d_i \) is written as \( C z = d \). The \( y \) update step (1.17b) is a projection onto the polyhedron \( \{ y | Ay \leq b \} \):

\[
y^{t+1} = \text{proj}_{Ay \leq b} (z^{t+1} + \frac{1}{\rho} \lambda^t).
\]

(1.20)

where \( Ay \leq b \) is lumped inequality constraint of \( A_i y_i \leq b_i \forall i \in \{1, \ldots, r\} \). This projection step can be computed in parallel because the inequality constraints are not coupled. This projection step is particularly easy to compute for some sets, such as box constraints. If the polyhedron \( \{ y_i | A_i y_i \leq b_i \} \) is a box constraint set, then every element of the \( y_i \) update can be done in parallel. The \( z \) update step (1.17a) is not generally parallelizable, however.

Similar to dual decomposition, there is non-negligible communication between steps of the ADMM algorithm (1.17). Between the \( z \) and \( y \) updates, the \( z \) vector must be scattered to the distributed processors that update \( y \). The \( y \) updates can be done on parallel processors, but must be gathered and scattered before the \( \lambda \) update step, unless the \( \lambda \) updates are split the same way as \( y \). Then, the \( \lambda \) update must be gathered before going back to the \( z \) update.

The Alternating Direction Method of Multipliers can be slow to converge to high accuracy in practice. In fact, only under certain conditions can local linear convergence be proved [27] for the ADMM form presented in steps (1.17). However, a common suggestion for implementation of ADMM in practice is to terminate the algorithm early in order to quickly obtain a solution that is not highly accurate [17]. This is suggested especially if the optimization problem does not require a highly accurate solution; this early termination idea appears to work well in practice for some Model Predictive Control problems [28].

Selection of the parameter \( \rho \) also has an impact upon the performance of the ADMM algorithm. Fortunately, there have been some studies of the optimal parameter \( \rho \) for specific applications of the ADMM algorithm (1.17) applied to MPC problems with quadratic cost and linear constraints [29], [30] when the equality constraints have been eliminated by substitution. However, the parameter choice is not well understood for all forms of ADMM.

### Distributed Optimization in Practice

The two methods just described, dual decomposition and ADMM, are probably the most popular distributed optimization algorithms for DMPC, but many other algorithms exist in the literature. The Alternating Minimization Algorithm [18] has been studied extensively for DMPC, with interesting results in bounding the number of iterations to algorithm termination [31], and using inexact computations and/or communications [32]. Other distributed optimization methods include variants on the Gauss-Seidel method [33], and inexact Newton methods [34], [35].

Distributed optimization solutions have been suggested for model predictive control problems of various sizes. There are some distributed optimization solvers that are written for embedded platforms such as FPGA (field-programmable gate array) [36] and other embedded systems with limited memory [37, 38]. For large-scale control problems, it is often alluded to use the distributed platforms that are installed with such large-scale systems. However,
to the best of the author’s knowledge, the implementation of these distributed algorithms on these existing distributed platforms has not been done. In fact, most large-scale control problems are programmed in a centralized manner, even if using a distributed optimization algorithm. This is key motivation for the research done in this thesis – how well can distributed optimization algorithms perform on existing embedded systems for large scale systems? This question is investigated in depth for the application of building control systems. The results show that dual decomposition and ADMM are prohibitively slow upon implementation on embedded systems used for building control.

In order to improve performance of distributed MPC solutions on existing embedded control systems, a novel distributed algorithm is proposed in Chapter 4. This algorithm is based on a primal-dual active-set method and exhibits very fast convergence for MPC problems with quadratic cost and linear constraints. This distributed algorithm is shown to perform very well on systems with high communication delay. We show that the active-set method is able to perform in real-time on very slow communication networks while the dual decomposition and ADMM algorithms are considerably slow.

The second main issue regarding most distributed optimization solutions is that they do not have any convergence guarantees for general non-convex nonlinear programs. Therefore, a second novel distributed algorithm is proposed in Chapter 7 to address this problem. This method is based on an inexact interior point method and can be applied to general nonlinear optimization problems, including non-convex optimization problems. We show how the algorithm is able to be implemented in a distributed way by using iterative linear algebra methods. The algorithm has convergence guarantees under some mild assumptions. To the best of our knowledge, there does not exist another distributed algorithm that is guaranteed to converge for nonlinear non-convex optimization problems under such mild assumptions.

In this thesis, these novel distributed optimization algorithms are applied to the building control problem and the ramp metering control problem. The next two sections introduce the motivation for the study of these two applications.

### 1.3 Building Thermal Control

The building sector is the largest energy consumer in the world. The building sector consumes about 40% of the energy used in the United States and is responsible for nearly 40% of greenhouse gas emissions [39]. Therefore, it is economically and environmentally significant to reduce the energy consumption of buildings.

In buildings, performance improvement using forecasted information is possible due to two basic mechanisms. The first mechanism is referred to as load shifting or active storage. Load shifting consists of shaping the energy profile delivered to a building, exploiting the possibility of storing energy for later use. Thermal storage is inherent to building structures with thermal capacity and can be increased by including additional external energy storage devices. The optimal profile of delivered energy depends on various factors including time...
varying utility prices, availability of renewable energy, ambient temperature variation, and load shedding signals received from the utility grid. The second mechanism is component optimization. Buildings can be large systems with many control variables and degrees of freedom. Predictive models of building thermal dynamics and energy costs of control actuators allow computation of the optimal inputs to each actuator in order to deliver the desired energy profile in the most cost-effective way. These two mechanisms can be automatically combined in an MPC control strategy. Model predictive control [40] (MPC) is a control methodology that can systematically use predictions of weather, occupancy, renewable energy availability, and energy price signals [41], [42] to improve building thermal comfort, decrease peak demand, and reduce total energy costs.

In [43], [44], [2], [45], the authors have implemented MPC on building heating, ventilation, and air conditioning (HVAC) systems obtaining reduced peak power consumption and reduced overall energy usage compared to existing production logic. In the aforementioned literature, control computation is performed at a centralized unit. In practice, embedded building control platforms are physically distributed throughout the building. It is standard to have a central processor and router located at the main air supply location (called the Air Handling Unit) and an embedded control unit at the locations which regulate flow and temperature of each zone (called the Variable Air Volume box). Chapter 2 reviews in detail how building control systems are implemented in practice.

Chapter 3 reviews the state of the art control design for energy efficient buildings. The state-of-the-art solution for building control is called “Trim and Respond” and is a distributed control method. We discuss how the method is implemented on building control systems and propose a one-step distributed MPC solution which is similar to “Trim and Respond”. This DMPC method is able to fit the existing distributed control platforms in a similar manner. Moreover, the performance of the DMPC method matches the performance of “Trim and Respond”. Chapter 4 introduces a novel distributed primal-dual active-set method which matches the one-step distributed MPC solution. This novel method is able to solve $N$-step distributed MPC problems on the existing distributed control platforms.

### 1.4 Ramp Metering Control

Recently, traffic congestion has been increasing considerably [46]. In the United States, cities of all sizes have significant increases in travel delay over the past 20+ years, with the average hours of delay per traveler more than doubling from 1982 to 2002 [47]. Reducing traffic congestion not only saves drivers transportation time, but it also leads to less vehicle emissions and saves fuel. Thus, development of traffic management and control tools is essential to mitigate congestion and enable efficient utilization of the current infrastructure of metropolitan areas.

There are two types of traffic networks: freeways and urban arterials, with each type having its own model of traffic progression. In this thesis, we study the synthesis of control strategies that could be incorporated for ameliorating traffic conditions in freeways. In
freeways, the inputs available to the control engineer include ramp metering and variable speed limits. Ramp metering refers to the determination of the amount of flow entering the freeway through on-ramps; variable speed limits are a mechanism of assigning speed limits dynamically such that congestion occurrence is postponed or avoided.

Ramp metering has been shown to be capable of reducing delays in freeways \cite{48}; hence it is an effective strategy for increasing throughput of the network. As a result of this effectiveness, several ramp metering strategies have been developed ranging from fixed–time controllers \cite{48} to controller synthesis from signal temporal logic specifications \cite{49} to synthesis of robust state-feedback controllers \cite{50}. Among possible approaches for ramp metering, model-predictive-based controllers that optimize some performance metric are favored. In Chapter 5, we summarize how ramp metering systems are designed in practice, from the system architecture to the control algorithms used on highways across the globe.

Although model predictive control is a favored option for ramp metering control, it is generally not implemented with persistent feasibility and stability guarantees. Thus, in Chapter 6, we formulate the ramp metering problem as a hybrid model predictive control problem and apply stable hybrid MPC design methods to the problem. This design guarantees persistent feasibility and stability for the ramp metering problem. The design is implemented and simulated for a small freeway example in Chapter 6, but the ramp metering problem is in general a large-scale control problem. Thus, in Chapter 7, we study an inexact interior point method which can be applied to an approximated version of the hybrid model predictive control problem studied in Chapter 6. The inexact interior point method has global convergence guarantees and can be implemented in a distributed fashion.

1.5 Thesis Layout

This section summarizes the main research contributions presented and outlines the organization of the thesis.

Chapter 2 focuses on control system design of forced-air systems. The chapter describes the currently available computational platforms and communication systems for forced air heating, ventilation, and air conditioning systems. A brief overview of the design of model predictive controllers for energy efficient forced-air systems is provided. This chapter sets the stage for the next two chapters. The writing in this chapter is to appear in a book chapter:


In Chapter 3, we study the distributed control of heating, ventilation, and air conditioning systems in buildings. We present and compare two control design techniques. In the first part, a one-step distributed model predictive controller (DMPC) is introduced. We compute the explicit state-feedback solution and show how to implement it on existing networked buildings control platforms. In the second part of the chapter, we discuss a control logic
currently used in the buildings industry called “Trim and Respond”. We show that “Trim and Respond” controllers can be seen as a special class of one-step DMPC algorithms. We conclude the chapter with a simulation study which shows the DMPC control design and its equivalence with “Trim and Respond” heuristics. The work discussed in this chapter has previously appeared in the following publication:


In Chapter 4, we present a novel distributed primal-dual active-set method for Model Predictive Control (MPC). The primal-dual active-set method is used for solving MPC problems for large-scale systems with quadratic cost, linear dynamics, additive disturbance, and box constraints. The proposed algorithm is compared to dual decomposition and an Alternating Direction Method of Multipliers (ADMM). Theoretical and experimental results show the effectiveness of the proposed approach for large-scale systems with communication delays. The application to buildings control systems is thoroughly investigated. This work is previously published in:


In Chapter 5, we shift our focus to the control of ramp metering systems. The Asymmetric Cell Transmission Model (ACTM) models the dynamics of a freeway onramp metering system. The sensors, computational platforms, and communication choices in real ramp metering systems are summarized. We review some of the proposed ramp metering control algorithms used in practice. Model predictive control design of ramp metering systems is discussed with a brief summary of distributed model predictive control methods proposed in the literature.

In Chapter 6 we synthesize a hybrid Model Predictive Controller for the piecewise affine model such that persistent feasibility and stability are guaranteed. We do so by designing a terminal constraint and terminal cost for the equilibrium state of the system. We show how to formulate ACTM as a piecewise affine system defined over regions of the state and input space. We include a detailed analysis of the equilibrium point of the piecewise affine system, and define the region of demand under which an equilibrium point exists. We present an extensive comparison of the proposed hybrid MPC with the commonly used relaxed LP controller formulation. We show that our method achieves the same performance in terms of efficiency and exhibits much smoother behavior than that of the relaxed design. In addition, our approach has stability and feasibility guarantees whereas the relaxed controller does not. This work is published in:

Finally, in Chapter 7, we propose a framework for a distributed inexact interior point method. The inexact interior point method has guaranteed global convergence for smooth nonlinear optimization problems under some mild assumptions. The utility of the inexact method is that the main computational bottleneck of the algorithm – the numerical linear algebra used to solve the Karush-Kuhn-Tucker system of equations – can be solved by using an iterative linear algebra method. Iterative linear algebra methods are trivially parallelizable, making this a scalable algorithm. The algorithm is tested in serial, but a framework for computing the algorithm in parallel including several state-of-the-art code libraries is proposed. The algorithm is tested on a smoothed version of the ramp metering problem discussed in Chapter 6.
Chapter 2

Building Forced Air Systems

The objective of this chapter is to describe the way that building forced air systems work in practice. We describe both how heat transfer is delivered via the HVAC components and how the embedded control and communication platforms are installed to enable control of the building. This in depth understanding of the building automation system (BAS), meaning the underlying hardware and software systems of the building, will inform and significantly impact the design of distributed model predictive controllers for energy efficient buildings.

First, this chapter details the physical components of a typical forced air heating, ventilation, and air conditioning system. A gray box thermal model of the components’ effect on building temperature is derived from heat transfer principles.

Next, we describe how the building automation system (BAS) is designed in practice for control. In particular, we describe how embedded control platforms and communication systems are physically laid out in practice. The connections of these computational platforms are described in relation to the HVAC components as well as in relation to other computational and communication platforms in the building. It is standard to have a central processor and router located at the main air supply location (called the Air Handling Unit) and an embedded control unit at the locations which regulate flow and temperature of each zone (called the Variable Air Volume box).

Then, current building control practices are summarized. We highlight the limitations presented by the BAS systems used in practice; there exists limited computational power and communication speeds that are present in today’s buildings.

The chapter concludes with model predictive control design for forced air systems and discusses the challenges of implementing distributed model predictive controllers on the BAS described in this chapter.

2.1 Forced Air Systems

There are many approaches to building modeling, as seen in the literature [2, 56, 42, 57, 58]. These models range in granularity and complexity, from the high-level building envelope
down to heating, ventilation, and air conditioning (HVAC) system component modeling. This includes white box modeling such as that seen in Energy Plus [59] or Modelica [60] to black box modeling. We describe the components of forced air systems and review the thermal models that we use for model predictive control. Our thermal models are derived from physics and simplified for control purposes, making them grey box models.

**HVAC System Components**

This section introduces the components of a typical forced air heating, ventilation, and air conditioning system. Air Handling Units (AHU) and Variable Air Volume (VAV) boxes are the main components used to heat or cool and distribute air in a building (Figure 2.1). The AHU recirculates return air from building spaces, and mixes it with fresh outside air. The proportion of return air to outside air is controlled by damper positions in the AHU (Figure 2.1). The mixed air can be cooled by cooling coils that extract the cooling energy from chilled water produced by chillers.

![Figure 2.1: System Scheme](image)

The air temperature after these coils depends on the coil valve positions, the temperature of the chilled or heated water, the temperature of mixed air entering the cooling coil, the mass flow rate of the mixed air, and the physical characteristics as well as thermal effectiveness of the coils. Air is delivered to the building spaces by electrical fans. Before reaching a given space, the air goes through VAV boxes. At each VAV box, air temperature can be increased using reheat coils installed in the VAV box when needed. The space served by one VAV
A study on the effect of discretization methods in MPC can be found in [56].
It is assumed that in the AHU, the outside air fully mixes with the return air without delay, and the mixing proportion \( \delta \) between the return air and outside air is controlled by the damper configurations in the AHU system to obtain:

\[
T_m = \delta T_r + (1 - \delta)T_{oa},
\]

where \( T_{oa} \) is the outside air temperature, and \( T_m \) is the temperature of the mixed air. \( T_r \) is the return air temperature calculated as weighted average temperature of return air from each room

\[
T_r = \frac{\sum_{i \in I} \dot{m}_i T_i}{\sum_{i \in I} \dot{m}_i},
\]

where \( I \) is the set of indices for all zones in the building. The return air is not recirculated when \( \delta = 0 \), and no outside fresh air is used when \( \delta = 1 \). \( \delta \) can be used to save energy through recirculation but it has to be strictly less than one to guarantee a minimal outdoor fresh air delivered to the rooms.

The supply air temperature then is computed as

\[
T_{s,i} = T_m - \Delta T_c + \Delta T_{h,i}, \quad \forall i \in I,
\]

where \( \Delta T_c \) represents the supply air temperature difference across the cooling coil in the AHU, and \( \Delta T_{h,i} \) is the air temperature difference across the heating coil in the \( i \)-th VAV box. The cold water in the cooling coil is typically provided by a chiller system whereas the hot water in the heating coil is provided by a boiler system. The colder the cooling coil supply water is, the larger \( \Delta T_c \) can be, and likewise, the hotter the heating coil supply water is, the larger \( \Delta T_{h,i} \) can be.

The decomposition (2.2) matches equations (2.5)-(2.8). Substituting (2.8) into (2.5), the power input is expressed as:

\[
u_i = \dot{m}_i c_p (T_m - \Delta T_c + \Delta T_{h,i} - T_i).
\]

If we assume that \( T_m \approx T_i \), then it is clear that \( u_i \) has two parts:

\[
u_i = -\dot{m}_i c_p \Delta T_c + \dot{m}_i c_p \Delta T_{h,i}.
\]

The approximation \( T_m \approx T_i \) eliminates coupling between \( T_i \) of different zones. The difference between \( T_m \) and \( T_i \) can be treated as part of the disturbance term \( \dot{Q}_{d,i} \). Then, \( u_c \) and \( u_{h,i} \) are computed accordingly as:

\[
k_{c,i} u_c = -\dot{m}_i c_p \Delta T_c
\]
\[
u_{h,i} = \dot{m}_i c_p \Delta T_{h,i}.
\]

Let \( u_c = -\dot{m}_{max} c_p \Delta T_c \) where \( \dot{m}_{max} = \max_i (\dot{m}_i) \). Then \( k_{c,i} = \frac{\dot{m}_i}{\dot{m}_{max}} \).

For simplicity, we consider the control of \( \Delta T_c \) and \( \Delta T_{h,i} \) where the mass flow rate \( \dot{m}_i \) is controlled separately or is otherwise approximated as constant. This approximation may
CHAPTER 2. BUILDING FORCED AIR SYSTEMS

lead to a linear time varying model if $\dot{m}_i$ is not constant (i.e. $B_{c,i}$ in (2.4) is not time invariant). In many cases, the approximation of a constant $\dot{m}_i$ is sufficient.

If $\dot{m}_i$ is also controllable, the system dynamics (2.3) are bilinear. The Euler forward discretization method can also be applied with equations (2.5)-(2.8) substituted into (2.1), obtaining nonlinear discrete time system dynamics.

2.2 Building Automation Systems

Building automation systems (BASs) are large, complex, distributed systems. The building automation system is responsible for building management including control of the HVAC system, lighting system, and fire alarm system. In this section, we outline the embedded platforms and communication systems that are used for HVAC control in real buildings today.

Figure 2.2: Typical layout of building embedded platforms and routers. Physical components and connections are denoted by solid lines; control platforms and wired connections are denoted by dashed lines; routers and communication backbone are denoted by purple dotted lines.
Control Platforms

In practice, embedded building control platforms are physically distributed throughout the building. In a commercial building, the HVAC typically includes a chiller plant, a boiler plant, AHU(s), and VAVs. Figure 2.1 shows a single AHU. Figure 2.2 generalizes Figure 2.1 to represent one or more chiller plants, multiple AHUs, VAVs. In practice, there is an embedded controller at each VAV, each AHU, and the chiller plant. The larger and more complex the component or subsystem, the larger the corresponding controller; a common installation uses a small zone controller at each VAV box, a midsize controller at each AHU, and a more powerful control platform at the chiller plant. For instance, for the Automated Logic Corporation (ALC) vendor [61], The “ZN” controller is often used at the VAV or “zone controller” (ZC) level, the “SE” controller at the AHU or “equipment controller” (EC) level, and the “ME” controller at the chiller plant or “plant controller” (PC) level, as illustrated in Figure 2.2.

At the VAV box, the zone controller “ZC” has a small processor with limited memory and computational power. A typical “ZC” has 512 kB non-volatile memory, 1 MB Flash memory, and a 16-bit processor [62]. This processor typically handles control of the VAV box (damper position and heating coil valve position $\delta_{h,i}$) as well as reading the sensor information in the VAV box (supply temperature $T_{s,i}$, mass flow rate $\dot{m}_i$) and the zone temperature $T$.

At the AHU level, the equipment controller “EC” reads AHU sensor information such as return air temperature $T_r$, outside air temperature $T_{oa}$, and mixed air temperature $T_m$. The controller also actuates the AHU controllable inputs, such as cooling coil valve position $\delta_c$, the supply fan speed, and the mixing ratio $\delta$ provided by the dampers.

Finally, at the plant level, the plant controller “PC” is used to control multiple independent components that must work together to form a system, such as pumps, cooling towers, chillers, etc. It reads information such as the chilled water supply and return temperatures, and controls the chillers (which run either a vapor-compression or absorption refrigeration cycle) as well as the pump that pushes chilled water to the AHUs and cooling towers or other heat rejection equipment that dump heat to the ambient environment.

Despite trends in computing such as Moore’s law which has enabled the rapid advancement of computers and mobile devices, it is unlikely that the existing processors in buildings will change much over time. Hardware elements that are used to implement the controls systems are not changed over the lifetime of a building and even new buildings are installed with these embedded systems. Thus, it is desirable to fit control algorithms on these processors.

Communication Systems

Each building comes with its own unique set of embedded platforms and networking. Separate devices in the same building may be provided by different vendors, each of which is packaged with its own embedded platform and software. In order to communicate amongst these devices, a buildings specific communication protocol is used. The standard data communication protocol for building control systems is known as BACnet, or “Building Automation
Control Network” [63]. BACnet was developed by ASHRAE (American Society of Heating Refrigerating and Air-conditioning Engineers) and is both an ANSI (American National Standards Institute) and ISO (International Organization for Standardization) standard.

BACnet provides an object-oriented model abstraction for devices to talk to each other over various types of network. The choice of network (both the physical wiring as well as the accompanying lower level communication protocol) has a significant effect on the overall performance of the communications. A simulation analysis of the most popular networking options used in buildings (MS/TP, Ethernet, and ARCNET) is done in [64]. In general, Ethernet is the fastest option, but is also the most expensive. On the other end, MS/TP is the cheapest option but is also the slowest. ARCNET however is only a little more expensive than MS/TP, but is noticeably faster [65]. The authors of this chapter have most commonly encountered ARCNET as the choice of networking.

The communication system of a large-scale forced-air system is pictured in Figure 2.2. A “BR” building router is connected to the “PC” embedded platform at the chiller plant [61]. This router can communicate with external servers such as cloud servers. The “PC” platform is connected to a daisy chain of the “EC” controllers at the AHUs. The zone controllers are then connected in a daisy chain to one of the “EC” control platforms directly or via a subnetwork router or equipment router “ER” such as the Arcnet to Arcnet Router (AAR) [66]. In very large buildings, multiple subnetwork routers may be used to split up the network; each AAR can handle up to 100 zone controllers [66].

Generally, point-to-point communication between embedded platforms is possible, but in practice, communication is done in a hierarchical fashion. That is, platforms communicate over a star communication network, such as that in Figure 4.2. The zone controllers “ZC” communicate with an “EC” controller, and the “EC” controllers communicate with the “PC” controller. On the uplink, devices may communicate by using BACnet services such as ReadProperty or ReadPropertyMultiple. These services are akin to receive and gather services seen in the Message Passing Interface (MPI) standard, respectively. The operations receive or ReadProperty allow the central hub to read a single data point from a single leaf node. The gather or ReadPropertyMultiple operation takes vector pieces from the leaf nodes and aggregates them into one large vector at the central hub. On the downlink, devices may communicate by using BACnet services WriteProperty or WritePropertyMultiple which are akin to send or scatter in MPI terminology. The send or WriteProperty operation sends a single data point from the central hub to a leaf node. A scatter or WritePropertyMultiple operation takes a vector at the central hub and sends each piece of that vector to the appropriate leaf node.

It is worth highlighting that the communication systems present in buildings are slow. We have found that in practice, ReadPropertyMultiple on an ARCNET network results in a communication delay of 0.03 to 0.4 s per floating point value. The simulation results in [64] state similar values for communication delay.
CHAPTER 2. BUILDING FORCED AIR SYSTEMS

2.3 Current Building Control Practices

Due to the large distributed nature of BASs, the most common implementation in practice is to decouple the building controls into separate single-input-single-output proportional-integral-derivative (PID) loops to control variables to track desired fixed setpoints [67, 2]. These setpoints include duct static pressure, AHU supply temperature, VAV supply temperature, etc.

The PID controllers provide setpoint tracking, but do not promote energy efficiency. A more sophisticated building control method which is used in practice is “Trim and Respond” [67, 68]. This method differs from the PID loops because it changes or resets setpoints over time. The main idea is to modify setpoints by responding to heating or cooling requests and trimming energy consumption. The actions of trimming and responding are enabled by communicating information about demand from the zone level processors to the centralized AHU processor. “Trim and Respond” logic is described in detail in the next chapter. The logic works well in practice and is used in efficient building control guidelines [67].

2.4 Model Predictive Control

Model Predictive Control (MPC) has the ability to utilize system models and disturbance forecasts to automatically synthesize an efficient building controller [41, 2, 44, 45]. Sensor feedback is used to inform the system about the current state, capturing unexpected temperature disturbances as well as model errors. Energy price signals and forecasts change from day to day, requiring controller flexibility on at least a day-to-day basis. Model predictive control (MPC) systematically incorporates models, forecasts, and feedback to flexibly and efficiently handle building thermal control.

In this section, a model predictive controller is designed to minimize an objective (e.g., total energy consumption of the HVAC system), and to satisfy constraints such as thermal comfort constraints and actuation constraints. A model predictive controller solves a constrained finite time optimal control (FTOC) problem at each time step in a receding horizon.
Consider the following FTOC problem at time step $t$:

\begin{align*}
\min_{X,U} & \quad J(X,U) \\
\text{s.t.} & \quad x(k + 1) = f(x(k), u(k), \hat{d}(k)) \quad (2.13a) \\
& \quad x(k) \in X, \ k = 1, \ldots, N \quad (2.13b) \\
& \quad u(k) \in U, \ k = 0, \ldots, N - 1 \quad (2.13c) \\
& \quad x(0) = x|_{t} \quad (2.13d)
\end{align*}

where the state at time step $k$ is $x(k) \in \mathcal{R}^n$ with $X = [x(0)^T, \ldots, x(N)^T]^T$, the input at time step $k$ is $u(k) \in \mathcal{R}^m$ with $U = [u(0)^T, \ldots, u(N - 1)^T]^T$, $\hat{d}(k)$ is the predicted load at time step $k$, the prediction horizon length is $N$, the dynamics are encoded by $f(\cdot)$, $X$ is a polyhedron encoding state constraints such as thermal comfort constraints, $U$ is a polyhedron encoding actuator constraints, and the current state feedback $x|_{t}$ is used.

Let the optimal solution to problem (4.8) at time step $t$ be

\[ U^* = [u^*(0)^T, u^*(1)^T, \ldots, u^*(n)^T]^T. \quad (2.14) \]

At time step $t$, the solution $u^*(0)$ is implemented. The optimization (4.8) is repeated at time $t + 1$, with the updated state estimation $x|_{t+1}$ and estimated load, yielding a receding horizon control strategy.

**Cost Design**

The objective of the controller is directly encoded into the cost function $J(X,U)$ of the MPC problem (4.8). This section describes typical cost functions that are used for building thermal control. We describe the cost functions for power inputs, i.e. $u(k) = [u_1(k), \ldots, u_n(k)]$ where $u_i(k)$ is the input defined in (2.2) at time step $k$.

First, the objective of minimal total energy consumption is encoded by integrating the power inputs over the prediction horizon:

\[ J_{\text{energy}}(X,U) = \sum_{k=0}^{N-1} |u(k)| \Delta t. \quad (2.15) \]

The total energy may also be minimized by expressing the cost function as a summation of the HVAC component power consumption equations such as the fan power, cooling coil power, heating coil power, etc. Detailed equations of component power consumption are included in [2].

Electrical pricing can be accounted for in a similar fashion:

\[ J_{\text{price}}(X,U) = \sum_{k=0}^{N-1} c_k(u(k)) \quad (2.16) \]
where the cost function $c_k(\cdot)$ in general varies over time. This allows the controller to plan a control trajectory that can take advantage of lower cost periods and lessen power consumption at periods of higher cost. Similarly, the peak power usage of the day can be optimized by minimizing the following cost function:

$$J_{\text{peak}}(X,U) = \|U\|_\infty = \max(u(0), \ldots, u(N-1)).$$  \hfill (2.17)

The infinity norm is used to indicate the largest magnitude power input over the prediction horizon. The usage of $J_{\text{peak}}(X,U)$ encourages precooling, or shifting the energy consumption to be before a high thermal load period, rather than during such a high thermal load period.

For an illustrative example of the effect of the various cost functions discussed here, see [2]. Note that the prediction horizon $N$ affects the controller’s ability to look ahead into the future. For the majority of the cost functions described here, the prediction horizon should be long, covering as much of a day as possible in order to accomplish pre-cooling, load shifting, etc.

In general, a weighted combination of cost functions (2.16)-(2.17) may be used. The inputs may be penalized by a 2-norm cost function rather than a 1 or $\infty$-norm cost function. For example,

$$J(X,U) = p(x(N)) + \sum_{k=0}^{N-1} q(x(k), u(k))$$  \hfill (2.18)

where the stage cost $q(x(k), u(k))$ is defined by the following in the 2-norm case

$$q(x(k), u(k)) = (x(k) - x_{ss})^T Q (x(k) - x_{ss}) + u(k)^T R u(k)$$ \hfill (2.19)

with $Q \succeq 0, R \succ 0$, and $x_{ss}$ is the desired steady state for $x(k)$. The desired $x_{ss}$ is the desired zone temperature setpoint. The terminal cost $p(x(N))$ is defined by the following in the 2-norm case

$$p(x(N)) = (x(N) - x_{ss})^T P (x(N) - x_{ss})$$ \hfill (2.20)

with $P \succ 0$. The 2-norm cost functions are popularly used in MPC design, especially for the design of an MPC controller with stability guarantees [7].

**Constraints**

While the controller optimizes an objective such as energy consumption, it must also obey constraints such as maintaining comfort conditions for occupants. Occupancy comfort is modeled as a band of temperature in which the zone temperature should remain [69]. That is, the zone temperature is constrained by an upper bound $\bar{T}$ and a lower bound $\underline{T}$:

$$x_i \in [\underline{T}, \bar{T}].$$  \hfill (2.21)
This constraint (2.21) would define the state constraint $X$. In some cases, these bounds on zone temperature are implemented as soft constraints since it is acceptable to have some constraint violation. See more details on adding soft constraints into the MPC design in [2].

In addition to state constraints, there are also constraints on the control inputs. The control inputs are physically constrained by saturation limits. For example, the power input $u_i$ is limited by upper and lower bounds:

$$u_i \in [\underline{u}, \bar{u}].$$

(2.22)

The constraints on low level control inputs of a forced air system in equations (2.5)-(2.8) include:

- $\dot{m}_i \in [\dot{m}_i, \bar{m}_i]$: The mass flow rate must be above minimum ventilation requirements and cannot exceed maximum flow capacity
- $\delta \in [0, 1]$: The damper position ranges between completely open and completely closed
- $\Delta T_c \in [0, \Delta T_c]$: The cooling coil may cool the air up to a maximum amount
- $\Delta T_h \in [0, \Delta T_h]$: The heating coil may heat the air up to a maximum amount

These constraints are encoded in $U$, the polyhedron describing input constraints.

In some cases, the design of a terminal constraint such that $x_N \in X_f$ is added to the MPC problem (4.8) for persistent feasibility and stability, as briefly discussed in the Introduction of this thesis and as discussed in detail in [7].

2.5 Distributed Model Predictive Control for Forced Air Systems

In most demonstrations of MPC on building heating, ventilation, and air conditioning (HVAC) systems, control computation is performed at a centralized unit [43], [44], [2], [45]. The size of the centralized predictive control problem rapidly grows when a realistic number of rooms together with a meaningful control horizon are considered. Therefore the real-time implementation of an MPC scheme is a challenge for the low-cost embedded platforms and low-speed communication networks currently used for HVAC control algorithms.

In the following chapters, we present algorithms for distributed model-based predictive control (DMPC) for regulating heating and cooling in order to minimize energy consumption while satisfying comfort constraints. These distributed MPC control algorithms fit the aforementioned existing distributed control platforms.

In Chapter 3, we propose a distributed solution that is the explicit solution of a one-step (i.e. prediction horizon $N = 1$) MPC problem. We show that this explicit solution matches both the control logic and the distributed form of “Trim and Respond”.

This special class of DMPC is nice because it fits the limited computation and communication requirements of real building systems. However, this special class is limited since it does not include a long prediction horizon. In Chapter 4, we show how to extend this distributed solution for a general prediction horizon by using a primal-dual active-set optimization method. We compare the primal-dual active-set method to other popular distributed optimization methods, dual decomposition and the alternating direction method of multipliers (ADMM). We show that these popular methods for large-scale MPC cannot perform well on real building computing and communication systems whereas the primal-dual active-set method can.
Chapter 3

Building Temperature Distributed Control via Explicit MPC and “Trim and Respond” Methods

3.1 Introduction

In this chapter, we present and compare two distributed control design approaches: a one-step distributed MPC and a heuristic control logic called “Trim and Respond”. “Trim and Respond” is a distributed control logic currently utilized in buildings [67]; it has been shown to be more efficient and is considered to be more intuitive to tune than control logic used by equipment manufacturers.

The chapter is divided into two parts. In the first part, we compute the explicit solution of a one-step distributed MPC controller and show how to implement it on existing networked buildings control platforms. In the second part, we present the “Trim and Respond” logic and compare it with the explicit solution of a one-step DMPC. We show that “Trim and Respond” is a special class of one-step DMPC. The results of this chapter can impact the way distributed building control logics are designed. On one hand, they allow the implementation of DMPC on existing distributed building control platforms with a small amount of communication; on the other hand, they can be used to improve “Trim and Respond” logic by decreasing the amount of manual parameter tuning.

The chapter is organized in the following way. First, the constrained finite time optimization problem for MPC is formulated in Section 3.2. The distributed explicit solution is derived from KKT conditions in Section 3.3. “Trim and Respond” logic is introduced in Section 3.4. The DMPC and “Trim and Respond” methods are compared in Section 3.5, a simulation of both logics is shown in Section 3.6, and finally, conclusions are drawn in Section 3.7.
3.2 Model Predictive Control

In this section, a model predictive controller is designed to minimize the total energy consumption of the linearized HVAC system defined in Section 2.1 in equation (2.3), and to satisfy the thermal comfort constraints in equation (2.21) and the reheating and cooling constraints (2.22). Consider the following MPC problem at time step $k$:

\[
\begin{align*}
\min_{u_c(k), u_{h,i}(k)} & \quad u_c(k)^2 + \sum_{i=1}^{n} u_{h,i}(k)^2 \\
\text{subject to} & \quad A_i T_i(k) + B_{c,i} u_c(k) + B_{h,i} u_{h,i}(k) + \hat{w}_i(k) \leq \bar{T} \\
& \quad A_i T_i(k) + B_{c,i} u_c(k) + B_{h,i} u_{h,i}(k) + \hat{w}_i(k) \geq T \\
& \quad u_{h,i}(k) \geq 0 \\
& \quad u_c(k) \leq 0 \\
& \quad \forall i \in \{1, \ldots, n\},
\end{align*}
\]

where the prediction horizon length is one, $\hat{w}_i(k)$ is the estimated load at the $i$-th zone, (3.1b) and (3.1c) are thermal comfort constraints at the next time step $k + 1$, (3.1d) is the local actuation reheating constraint, and (3.1e) is the central cooling constraint.

Let the optimal solution to problem (3.1) be

\[
U^* = \{u_c(k)^*, u_{h,1}(k)^*, \ldots, u_{h,n}(k)^*\}.
\]

At time step $k$, the solution $u_c(k)^*$ is implemented by the AHU and $u_{h,i}(k)^*$ is implemented by the $i$-th VAV box. The optimization (3.1) is repeated at time step $k + 1$, with the updated state estimation $x_i(k + 1)$ and estimated load $\hat{w}_i(k + 1)$, yielding a receding horizon control strategy.

3.3 Explicit Distributed MPC Solution

The solution $U^*$ in (3.2) can be computed by solving a quadratic program at each time step $k$ for a given $x_1(k), \hat{w}_1(k), \ldots, x_n(k), \hat{w}_n(k)$. However, this would result in a centralized approach and would require software not readily available on current buildings embedded control systems.

Karush-Kuhn-Tucker (KKT) necessary and sufficient conditions are used to find the explicit expression of the optimal controller solution to problem (3.1). Compared to standard explicit MPC control design [70], our procedure provides a distributed explicit controller. The Lagrangian for problem (3.1) is

\[
L(u_c(k), u_{h,1}(k), \ldots, u_{h,n}(k), \lambda_c(k), \lambda_1(k), \ldots, \lambda_n(k), \\
\bar{\mu}_1(k), \ldots, \bar{\mu}_n(k), \mu_1(k), \ldots, \mu_n(k)) = \\
u_c(k)^2 + \lambda_c(k)(u_c(k)) + \sum_{i=1}^{n} \left(u_{h,i}(k)^2 + \lambda_i(k)(-u_{h,i}(k))\right)
\]
CHAPTER 3. BUILDING TEMPERATURE DISTRIBUTED CONTROL VIA
EXPLICIT MPC AND “TRIM AND RESPOND” METHODS

\[ + \bar{\mu}_i(k)(A_iT_i(k) + \bar{B}_{c,i}u_c(k) + B_{h,i}u_{h,i}(k) + \bar{\omega}_i(k) - \bar{T}) \]
\[ + \mu_i(k)(-A_iT_i(k) - \bar{B}_{c,i}u_c(k) - B_{h,i}u_{h,i}(k) - \bar{\omega}_i(k) + \bar{T}) \] \hfill (3.3)

Three dual variables exist for each zone \( i \): \( \lambda_i(k) \) is associated with the non-negativity constraint on \( u_{h,i}(k) \) (3.1d), \( \bar{\mu}_i(k) \) is associated with the upper bound on the temperature \( T_i(k + 1) \) (3.1b), and \( \mu_i(k) \) is associated with the lower bound on the temperature \( T_i(k + 1) \) (3.1c). Also, \( \lambda_c(k) \) corresponds to the non-positivity constraint on \( u_c(k) \) (3.1e). The KKT conditions for problem (3.1) at time \( k \) are:

\[ 2u_c(k) + \sum_{i=1}^{n} B_{c,i}(\bar{\mu}_i(k) - \mu_i(k)) + \lambda_c(k) = 0 \] \hfill (3.4a)
\[ 2u_{h,i}(k) - \lambda_i(k) + B_{h,i}(\bar{\mu}_i(k) - \mu_i(k)) = 0 \] \hfill (3.4b)
\[ \lambda_c(k)u_c(k) = 0 \] \hfill (3.4c)
\[ \lambda_i(k)u_{h,i}(k) = 0 \] \hfill (3.4d)
\[ \bar{\mu}_i(k)(A_iT_i(k) + \bar{B}_{c,i}u_c(k) + B_{h,i}u_{h,i}(k) + \bar{\omega}_i(k) - \bar{T}) = 0 \] \hfill (3.4e)
\[ \mu_i(k)(A_iT_i(k) + \bar{B}_{c,i}u_c(k) + B_{h,i}u_{h,i}(k) + \bar{\omega}_i(k) - \bar{T}) = 0 \] \hfill (3.4f)
\[ u_c(k) \leq 0 \] \hfill (3.4g)
\[ u_{h,i}(k) \geq 0 \] \hfill (3.4h)
\[ A_iT_i(k) + \bar{B}_{c,i}u_c(k) + B_{h,i}u_{h,i}(k) + \bar{\omega}_i(k) \leq \bar{T} \] \hfill (3.4i)
\[ A_iT_i(k) + \bar{B}_{c,i}u_c(k) + B_{h,i}u_{h,i}(k) + \bar{\omega}_i(k) \geq \bar{T} \] \hfill (3.4j)
\[ \lambda_c(k) \geq 0 \] \hfill (3.4k)
\[ \lambda_i(k) \geq 0 \] \hfill (3.4l)
\[ \bar{\mu}_i(k) \geq 0 \] \hfill (3.4m)
\[ \mu_i(k) \geq 0 \] \hfill (3.4n)
\[ i \in \{1, \ldots, n\}, \]

where equations (3.4a) to (3.4b) are the components of the KKT condition \( \nabla L(u_c(k), u_1(k), \ldots) = 0 \), equations (3.4c) to (3.4f) are the complementary slackness conditions, equations (3.4h) to (3.4j) are the primal feasibility conditions, and equations (3.4k) to (3.4n) are the dual feasibility conditions.

In order to compute the feedback control law solving (3.4), we need to enumerate all possible combinations of active primal constraints. Intuition from physics makes this task easier. Only combinations of temperature bound constraints (3.4i) and (3.4j) need be considered, as the activeness of (3.4g) and (3.4h) are fully determined from active sets of (3.4i) and (3.4j). Let \( \mathcal{I}_H \) and \( \mathcal{I}_C \) be two subsets of the set \( \mathcal{I} = \{1, \ldots, n\} \). One can prove that only the following cases will generate a primal and dual feasible controller.

**Case I** Lower bound temperature constraints (3.4j) are active for all \( i \in \mathcal{I}_H \); upper bound temperature constraints (3.4i) are not active for all \( i \in \mathcal{I} \)

**Case II** Upper bound temperature constraints (3.4i) are active for all \( i \in \mathcal{I}_C \); lower bound temperature constraints (3.4j) are not active for all \( i \in \mathcal{I} \)
CHAPTER 3. BUILDING TEMPERATURE DISTRIBUTED CONTROL VIA EXPLICIT MPC AND “TRIM AND RESPOND” METHODS

Case III Upper bound temperature constraints (3.4i) are active for all \( i \in \mathcal{I}_C \); lower bound temperature constraints (3.4j) are active for all \( i \in \mathcal{I}_H \).

Case IV Both Temperature bound constraints (3.4i) and (3.4j) are not active for all \( i \in \mathcal{I} \).

Note that in Case III a single zone cannot simultaneously have an active upper temperature constraint and an active lower constraint, i.e. \( \mathcal{I}_H \cap \mathcal{I}_C = \emptyset \) in Case III.

Remark 1. All primal variables, dual variables, and inputs are a function of time step \( k \), but for readability this argument will be dropped from the notation for the remainder of this section.

The primal variables can be written as a function of the dual variables:

\[
\begin{align*}
    u_c &= \frac{1}{2} \left( -\lambda_c + \sum_{i \in \mathcal{I}} B_{c,i}(\mu_i - \bar{\mu}_i) \right) \\
    u_i &= \frac{1}{2} \left( B_{h,i}(\mu_i - \bar{\mu}_i) + \lambda_i \right).
\end{align*}
\]

(3.5a)

(3.5b)

Note that the optimal control law for Case IV is \( u_c = u_i = 0 \) \( \forall i \in \mathcal{I} \). The derivation of the optimal distributed controller for Case I is described next and a summary of the other cases are provided in the following subsections.

Case I: Active Lower Bound Temperature Constraints and No Active Upper Bound Temperature Constraint

In Case I we have:

\[
\begin{align*}
    \bar{\mu}_i &= 0 \quad \forall i \in \mathcal{I} \\
    \mu_i &= 0 \quad \forall i \in \mathcal{I} \setminus \mathcal{I}_H \\
    A_i T_i + B_{c,i} u_c + B_{h,i} u_i + \dot{d}_i &= T \quad \forall i \in \mathcal{I}_H.
\end{align*}
\]

(3.6a)

(3.6b)

(3.6c)

Simplifying (3.5a) by using complimentary slackness (3.4d) with dual feasibility (3.4k) and (3.4m), we obtain:

\[
\begin{align*}
    u_c &= 0, \\
    \lambda_c &= \sum_{i \in \mathcal{I}_H} B_{c,i} \mu_i.
\end{align*}
\]

(3.7a)

(3.7b)

Thus, there is no cooling involved in Case I. Simplifying equation (3.5b) using complimentary slackness condition (3.4c), it is evident that \( \lambda_i = 0 \) \( \forall i \in \mathcal{I} \). The local control law can be written as:

\[
    u_i = \frac{B_{h,i}}{2} \frac{\mu_i}{\bar{\mu}_i} = \frac{T - A_i T_i - \dot{d}_i}{B_{h,i}} \quad \forall i \in \mathcal{I}_H.
\]

(3.8a)
In order for \( u_i \geq 0 \) and \( \mu_i \geq 0 \) for all \( i \in \mathcal{I}_H \), it is necessary that \( T - A_i T_i - \hat{d}_i \geq 0 \) (note that \( B_{h,i} > 0 \)). Also, if \( T - A_i T_i - \hat{d}_i = 0 \) for all \( i \in \mathcal{I}_H \), the lower temperature bound is weakly active for all \( i \in \mathcal{I}_H \) and the control law corresponds to the Case IV control law.

**Remark 2.** Intuitively, the optimality region of the Case I control law can be thought of as a non-cooling region because \( u_i \geq 0 \) and \( u_c = 0 \). The zones \( i \in \mathcal{I}_H \) must request heat because they would violate the lower temperature bound on \( T_i(k+1) \) if \( u_c = u_i = 0 \).

**Case II: Active Upper Bound Temperature Constraints and No Active Lower Bound Temperature Constraint**

In Case II we have:

\[
\begin{align*}
\mu_i &= 0 \quad \forall i \in \mathcal{I} \\
\bar{\mu}_i &= 0 \quad \forall i \in \mathcal{I} \setminus \mathcal{I}_C \\
A_i T_i + B_{c,i} u_c + B_{h,i} u_i + \hat{d}_i &= \bar{T} \quad \forall i \in \mathcal{I}_C.
\end{align*}
\]

Following a similar approach to Case I, we obtain:

\[
\begin{align*}
u_c &= -\frac{(A_i T_i + \hat{d}_i - \bar{T})}{B_{c,i}} \quad \forall i \in \mathcal{I}_C \quad (3.10a) \\
u_i &= 0 \quad \forall i \in \mathcal{I} \quad (3.10b)
\end{align*}
\]

Note that \( u_c \) must be the same for all \( i \in \mathcal{I}_C \) and \( (A_i T_i + \hat{d}_i - \bar{T}) \geq 0 \) if \( i \in \mathcal{I}_C \). If \( u_c = 0 \), the constraints here are weakly active and the optimal controller is the same as in Case IV.

**Remark 3.** Intuitively, this region is a non-heating region because \( u_c \leq 0 \) and \( u_i = 0 \) for all \( i \in \mathcal{I} \).

**Case III: Active Upper Bound Temperature Constraints and Active Lower Bound Temperature Constraints**

In Case III we have:

\[
\begin{align*}
\bar{\mu}_i &= 0 \quad \forall i \in \mathcal{I} \setminus \mathcal{I}_C \\
A_i T_i + B_{c,i} u_c + B_{h,i} u_i + \hat{d}_i &= \bar{T} \quad \forall i \in \mathcal{I}_C \\
\mu_i &= 0 \quad \forall i \in \mathcal{I} \setminus \mathcal{I}_H \\
A_i T_i + B_{c,i} u_c + B_{h,i} u_i + \hat{d}_i &= T \quad \forall i \in \mathcal{I}_H.
\end{align*}
\]
CHAPTER 3. BUILDING TEMPERATURE DISTRIBUTED CONTROL VIA EXPLICIT MPC AND “TRIM AND RESPOND” METHODS

By following the same approach as Case I and Case II, we can compute the optimal control law for Case III.

\[ u_c = -\frac{(A_i T_i + \hat{d}_i - \bar{T})}{B_{c,i}} \quad \forall i \in \mathcal{I}_C \]  

(3.12a)

\[ u_i = 0 \quad \forall i \in \mathcal{T} \setminus \mathcal{I}_H \]  

(3.12b)

\[ u_i = \frac{T - A_i T_i - \hat{d}_i - B_C u_c}{B_{h,i}} \quad \forall i \in \mathcal{I}_H. \]  

(3.12c)

**Remark 4.** Intuitively, this region can be thought of as a mixed heating and cooling region because \( u_c \leq 0 \) and \( u_i \geq 0 \) \( \forall i \in \mathcal{I}_H \).

**Summary of Distributed Feedback Control Algorithm**

For all cases, it is convenient to define an initial guess for the dual variables \( \mu_i \) and \( \bar{\mu}_i \) as follows:

\[ \mu_0^i = (T - A_i T_i - \hat{d}_i)_+ \]  

(3.13a)

\[ \bar{\mu}_0^i = (-\bar{T} + A_i T_i + \hat{d}_i)_+ \]  

(3.13b)

where \((x)_+ := max(x, 0)\). Zone \( i \) requests heat if \( \mu_0^i > 0 \), and likewise, \( \bar{\mu}_0^i > 0 \) indicates that zone \( i \) requests cooling. In fact, given parameter values \( T_i \) and \( \hat{d}_i \) for all \( i \in \mathcal{T} \), the sets \( \mathcal{I}_C \) and \( \mathcal{I}_H \) can be determined as follows:

\[ \mathcal{I}_C = \left\{ i \in \mathcal{T} \mid i = \text{arg min}_i \frac{-\bar{\mu}_0^i}{B_{c,i}} \right\} \]  

(3.14)

where \( \mathcal{I}_C = \emptyset \) if the minimizing value is zero, and

\[ \mathcal{I}_H = \left\{ i \in \mathcal{T} \mid \mu_0^i - B_{c,i} u_c > 0 \right\}. \]  

(3.15)

The zones in \( \mathcal{I}_C \) correspond to the zones requesting the maximum cooling required for any zone to maintain the upper temperature bound constraint \((3.4i)\). Similarly, the zones in \( \mathcal{I}_H \) request heat in order to maintain the lower temperature bound constraint \((3.4j)\), assuming \( u_c \) has been calculated. Case I has \( \mathcal{I}_C = \emptyset \), Case II has \( \mathcal{I}_H = \emptyset \), and Case IV has \( \mathcal{I}_C = \mathcal{I}_H = \emptyset \).

Because the optimal law for \( u_c \) is determined from a sorting of \( \bar{\mu}_0^i \) and does not rely on \( u_i \), only one local-to-central and one central-to-local communication is required. Therefore, the structure of this problem allows us to design a dual decomposition algorithm which converges in one step. The one-step DMPC solution is provided in Algorithm 1. Throughout the algorithm, it is assumed that each zone only knows its own model parameters \((A_i, B_{c,i}, B_{h,i}, \hat{d}_i)\), and the central processor knows \( B_{c,i} \). The keyword “Local” indicates a computation at the VAV processor, and the keyword “Central” indicates a computation at the AHU processor.
CHAPTER 3. BUILDING TEMPERATURE DISTRIBUTED CONTROL VIA
EXPLICIT MPC AND “TRIM AND RESPOND” METHODS

Algorithm 1 Explicit Solution of one-step DMPC

**Computation of initial dual guess:**
1: Local: Calculate $\mu_i^0$ from (3.13a) and $\bar{\mu}_i^0$ from (3.13b)
2: Send $\mu_i^0$ and $\bar{\mu}_i^0$ to central AHU processor.

**Computation of Optimal Controller:**
1: Central: Compute $I_C$ from (3.14)
2: if $I_C == \emptyset$ then
3:   Central: $u_c := 0$
4: else
5:   Central: Compute $u_c$ from (3.12a)
6: end if
7: Send $u_c$ to local VAV processors.
8: Local: Compute if $i \in I_H$ from (3.15)
9: if $i \in I_H$ then
10:   Local: Compute $u_i$ from (3.12c)
11: else
12:   Local: $u_i := 0$
13: end if

**Remark 5.** The Explicit DMPC solution presented in this Section 3.3 is a fixed point solution of the dual-decomposition algorithm [71]. In fact, at each time step $k$ the iterative dual-decomposition algorithm converges to one of the optimal controllers (Cases I-IV) presented in Section 3.3.

### 3.4 Trim and Respond

A well-regarded example of distributed control logic used in buildings is the “Trim and Respond” logic [67]. In this logic, each reheating input $u_{h,i}(k)$ is controlled locally by a PI loop regulating zone temperature to remain within comfort bounds. Each zone generates a heating request when the reheating coil is approaching its maximum capacity. This maximum capacity is denoted as $\bar{u}(k)$ and is changed over time by the “Trim and Respond” logic. Likewise, each zone generates a cooling request when the zone is close to its lower thermal bound. Since there is no cooling capacity at each zone $i$, the zone must request that the central cooling input $u_c(k)$ add more cool air to the system, and thus $u_c(k)$ is also modulated over time by the “Trim and Respond” logic as a function of cooling requests.

Next, we provide more details about the “Trim and Respond” logic. A heating request $r_{H,i}(k)$ is generated by zone $i$ at time step $k$ if its local reheat control $u_{h,i}(k)$ is near its maximum capacity $\bar{u}(k)$. Similarly, zone $i$ generates a cooling request $r_{c,i}(k)$ if its local reheating $u_{h,i}(k)$ is near zero and the temperature is close to its upper bound $\bar{T}$. Each zone has an importance multiplier $M_i \in \mathbb{Z}_{>0}$ which can be manually increased for a zone to have
more influence on central control. The following formulas describe request generation for zone \( i \):

\[
\text{If } u_{h,i}(k) \leq 0.05, T_i(k) \geq \bar{T} - 0.1, \quad r_i^c(k) = M_i. \\
\text{If } u_{h,i}(k) \geq 0.95 \bar{u}(k), \quad r_i^h(k) = M_i. \tag{3.16a}
\]

If no cooling request is generated (conditional for (3.16a) is not met), \( r_i^c(k) = 0 \), and if no heating request is generated (conditional for (3.16b) is not met), \( r_i^h(k) = 0 \).

Requests are sent to centralized platforms associated with the main air handling unit and a central building boiler. The value for \( u_c(k) \) is reset at the AHU by adjusting the cooling coil valve position, and \( \bar{u}(k) \) is reset at the boiler by updating the supply water temperature that is provided to local reheating coils. In particular, the “Trim and Respond” logic responds to heating requests by decreasing \( u_c(k) \) by an amount proportional to the total number of cooling requests and by increasing \( \bar{u}(k) \) by an amount proportional to the number of heating requests.

At every time step \( k \), regardless of the number of requests, the values of \( u_c(k) \), \( \bar{u}(k) \) are also “trimmed”, or \( u_c(k) \) and \( \bar{u}(k) \) are decremented toward 0. Note that \( u_c(k) \) can only take on non-positive values whereas \( \bar{u}(k) \) can only take on non-negative values. This trimming effect ensures reduction in energy usage by decreasing centralized control efforts when no requests are made.

Let the sum of requests be \( \text{req}_c(k) \) and \( \text{req}_h(k) \):

\[
\text{req}_c(k) = \sum_{i=1}^{n} r_{i}^c(k), \quad \text{req}_h(k) = \sum_{i=1}^{n} r_{i}^h(k). \tag{3.17}
\]

The central controls then “trim” and “respond” to these summed requests as follows:

In cooling mode, \( (\text{req}_c(k) > \text{req}_h(k)) \)

\[
u_c(k+1) = u_c(k) + \text{trim}_c + (\text{req}_c(k) - r_I)\text{res}_c \tag{3.18a}
\]

\[
\bar{u}(k+1) = \bar{u}(k) + \text{trim}_b. \tag{3.18b}
\]

In heating mode, \( (\text{req}_h(k) > \text{req}_c(k)) \)

\[
u_c(k+1) = u_c(k) + \text{trim}_c \tag{3.19a}
\]

\[
\bar{u}(k+1) = \bar{u}(k) + \text{trim}_b + (\text{req}_h(k) - r_I)\text{res}_h. \tag{3.19b}
\]

where the tuning parameters are: \( r_I \), a fixed number of ignored requests; \( \text{res}_c \), \( \text{res}_b \), the proportional “response” per request for \( u_c \) in cooling mode and \( \bar{u} \) in heating mode, respectively; and \( \text{trim}_c \), \( \text{trim}_b \) “trim” amounts. If \( \text{req}_c(k) = \text{req}_h(k) \), then the control law uses (3.18b) and (3.19a).

The values \( u_c(k) \), \( \bar{u}(k) \), \( \text{req}_c(k) \) and \( \text{req}_h(k) \) are also limited by the following upper and lower bounds:

\[
\text{req}_c(k) \leq \frac{\max_c}{\text{res}_c} + r_I, \quad \text{req}_h(k) \leq \frac{\max_b}{\text{res}_b} + r_I. \tag{3.20a}
\]
\[ u_c \in [u_{c_{\min}}, u_{c_{\max}}], \quad \bar{u} \in [0, \bar{u}_{\max}]. \quad (3.20b) \]

An example set of parameters used in cooling mode on an actual building is \( \text{trim}_c = 5/9^\circ C, \) \( \text{res}_c = -10/9^\circ C, \) \( u_{c_{\max}} = 0^\circ C, \) \( u_{c_{\min}} = -12^\circ C, \) \( \text{max}_c = -4^\circ C, \) \( r_I = 0. \)

**Remark 6.** The logic presented in this section is a simplified version of “Trim and Respond” logic used in buildings today. In particular, we have only considered heating and cooling requests. In practice, the logic extends to pressure control as well.

### 3.5 Comparison of Methods

The explicit MPC method and the “Trim and Respond” method lead to controllers with the same structure and similar data communication. A detailed explanation is provided next.

The initial dual variables and requests have the same meaning: initial dual variables are a real-valued measurement of temperature bound violation (3.13), whereas requests (3.16) are integer-valued indicators of zone heating or cooling demand. While dual variables are the one-step predicted heating or cooling demand, requests are manually tuned to demand an appropriate amount of central control adjustment. The “Trim and Respond” algorithm handles this coarse measurement of demand by ignoring some requests (using \( \text{req}_I \)) and setting a maximum change in centralized control, e.g. \( \text{max}_C \) to prevent overreaction. If the model is accurate, the dual variables provide more valuable information than the boolean requests values.

The “Trim and Respond” importance factor \( M_i \) attempts to manually find the zone that has the highest cooling demand. Typically, one zone tends to request the most cooling (e.g. a computer server room or other zone with regularly high levels of thermal load), and so it is given the highest importance factor. The MPC algorithm automatically finds this zone by calculating \( I_C. \)

Even if requests are only a function of the current state \( T(k) \), predictive capability is introduced in practice. Requests are generated when local controls are close to their limits instead of at saturation. Thus, requests are generated when temperature bounds are likely to be saturated in the near future.

The explicit MPC algorithm and the “Trim and Respond” logics communicate between processors in a similar fashion. “Trim and Respond” sends information from local processors to centralized processors. Communication from central to local processors is not present because each local PI controller will implicitly respond to changes in \( u_c. \) Explicit MPC requires one communication each way between local and central processors (see Algorithm 1).

Both methods address energy efficiency. MPC directly addresses this by minimizing an effective cost of energy use. “Trim and Respond” addresses energy consumption by using a “trim” factor to decrease the control setpoints.
Finally, “Trim and Respond” does not require load estimation or a system dynamics model. However, there are many parameters to manually tune in the “Trim and Respond” algorithm.

3.6 Results: Simulations

Simulation results are presented in this section for the Model Predictive Control algorithm and the “Trim and Respond” algorithm. The simulation is run with $n = 10$ zones. The MPC has a sampling time of 5 minutes; the “Trim and Respond” logic updates the centralized controls $u_c(k), \bar{u}(k)$ every 5 minutes with zone-level PI loops running at 1 minute intervals. Model parameters from equation (2.1) are: $C_i = 9.2e3\text{kJ}/\text{C}$, $R_i = 50\text{C}/\text{kW}$, $T_0 = 21\text{C}$, and $\bar{T} = 26\text{C}$.

The disturbance $w$ is shown in Figure 3.1a, estimated from data from the UC Berkeley Bancroft Library. Simulation results are provided in Figure 3.1. The MPC implementation uses a Luenberger observer to estimate the load, allowing a slight violation of temperature bounds. The “Trim and Respond” profile shows many small variations in temperature, while MPC provides a smoother result. The small variations are an artifact of the integer requests and the prefixed resolution by which central controls can change in the “Trim and Respond” logic. Similar comparisons exist between the control inputs of the two methods as well as between the MPC initial dual variables and the “Trim and Respond” requests. A calculation of total power used illustrates that the explicit MPC outperforms the “Trim and Respond” method.

3.7 Conclusion

In this chapter, we have demonstrated two ways to approach the design of distributed control in building HVAC systems. A one-step DMPC was formulated and explicitly solved by using KKT conditions. It was then compared to an existing control strategy, “Trim and Respond”. Simulation results of both algorithms are very similar.
CHAPTER 3. BUILDING TEMPERATURE DISTRIBUTED CONTROL VIA
EXPLICIT MPC AND “TRIM AND RESPOND” METHODS

(a) Disturbance Profile $d_i$ over one day

Figure 3.1: Top: disturbance profile over one day. Left column: MPC simulation results. Right column: “Trim and Respond” simulation results
Chapter 4

A Primal-Dual Active-Set Method for Distributed Model Predictive Control

Active-set methods are a class of constrained optimization algorithms. Active-set algorithms work by iteratively finding the subset of inequality constraints for which equality holds at the optimum. This set is called the optimal active set. In traditional active-set methods, only one constraint is added or removed from the active set during each iteration. As a result, traditional active-set methods can require an exponential number of iterations to find the optimal active set. In this chapter, we examine a primal-dual active-set method for strictly convex quadratic programs that was initially proposed by Hintermüller [72] and expanded upon by Curtis [73]. In this algorithm, multiple constraints are added to or removed from the active set during each iteration of the algorithm. As a result, the algorithm often converges in very few iterations and it exhibits local superlinear convergence [74]. The ability of this primal-dual active-set method to find the optimal active set quickly makes the method competitive with interior-point methods. In addition, interior point methods are known to lose accuracy in the presence of ill-conditioning; this does not happen for active-set methods [75]. These differences are particularly important in nonlinear optimization algorithms such as sequential quadratic programming where many quadratic programs (QPs) need to be solved.

In this chapter, we build on the work of [73] and present a novel distributed primal-dual active-set method. We consider a class of Model Predictive Control (MPC) problems that have quadratic cost, linear dynamics, additive disturbance, and box constraints with dynamics coupled through the control inputs. This class of problems includes large-scale control problems such as buildings, network flow [17], or multi-agent coordination [18].

Deploying model predictive controllers on large-scale distributed systems is challenging since it requires solving large-scale optimization problems in real time on embedded hardware. One popular solution to this problem is to distribute the computations performed by the optimization algorithm amongst distributed platforms. Methods include matrix splitting [76], dual decomposition [25], the Alternating Direction Method of Multipliers (ADMM) [17], an alternating minimization algorithm [18], a distributed inexact Newton method [34], and
more. The disadvantage of these solutions is high communication costs. These algorithms require the communication of a large number of variables during each iteration. Additionally, it may require many iterations to converge, involving substantial communication to find the solution. Moreover, it is well known in the parallel computing community that communication between processors is dramatically slower than computation on a single chip [77], [78]. The proposed primal-dual active-set algorithm is particularly well suited for distributed implementation in systems with high communication costs. The algorithm is naturally decomposable for optimization on a star communication network for large-scale systems having dynamics coupled through the control inputs. Additionally, the decomposition fits a typical hardware architecture where the leaf nodes of the network have limited computational resources and the central hub of the star network has substantially more computing power.

One example of a system with prohibitively high communication costs is heating, ventilation, and air-conditioning (HVAC) of modern commercial buildings. Commercial buildings contain many thermal zones, each having dedicated computing resources. However, communicating between these computing resources can be expensive. Using standard building protocols such as BACnet and ARCnet, we have come across communication delays ranging 0.03 to 0.4 seconds per floating point number. A proposed solution for distributed optimization of building control using MPC is dual decomposition [79]. However, the effect of these communication delays is detrimental to the overall performance of the controller. Our recent work has shown that the large amounts of communication required by dual decomposition on an HVAC system would prohibit the controller from being implemented in real time [53]. Perhaps the most popular algorithm for distributed optimization today is ADMM. There are numerous publications in the controls, machine learning, and other communities that show that ADMM has gained widespread acceptance and usage; see [17], [36], [80], [81], [82]. ADMM can be implemented on a star communication network of distributed processors and in practice it has better convergence rates than dual decomposition [30].

In this chapter we compare the computational and communication complexities of the primal-dual active-set algorithm with ADMM and dual decomposition. Our theoretical analysis shows that for the class of problems considered in this chapter, both dual decomposition and ADMM have lower computational costs per iteration but they require many more iterations to converge, resulting in much higher overall communication costs than the primal-dual active-set method. Therefore, the primal-dual active-set method is suited for distributed control with high communication latency. We confirm our theoretical findings with numerical tests which compare these algorithms. We find that the smaller number of iterations and number of variables per iteration significantly reduces the computation time. As a result, the primal-dual active-set method outperforms dual decomposition and ADMM in the considered systems.

We note that other similar active-set methods have been proposed in the literature and have been tailored for MPC problems: see [83], [84], [85]. In [83] and [84], the algorithms reformulate a strictly convex quadratic program as an unconstrained quadratic convex spline. These algorithms are able to change multiple active sets at a time, and do not exhibit cycling. In [85], the algorithm takes an online active-set strategy that exploits the parametric
formulation of MPC. None of these algorithms have been shown to be implementable in a
distributed fashion. Also, all of these algorithms utilize dense linear algebra whereas our
algorithm uses sparse linear algebra, which performs much better for large-scale problems.
We show in our results that sparse linear algebra in our proposed algorithm outperforms
dense linear algebra in the qpOASES algorithm [85].

The chapter is organized as follows. First, we introduce the primal-dual active-set method
proposed by Hintermüller [72]. We show how this algorithm can be applied to Model Pre-
dictive Control problems. Next we describe the distributed primal-dual active-set method.
We discuss the theoretical complexities of the primal-dual active-set method, dual decompo-
sition, and ADMM. Then, we show how to apply our algorithm to building control. Finally
we present numerical results comparing the proposed distributed algorithm with ADMM,
dual decomposition, as well as with a centralized active-set solver qpOASES [85].

4.1 Primal-Dual Active-Set Method

In this section we describe the primal-dual active-set algorithm presented in [72] and [73].
This algorithm can be reformulated as a semi-smooth Newton method with local superlinear
convergence.

Active-Set Algorithm

Consider the following strictly convex quadratic program,

\[
\begin{align*}
\min_{z} & \quad \frac{1}{2} z^T H z + c^T z \\
\text{s.t.} & \quad Cz = d \\
& \quad \underline{z} \leq z \leq \overline{z}
\end{align*}
\]

where \( z \in \mathbb{R}^p, \ H = H^T \succ 0, \ c \in \mathbb{R}^p, \ C \in \mathbb{R}^{q \times p}, \ d \in \mathbb{R}^q, \) and \( \underline{z} < z < \overline{z} \in \mathbb{R}^p \) where \( \mathbb{R} = \mathbb{R} \cup \{\pm \infty\} \) is the extended real-line.

Let \( \nu \in \mathbb{R}^q \) be the dual variables corresponding to the equality constraints (4.1b) and
let \( \bar{\lambda} \in \mathbb{R}^p \) and \( \underline{\lambda} \in \mathbb{R}^p \) be the dual variables for the upper and lower bound constraints
(4.1c), respectively. The Karush-Kuhn-Tucker (KKT) necessary and sufficient conditions for
optimality for the quadratic program (4.1) are

\[
\begin{align*}
Hz + c + C^T \nu + \bar{\lambda} - \underline{\lambda} & = 0 \\
Cz - d & = 0 \\
\min(\overline{z} - z, \bar{\lambda}) & = 0 \\
\min(z - \underline{z}, \underline{\lambda}) & = 0
\end{align*}
\]

Let \( J = \{1, \ldots, p\} \) be the index set for the elements of the vector \( z \in \mathbb{R}^p \). For a primal
variable \( z \), we define the following disjoint subsets of \( J \):

\[
\mathcal{A} = \{ j \in J : z_j = \overline{z}_j \},
\]

(4.3a)
The set \( \mathcal{A} \) is the set of components of \( z \) that are active at their upper bound \( z \) and the set \( \mathcal{A} \) is the set of components of \( z \) that are active at their lower bound \( z \). We define \( \mathcal{A} = \mathcal{A} \cup \mathcal{A} \) as the set of elements that are active at the upper-bound or lower-bound. The set \( \mathcal{I} = \mathcal{J} \setminus \mathcal{A} \) is the set of elements of \( z \) that are not active. We will denote by \( z_j \) the elements \( z_j \) of \( z \) for \( j \in \mathcal{A} \). Similarly, \( z_j \) is the elements \( z_j \) of \( z \) for \( j \in \mathcal{I} \).

The primal-dual active-set algorithm uses the active-sets \( \mathcal{A} \) and \( \mathcal{A} \) to update the primal \( z \) and dual variables \( \nu, \bar{\lambda}, \bar{\lambda} \). The primal and dual variables are then used to update the active sets.

The algorithm is initialized with an active-set partition \( \mathcal{A}, \mathcal{A}, \mathcal{I} \). The active elements, \( z_j \), are set equal to their active bound and \( z_j \) is left as a free variable. The active-set partition must satisfy feasibility (4.2b), i.e. there must exist some \( z_j \) such that the set \( Z_j \) is not empty, where \( Z_j \) is defined as follows:

\[
Z_j := \{ z_j | C_j = C_\mathcal{A} z_\mathcal{A} + C_\mathcal{I} z_\mathcal{I} = d \}
\]

The first step of the algorithm is to find primal variables \( z_j \) and equality dual variables \( \nu \) that satisfy the stationarity (4.2a) and feasibility (4.2b) conditions

\[
\begin{bmatrix}
H_{\mathcal{I}, \mathcal{I}} & C_\mathcal{I}^T \\
C_\mathcal{I} & 0
\end{bmatrix}
\begin{bmatrix}
z_j \\
\nu
\end{bmatrix}
= 
\begin{bmatrix}
-H_{\mathcal{I}, \mathcal{A}} z_\mathcal{A} - c_\mathcal{I} \\
-C_\mathcal{A} z_\mathcal{A} + d
\end{bmatrix}
\]

where \( \bar{\lambda}_j = \bar{\lambda}_j = 0 \). Solving (4.5) is called subspace minimization.

Next, the algorithm updates the inequality dual variables \( \lambda_j \) and \( \bar{\lambda}_j \) to satisfy the stationarity (4.2a) condition

\[
\begin{align*}
\bar{\lambda}_j &= \begin{cases} 
H_j z + c_j + (C_j^T \nu)_j & \text{if } j \in \mathcal{A} \\
0 & \text{if } j \notin \mathcal{A}
\end{cases} \\
\lambda_j &= \begin{cases} 
H_j z + c_j + (C_j^T \nu)_j & \text{if } j \in \mathcal{A} \\
0 & \text{if } j \notin \mathcal{A}
\end{cases}
\end{align*}
\]

where \( H_j \) is the \( j \)-th row of \( H \). Note that \( H_j z \) can be reduced to scalar multiplication \( H_{jj} z_j \) if \( H \) is diagonal.

The active-sets are updated

\[
\begin{align*}
\mathcal{A}^+ &\leftarrow \{ j | (z_j > z_j \text{ and } j \in \mathcal{I}) \text{ OR } (\bar{\lambda}_j > 0 \text{ and } j \in \mathcal{A}) \} \\
\mathcal{A}^- &\leftarrow \{ j | (z_j < z_j \text{ and } j \in \mathcal{I}) \text{ OR } (\Delta_j > 0 \text{ and } j \in \mathcal{A}) \} \\
\mathcal{I}^+ &\leftarrow \{ j | j \notin \mathcal{A} \cup \mathcal{A}^+ \}
\end{align*}
\]

where the plus superscript indicates the update of the active-set at the next iteration. The algorithm then iterates using (4.5)-(4.7).
Theorem 1 (c.f. Theorem 1 of [73]). Assume problem (4.1) is feasible. If the active set update (4.7) does not change the active set from its previous iteration, then the algorithm has found the optimal solution to (4.1) and thus terminates.

Proof. If (4.7) does not change the active set, then the complementarity conditions (4.2c) and (4.2d) are satisfied. KKT conditions (4.2a), (4.2b) are satisfied by (4.5) and (4.6). The algorithm can terminate because the algorithm has found primal \( z \) and dual \( \nu, \lambda, \bar{\lambda} \) variables that satisfy the KKT conditions (4.2).

Thus, optimality can be verified by checking if the active-sets \( \mathcal{A} \) and \( \overline{\mathcal{A}} \) have changed during the last iteration; the algorithm is summarized in Algorithm 2. In [73], extensions are made to this algorithm to deal with potential cycling of the active-set updates. We do not consider this case in our chapter.

Algorithm 2 Primal-Dual Active-Set Method

1: Find feasible initial active-set partition \( \mathcal{A}_0, \mathcal{A}_0, \mathcal{I}_0 \).
2: \textbf{while} \( \mathcal{A}^+ \neq \mathcal{A} \) or \( \overline{\mathcal{A}}^+ \neq \overline{\mathcal{A}} \) \textbf{do}
3: \quad Solve subspace minimization problem (4.5)
4: \quad Update duals using (4.6)
5: \quad Update active-sets \( \mathcal{A} \) and \( \overline{\mathcal{A}} \) using (4.7)
6: \textbf{end while}

In previous studies of the algorithm, the active sets are initialized by randomly sorting the indices \( J \) into the sets \( \mathcal{A}, \mathcal{A} \), and \( \mathcal{I} \), and then followed by a feasibility adjustment algorithm to ensure that the active-set partition satisfies \( \{ z_T | Cz = d \} \neq \emptyset \). In Section 4.2 we propose an initialization scheme for our MPC problem.

Remark 7. While matrix factorization updates for solving (4.5) are easy for traditional active-set updates, the same updates do not simply apply to this primal-dual active-set algorithm as more than one index changes active sets per iteration. Iterative solutions are a possibility, but are less numerically stable.

Application to Model Predictive Control

We apply the primal-dual active-set algorithm to the following linear model predictive control problem:

\[
\begin{align*}
\min_{x, u} & \quad x(N)^T P x(N) + \sum_{k=0}^{N-1} (x(k)^T Q x(k) + u(k)^T R u(k)) \\
\text{s.t.} & \quad x(k+1) = A x(k) + B u(k) + d(k) \\
& \quad x \leq x(k) \leq \bar{x}, \ k = 1, \ldots, N \\
& \quad u \leq u(k) \leq \bar{u}, \ k = 0, \ldots, N - 1
\end{align*}
\] (4.8a)
where $x(k) \in \mathbb{R}^n$ is the predicted state under the control action $u(k) \in \mathbb{R}^m$ and forecasted disturbance $d(k) \in \mathbb{R}^n$ over the horizon $N$, and symmetric $Q, R, P > 0$. We use the shorthand notation $X = [x(1)^T, \ldots, x(N)^T]^T$ and $U = [u(0)^T, \ldots, u(N-1)^T]^T$. We assume the given problem is recursively feasible.

It is straightforward to show that (4.8) can be reformulated as (4.1). The model predictive control problem (4.8) can be posed as a quadratic program in the standard form (4.1). The decision variables $z$ are the state $x(k)$ and input $u(k)$ trajectories, $z = [x(1)^T \ldots x(N)^T \ u(0)^T \ldots u(N-1)^T]^T$. (4.9)

The dynamics (4.8b) in the model predictive control problem (4.8) produce the equality constraints (4.2b) in the quadratic program (4.1). The equality constraint matrix $C$ and vector $d$ are given by

$$C = 
\begin{bmatrix}
I & 0 & \ldots & 0 & -B & 0 & \ldots & 0 \\
-A & I & \ldots & 0 & 0 & -B & 0 & \ldots & 0 \\
0 & \ldots & \ddots & \vdots & \vdots & \ddots & \vdots & \ldots & 0 \\
0 & \ldots & -A & I & 0 & 0 & \ldots & -B
\end{bmatrix}$$  (4.10a)

$$d = [(Ax(0) + d(0))^T \ d(1)^T \ldots d(N-1)^T]^T.$$  (4.10b)

The cost function (5.10) in the model predictive control problem (4.8) produces the quadratic cost in the quadratic program (4.1). The Hessian matrix $H$ is given by

$$H = \text{diag}(2Q \ldots 2Q 2P 2R \ldots 2R).$$  (4.11)

Finally the lower and upper bounds $\underline{z}$ and $\overline{z}$ on the decision variables $z = [X^T, U^T]^T$ are given by

$$\underline{z} = [\underline{x}^T \ldots \underline{x}^T \ \underline{u}^T \ldots \underline{u}^T]^T$$  (4.12a)

$$\overline{z} = [\overline{x}^T \ldots \overline{x}^T \ \overline{u}^T \ldots \overline{u}^T]^T.$$  (4.12b)

Note that if a particular state or input does not have a lower or upper bound, we can use $\pm \infty$ as the bound on the state.

**Theorem 2.** Any linear model predictive control problem with strictly convex quadratic cost and polytopic constraints can be posed as a quadratic program of the form (4.1) with $H > 0$, e.g. by adding slack variables and an augmented lagrangian penalty.

**Proof.** A linear MPC problem with strictly convex quadratic cost and polytopic constraints is written as:

$$\min_z \ \frac{1}{2} z^T H z + c^T z$$  (4.13a)
CHAPTER 4. A PRIMAL-DUAL ACTIVE-SET METHOD FOR DISTRIBUTED MODEL PREDICTIVE CONTROL

\[ s.t. \quad Cz = d \quad (4.13b) \]
\[ Kz \leq k \quad (4.13c) \]

where \( z \) is the stacked states and inputs \((4.9)\), constraint \((4.13b)\) includes the dynamics and any other equality constraints, and \((4.13c)\) includes any inequality constraints. To convert \((4.13)\) into \((4.1)\), the polytopic constraint \((4.13c)\) must be reformulated as a combination of equality constraints and bound constraints and the strict convexity of the cost must be preserved. We next describe an approach that satisfies these requirements.

First, add a slack variable \( s \) to the polytopic constraint \((4.13c)\):

\[ Kz + s = k \quad (4.14a) \]
\[ s \geq 0. \quad (4.14b) \]

Then, \((4.14a)\) are equality constraints, and \((4.14b)\) are lower bound constraints on the variable \( s \). The constraints therefore match the form of constraints in problem \((4.1)\). However, the full vector of decision variables has been augmented to \( z_{\text{aug}} = [z^T s^T]^T \), and the Hessian is only positive semi-definite. To circumvent this issue, add an augmented Lagrangian penalty for equation \((4.14a)\) to the cost where \( \rho \geq 1 \):

\[
J = \frac{1}{2} z^T H z + c^T z + \frac{\rho}{2} ([K \ I] [z \ s]^T - k)^T ([K \ I] [z \ s]^T - k). \quad (4.15)
\]

The cost \( J \) is equivalent to the original cost for any feasible point of the QP \((4.13)\). Moreover, the Hessian of \( J \) is now positive definite. The Hessian of \( J \) \((4.15)\) is:

\[
\begin{bmatrix}
H & 0 \\
0 & 0
\end{bmatrix}
+ \rho [K \ I]^T [K \ I] =
\begin{bmatrix}
H + \rho K^T K & K^T \\
K & I
\end{bmatrix} \quad (4.16)
\]

By Schur complement, the Hessian \((4.16)\) is strictly positive definite if and only if:

\[
H + \rho K^T K - K^T IK \succ 0 \quad (4.17a)
\]
\[
I \succ 0 \quad (4.17b)
\]

The first condition is true by definition since \( H \succ 0 \) and \((\rho - 1) K^T K \succeq 0 \). The second condition is trivially true. \( \square \)

Remark 8. From Theorem 2, the proposed primal-dual active-set algorithm can be used for a linear model predictive control problem more general than form \((4.8)\).

4.2 A Distributed Implementation on Star Communication Networks

In this section, we present a distributed implementation of the primal-dual active-set algorithm for large-scale systems with star communication networks. We describe the algorithm
for a general network flow problem, as depicted in Figure 4.1. For clarity, we refer to the network in Figure 4.1 as the dynamics network. In such a dynamics network, node $i$ has a state of dimension $n_i$ with linear dynamics and disturbances. The flow over the links of the dynamics network are the control inputs, where each link input has dimension $m_i$. This abstraction is suitable for a variety of large-scale problems, including thermal building control, optimal power flow, and multi-agent coordination. The connectivity of the dynamics network is arbitrary, but the communication is completely separate from the coupling of the dynamics.

![Figure 4.1: Network Flow Dynamic Coupling Example](image)

Star communication networks are very commonly implemented in practice as they are more robust than other communication topologies [86]. In fact, many networked systems or multi-agent systems are configured in such a way that embedded processors exist at each of the nodes of the network and a larger central computer exists for coordinating the nodes or agents of the network, as in Figure 4.2. Both dual decomposition and ADMM are commonly designed for star communication networks [17], [87], [16], [36]. Typically, such a distributed algorithm goes through iterations of computations in parallel at the leaf nodes, computation at the central hub, and communications between the leaf nodes and central hub. We next describe how the primal-dual active-set algorithm also fits this paradigm.

![Figure 4.2: Star Network Communication Graph](image)
Distributed Model Predictive Control Problem

We begin by characterizing the distributed model predictive control problem. The dynamics network consists of \( r \) nodes and \( s \) links, with \( n = \sum_{i=1}^{r} n_i \) being the total number of states and \( m = \sum_{i=1}^{s} m_i \) being the total number of inputs at time step \( k \), as in (4.8). Each node \( i \) of the dynamics network has its own state of dimension \( n_i \) with matrices \( A_i \in \mathbb{R}^{n_i \times n_i} \) and \( B_i \in \mathbb{R}^{n_i \times m} \) and is connected to the set of links \( \mathcal{L}(i) \). Each node \( i \) has linear dynamics:

\[
   x_i(k+1) = A_i x_i(k) + B_i u(k) + d_i(k) \tag{4.18}
\]

The \( B_i \) matrix is sparse and describes the coupling of node \( i \) to other nodes via the inputs in the dynamics; e.g. if describing network flow where each node has a scalar state \((n_i = 1)\) and the link flows are scalar inputs \((m_i = 1)\), the number of nonzeros in \( B_i \) is equal to \(|\mathcal{L}(i)|\).

Synthesizing problem (4.8), \( x(k) \) is a vector of the stacked states of the nodes at time step \( k \), \( u(k) \) is a vector of the stacked link inputs at time \( k \), \( A \) is block diagonal composed of \( r \) sub-blocks \( A_i \), and \( B \) is a vector of the stacked \( B_i \) matrices:

\[
   x(k) = \begin{bmatrix} x_1(k) \\ \vdots \\ x_r(k) \end{bmatrix}, \quad u(k) = \begin{bmatrix} u_1(k) \\ \vdots \\ u_s(k) \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & 0 & \cdots \\ 0 & \ddots & \vdots \\ \vdots & \cdots & A_r \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ \vdots \\ B_r \end{bmatrix}. \tag{4.19}
\]

Remark 9. Coupling in the \( A \) matrix between subsystems can be handled by moving coupling into the disturbance and designing a robust controller; i.e. for each subsystem \( i \), the mean of coupling term \( \sum_{j \neq i} A_{ij} x_j \) is included in disturbance \( d_i \) and its deviations are bounded and robustly addressed. See [11], [15], [88].

Additionally, the cost matrices \( P, Q, \) and \( R \) of (5.10) are block diagonal:

\[
   P = \begin{bmatrix} P_1 & \cdots \\ \vdots & \ddots \end{bmatrix}, \quad Q = \begin{bmatrix} Q_1 & \cdots \\ \vdots & Q_r \end{bmatrix}, \quad R = \begin{bmatrix} R_1 & \cdots \\ \vdots & R_s \end{bmatrix}. \tag{4.20}
\]

Each state and input has box constraints as in problem (4.8) which are naturally decoupled.

Distributed Primal-Dual Active-Set Algorithm

The splitting of the algorithm for the distributed model predictive control of a network flow problem is as follows. The \( i \)-th leaf node is responsible for the states corresponding to sub-block \( A_i \) and the corresponding dual variables \( \lambda \) and \( \overline{\lambda} \). The \( s \) components of the control \( u \) are each computed at a node neighboring that link; the exact assignment of each link to a node depends on the networked system. For example, in Figure 4.1, control input \( u_i(k) \) could be assigned to either node \( i \) or node \( j \). In a network flow with star configuration (i.e. input coupling matches the communication), each link could be assigned to its corresponding
CHAPTER 4. A PRIMAL-DUAL ACTIVE-SET METHOD FOR DISTRIBUTED MODEL PREDICTIVE CONTROL

leaf node. Note that multiple links may be assigned to a single node. The corresponding dual variables $\lambda$ and $\bar{\lambda}$ are similarly assigned to the same node as $u_i$.

The distributed primal-dual active-set method is summarized in Algorithm 3. This splitting of the computation matches the star network communication graph (see Figure 4.2) where each leaf node has a simple embedded processor and the central hub has a much larger computing platform. The two communication operations used are gather and scatter, as seen in Message Passing Interface (MPI) terminology. A gather operation takes vector pieces from the leaf nodes and aggregates them into one large vector at the central hub. A scatter operation takes a vector at the central hub and sends each piece of that vector to the appropriate leaf node. The distributed algorithm begins with the active sets $\overline{A}$, $A$, and $I$ initialized at the leaf nodes. Each leaf node sends its active-sets up to the central hub, where the full $\overline{A}$, $A$, and $I$ sets are gathered. The central hub computes the subspace minimization problem (4.5). Then, the computed $z_I$ and $\nu$ are scattered to the leaf nodes: each leaf node receives its relevant elements of $x(k)$ and $u(k)$ that were also elements of $z_I$. The $i$-th leaf node then computes its components of dual variables $\lambda$ and $\bar{\lambda}$.

Algorithm 3 Distributed Primal-Dual Active-Set

2: while $A^+ \neq \overline{A}$ or $\overline{A}^+ \neq \overline{A}$ do
3: Uplink: Send current active-set to central hub.
4: Central Hub: Solve subspace min problem (4.5)
5: Downlink: Send $z_I$ and $\nu$ to leaf nodes.
6: Leaf Node $i$: Update the leaf’s components of dual variables $\lambda$ and $\bar{\lambda}$ using (4.6).
7: Update active-sets $\overline{A}$, $A$ by (4.7).
8: end while

In [73], the primal-dual active-set is initialized by randomly dividing the elements of $J$ into $\overline{A}, A, I$, and then proceeding with a feasibility checking algorithm to make sure $Z_I$ (4.4) is not empty. This feasibility checking algorithm is centralized. Another possible simple initialization algorithm would be to assign all indices to the set $I$ in the initialization. This would satisfy (4.2b) if the problem (4.1) is feasible. The downside to this approach however, is that the subspace minimization subproblem (4.5) is at its largest possible size.

In this chapter, we propose a simple initialization procedure for the MPC problem (4.8) with structure defined in (4.19) and (4.20) that is completely decentralized; i.e. the nodes can initialize the active sets without any communication. The algorithm applies to problems where $m \geq n$ and $B$ is full row rank. The main idea is to set all inputs $u(k) = 0$ for all $k$ over the prediction horizon. Then, $x(k)$ are computed from the dynamics (4.8b) which are now decoupled. Then, the components of $x(k)$ are sorted into active sets. The initialization algorithm is summarized in Algorithm 4.

The intuition behind this algorithm is that the initial active-sets indicate a node’s request for control action; if the initial active-sets are all empty, then all actuators should do nothing. This initialization method was used in previous work to achieve a one-step communication
solution for a distributed model predictive controller [52]. The idea of nodes requesting control action comes from the logic exhibited in the “Trim and Respond” algorithm in [52]. Moreover, this algorithm is particularly suited for problems where the penalty on $u$ is higher than the penalty on $x$, i.e. $R > Q$, and the number of links is greater than the number of nodes, i.e. $s \geq r$. If $R$ dominates $Q$ in the cost function, then the optimal solution should have values of $u(k)$ close to 0.

Algorithm 4 Initialization Algorithm

1: Set $u(k) = 0 \ \forall k$.
2: Propagate node dynamics (4.18) to obtain states $x(k) \ \forall k = 0, \ldots, N - 1$.
3: $\mathcal{A}_0 = \{ j = (k - 1)n + i \mid x_i(k) > \bar{x} \}$
4: $\mathcal{A}_0 = \{ j = (k - 1)n + i \mid x_i(k) < \underline{x} \}$
5: $\mathcal{I}_0 = \{ j = (k - 1)n + i \mid i \notin \mathcal{A}_0 \cup \mathcal{A}_0 \}$

Proposition 1. Consider a network flow problem where $m \geq n$, and $B$ is full row rank. Assume, (4.8) is feasible. Then, Algorithm 4 necessarily satisfies $Z_I \neq \emptyset$.

Proof. Examining (4.10a), $C \in \mathbb{R}^{Nn \times N(n+m)}$ is a full row rank matrix because $B$ is full row rank and $C$ is wide. Algorithm 4 saturates some elements of $x(k)$; this removes the corresponding columns from the first $Nn$ columns of $C$ so that $C_I z_I = d - C_A z_A$, where $z_I$ are free variables. The matrix $C_I$ always includes the sub-matrix diag$(-B, -B, \ldots, -B)$ after Algorithm 4 is executed, so $C_I$ is full row rank. Therefore, there exists some $z_I$ such that $C_I z_I = d - C_A z_A$.

Such an initialization as algorithm 4 is only necessary for the first step of the MPC solution as a warm-start approach can be utilized in subsequent time steps. However, the algorithm is particularly of use for short horizon problems, where structure of the problem may be further exploited to come up with simpler algebraic solutions, as was done in [52].

Alternating Direction Method of Multipliers

In this section, we present a sparse distributed implementation of the Alternating Direction Method of Multipliers (ADMM). The form of ADMM algorithm presented here is very similar to that in [36], except we take advantage of the sparse structure of the problem in order to reduce computation time. In Section 4.3 we will compare this algorithm with our distributed primal-dual active-set algorithm. We next describe the steps of ADMM.

First, we formulate the augmented Lagrangian, where variable $y$ is a copy of $z$ used to decouple the dynamics and the constraints:

$$L_\rho(y, z, \mu) = \frac{1}{2} y^T H y + I_h(y) + I_K(z) + \mu^T (y - z) + \frac{\rho}{2} \|y - z\|^2. \quad (4.21)$$
Note that the variables $y$ and $z$ are not exact copies throughout the algorithm iterations, but are meant to approach consensus at the termination of the algorithm. The indicator functions $I_A(z)$ and $I_K(y)$ are defined as follows:

$$I_A(y) = \begin{cases} 0 & \text{if } Cy = d \\ \infty & \text{otherwise} \end{cases} \quad (4.22)$$

$$I_K(z) = \begin{cases} 0 & \text{if } \bar{z} \leq z \leq \overline{z} \\ \infty & \text{otherwise} \end{cases} \quad (4.23)$$

The parameter $\rho$ is a weighting parameter that can be tuned to alter the number of steps to convergence. There exist a few rules in the literature for choosing this parameter $\rho$ [30], [29]. These rules are optimal for a formulation of ADMM where the state variables have been eliminated by substitution. To the best of our knowledge, an optimal parameter $\rho$ has not yet been found for the formulation presented here.

The steps of ADMM are then as follows, where the superscript $t$ denotes the algorithm’s iteration number:

$$y^{t+1} = \arg \min_y L_\rho(y, z^t, \mu^t) \quad (4.24a)$$

$$z^{t+1} = \arg \min_z L_\rho(y^{t+1}, z, \mu^t) \quad (4.24b)$$

$$\mu^{t+1} = \mu^t + \rho(y^{t+1} - z^{t+1}) \quad (4.24c)$$

The computations required at each step of ADMM can be written more explicitly for our class of problem. The $y$-update step (4.24a) can be computed by solving the following linear equation:

$$\begin{bmatrix} H + \rho I & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} y^{t+1} \\ \nu^{t+1} \end{bmatrix} = \begin{bmatrix} -\mu^t + \rho z^t \\ d \end{bmatrix}. \quad (4.25)$$

where $\nu$ is a dual variable for the equality constraints (4.1b). Note that the left-hand side KKT matrix is sparse. We can store the $LDL$ decomposition of this matrix offline and use this to compute the solution of (4.25) more quickly than refactorizing at every iteration. More detail is provided in the theoretical analysis provided in Section 4.3.

Next, the $z$-update in step (4.24b) can be computed by a simple element-wise projection, where $[\cdot]_i$ is indexing the $i^{th}$ element of a vector:

$$z_i^{t+1} = \begin{cases} z_i & \text{if } [y^{t+1} + \frac{1}{\rho}\mu^t]_i < z_i \\ [y^{t+1} + \frac{1}{\rho}\mu^t]_i & \text{if } \bar{z} \leq [y^{t+1} + \frac{1}{\rho}\mu^t]_i \leq \overline{z} \\ \overline{z} & \text{if } [y^{t+1} + \frac{1}{\rho}\mu^t]_i > \overline{z} \end{cases} \quad (4.26)$$

Convergence occurs when primal and dual residuals go to zero. The primal residual $r$ and dual residual $d$ are defined

$$r^{t+1} = y^{t+1} - z^{t+1} \quad (4.27)$$
and the algorithm is said to have converged when the magnitude of these residuals is less than some epsilon, i.e. \( \|r\| < \epsilon, \|d\| < \epsilon \).

The distributed implementation of the ADMM algorithm is then very similar to the distributed implementation of the primal-dual active-set algorithm (c.f. Algorithm 3). The splitting of states and inputs to the nodes of the network is the same as in Section 4.2. The \( y \) update (4.25) occurs at the central hub. The current iteration of \( y \) is then split up into \( x(k) \) and \( u(k) \) and scattered to the appropriate leaf nodes. Each leaf node computes an update for its elements of \( z \) and its elements of \( \mu \). Then, each leaf sends up its elements of \( z \) and \( \mu \) to the hub node where the \( y \) update can begin another iteration of the algorithm. The distributed implementation is summarized in Algorithm 5.

**Algorithm 5 Distributed ADMM**

1: Leaf Node \( i \): Initialize node \( i \)'s elements of \( z, \mu \).
2: while \( |r| > \epsilon \) or \( |s| > \epsilon \) do
3: Uplink: Aggregate components of \( z \) and \( \mu \) at hub.
4: Central Hub: Solve (4.25) for \( y \).
5: Downlink: Send components of \( y \) to leaf nodes.
6: Leaf Node \( i \): Update the node \( i \)'s components of \( z \) (4.26) and \( \mu \) (4.24c).
7: end while

**Dual Decomposition**

In this section, we present a fixed step-size dual decomposition algorithm. In Section 4.3 we will compare this algorithm with our decentralized primal-dual active-set algorithm.

Dual decomposition commonly uses a projected subgradient method to compute a solution [25], [16]. For (4.8), the projected subgradient method simplifies to a gradient projection method, where \([ \cdot ]_+ \) is a projection onto the nonnegative real numbers.

The algorithm initializes with feasible dual variables \( \lambda, \bar{\lambda}, \) and \( \nu \). Then, it updates the primal variables \( x(k) \) and \( u(k) \) based on the dual variables \( \lambda, \bar{\lambda}, \) and \( \nu \). Next, the dual variables \( \lambda, \bar{\lambda}, \) and \( \nu \) are updated based on the primal variables \( x(k) \) and \( u(k) \) and a constant step-size \( \alpha \). The algorithm terminates when the dual variable updates are sufficiently small, e.g. \( \|\lambda^{t+1} - \lambda^t\| < \epsilon \).

The steps of dual decomposition with a fixed step-size \( \alpha \) for the MPC problem (4.8) are described in Algorithm 6 where the \( t \)-th primal updates are computed by

\[
x(k)^t = \frac{1}{2} Q^{-1} (\lambda(k)^t - \bar{\lambda}(k)^t + A^T \nu(k+1)^t - \nu(k)^t) \quad \forall k < N \tag{4.29a}
\]

\[
x(N)^t = \frac{1}{2} P^{-1} (\lambda(k)^t - \bar{\lambda}(k)^t - \nu(k)^t) \tag{4.29b}
\]

\[
u(k)^t = \frac{1}{2} R^{-1} (\lambda(k)^t - \bar{\lambda}(k)^t + B^T \nu(k+1)^t) \tag{4.29c}
\]
and the \((t + 1)\)-th dual updates are computed by

\[
\begin{align*}
\Delta(k)^{t+1} &= \left[\Delta_k^t + \alpha(-x(k) + \bar{x})\right]_+ \quad (4.30a) \\
\Delta(k + N)^{t+1} &= \left[\Delta_{k+N}^t + \alpha(-u(k) + \bar{u})\right]_+ \quad (4.30b) \\
\bar{\lambda}(k)^{t+1} &= \left[\bar{\lambda}(k)^t + \alpha(x(k) - \bar{x})\right]_+ \quad (4.30c) \\
\bar{\lambda}(k + N)^{t+1} &= \left[\bar{\lambda}(k + N)^t + \alpha(u(k) - \bar{u})\right]_+ \quad (4.30d) \\
\nu(k)^{t+1} &= \nu(k)^t + \alpha(x(k) - Ax(k-1) - Bu(k-1) - d(k-1)) \quad (4.30e)
\end{align*}
\]

The splitting of the dual decomposition algorithm is the same as the primal-dual active-set method, except the dual variable \(\nu\) is computed on the leaf nodes. On the uplink, the dual variables \(\lambda\), \(\bar{\lambda}\), and \(\nu\) are sent from the leaf nodes to the hub. The primal updates of (4.29a)–(4.29c) are done at the central processor. On the downlink, the primal variables \(x(k)\) and \(u(k)\) are sent down to the leaf nodes. The \(i\)-th component of the dual updates (4.30a)–(4.30e) are done at the \(i\)-th leaf node.

Algorithm 6 Dual Decomposition Algorithm

Initialize dual variables, \(\bar{\Delta}(k)^t, \bar{\lambda}(k)^t, \nu(k)^t\) for iteration \(t = 0\).

while \(\|\nu(k)^{t+1} - \nu(k)^t\| > \epsilon \forall k \in \{1, \ldots, N\}\), OR \(\|\Delta(k)^{t+1} - \Delta(k)^t\| > \epsilon \forall k \in \{1, \ldots, 2N\}\), OR \(\|\bar{\lambda}(k)^{t+1} - \bar{\lambda}(k)^t\| > \epsilon \forall k \in \{1, \ldots, 2N\}\) do

Uplink: Send dual variables to the central hub.

Central Hub: Compute primal variables as in (4.29).

Downlink: Send primal variables to the leaf nodes.

Leaf Node \(i\): Update leaf’s components of the dual variables \(\bar{\Delta}, \bar{\lambda}, \nu\) by (4.30).

end while

4.3 Theoretical Analysis

In this section, we discuss and compare the theoretical computation and communication complexities of three distributed algorithms: the primal-dual active-set method, ADMM, and dual decomposition.

For a distributed algorithm on a star communication network the execution time is given by

\[
T = n_{iters}(\Delta t_{hub} + \Delta t_{leaf} + (m_{up} + m_{down})\Delta t_{delay}) \quad (4.31)
\]

where \(n_{iters}\) is the total number of iterations needed for the algorithm to converge, \(\Delta t_{hub}\) is the amount of time required to perform the hub computations, \(\Delta t_{leaf}\) is the amount of time required to perform the leaf computations, \(\Delta t_{delay}\) is the amount of time required to communicate a floating point between the hub and leaves, and \(m_{up}\) and \(m_{down}\) are the number of floating point variables passed between the central hub and leaf nodes. Note that
the communication time $\Delta t_{\text{leaf}}$ is the time it takes to communicate a floating variable within the context of a scatter or gather communication operation; this is significantly faster than sending or receiving messages one-at-a-time.

An implementation on distributed platforms must balance the tradeoff between distribution of computation and minimizing communication. The best distributed algorithm minimizes the execution time $T$. In this section we present a theoretical analysis showing the tradeoffs between the primal-dual active-set method, ADMM, and dual decomposition.

First, we compare the number of variables communicated $m_{\text{up}}$ and $m_{\text{down}}$ for each algorithm. The primal-dual active-set algorithm sends gathers active-sets $\mathcal{A}$, $\mathcal{A}$, and $\mathcal{I}$ from the leaves to the hub and scatters $z$ and $\nu$ from the hub to the leaves during each algorithm iteration. Therefore $m_{\text{up}} = N(n + m)$ and $m_{\text{down}} = N(2n + m)$ for the primal-dual active-set algorithm. ADMM scatters $y$ to the leaf nodes with $m_{\text{down}} = N(n + m)$ and receives $z$ and $\mu$ on the uplink for $m_{\text{up}} = 2N(n + m)$. Finally, the dual decomposition algorithm scatters primal variables $x, u$ from the hub to the leaves and gathers dual variables $\lambda, \bar{\lambda}, \nu$ from the leaves to the hub during each iteration. Therefore, for dual decomposition, $m_{\text{up}} = 2(n + m)N + nN$ and $m_{\text{down}} = N(n + m)$. Thus the primal-dual active-set algorithm requires the least communication per iteration. This result is amplified by the fact that the primal-dual active-set algorithm requires fewer iterations to converge, as discussed next.

Next, we compare the number of iterations $n_{\text{iters}}$ needed for convergence for each algorithm. For ADMM and dual decomposition, the rate of convergence is locally linear [27], [89]:

$$\|z^{s+1} - z^*\| \leq \alpha\|z^s - z^*\|. \quad (4.32)$$

On the other hand, the primal-dual active-set has local superlinear convergence since it can be written as a semi-smooth Newton method (c.f. Theorem (3.1) of [72] and Appendix A of [73]). Therefore, the primal-dual active-set algorithm requires fewer iterations than the dual decomposition and ADMM algorithms.

Finally, we compare the order of magnitude of the computation time required for each algorithm at the hub and at the leaves. The size of the primal-dual algorithm subproblems varies depending on the size of the active sets, so we review the worst case order of magnitude computation time. If $\mathcal{A}$ is empty, then the primal-dual algorithm requires $O((N(n + m))^3)$ time to solve the subspace minimization in the hub processor. The dual variables require worst case $O(N(n + m))$ to update (4.6) and only $O(N)$ to update the active-sets in each leaf node, (4.7).

For ADMM, the size of the hub and leaf computations do not vary in size. This is useful because we can precompute offline and store the $LDL^T$ factorization of the sparse KKT matrix used to solve (4.25). This reduces equation (4.25) to $O((N(n + m))^2)$, an order of magnitude less than a naive linear algebra solution. The leaves need only do a simple box projection and a few scalar multiplications and additions for $O(N)$ variables, so each leaf computes in $O(N)$ time. Dual decomposition requires $O(Nn(n + m))$ time to update the primal variables in the central hub and $O(N(n + m))$ to update the dual variables in each leaf node. Thus dual decomposition has a lower computational cost at the hub and leaves.
The results of our analysis are summarized in Table 4.1 where “DD” is an abbreviation for dual decomposition and “PDAS” for primal-dual active-set. The primal-dual active-set has the worst possible computation time at the hub, but does better than the other two algorithms on the size of the messages $m_{up} + m_{down}$ sent at each iteration. The splitting of the computation scales more favorably for the dual decomposition because the central node computation has a smaller order.

In general, dual decomposition performs best when the communication delay time $\Delta t_{delay}$ is negligible and the computational power of the hub is limited. This is the opposite of the situation found in typical networked systems. In practice, we find that the primal-dual active-set method performs best because $(m_{up} + m_{down})\Delta t_{delay} \gg \Delta t_{hub} + \Delta t_{leaf}$. ADMM total time $T$ is comparable to the primal-dual active-set method if the number of iterations happens to be small; i.e. the parameter $\rho$ is chosen such that there are few iterations.

<table>
<thead>
<tr>
<th>Alg</th>
<th>DD</th>
<th>PDAS</th>
<th>ADMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub</td>
<td>$O(N(n+m))$</td>
<td>$O((N(n+m))^3)$</td>
<td>$O((N(n+m))^2)$</td>
</tr>
<tr>
<td>Leaf</td>
<td>$O(N(n+m))$</td>
<td>$O(N(n+m))$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>$m_{up}$</td>
<td>$N(3n+m)$</td>
<td>$N(n+m)$</td>
<td>$2N(n+m)$</td>
</tr>
<tr>
<td>$m_{down}$</td>
<td>$N(n+m)$</td>
<td>$N(2n+m)$</td>
<td>$N(n+m)$</td>
</tr>
</tbody>
</table>

Table 4.1: Theoretical Trade-offs

### 4.4 Example: Thermal Building Control

Building dynamics are commonly described as having decoupled states and coupled inputs, as seen in Chapter 2 and [52], [56]. In practice, dynamic coupling in the dynamics is either negligible or is modeled by adding it into each node’s disturbance $d_i(k)$. The dynamics can therefore be described by equation (4.18), where $x_i(k)$ is the temperature of zone $i$ at time $k$, $u_m(k) \leq 0$ is the central cooling power input, $u_i(k) \geq 0 \forall i < m$ is local heating power input, and $d_i(k)$ is the thermal load on the zone. The $B$ matrix structure looks like:

$$B = \begin{bmatrix}
  b_{11} & 0 & \ldots & b_{1m} \\
  0 & b_{22} & \ldots & b_{2m} \\
  \vdots & \ddots & \ddots & \vdots \\
  0 & \ldots & b_{m-1,m-1} & b_{m-1,m}
\end{bmatrix}$$

(4.33)

The centrally produced cooling $u_m(k)$ couples the dynamics of the thermal zones. The building has $n$ thermal zones and $m = n + 1$ control inputs.

The constraints on the system are comfort bounds on the temperature $[\bar{x}, \bar{x}]$ and control input saturation limits $[\bar{u}, \bar{u}]$. The MPC cost function is a weighted sum of the squared inputs which approximates the power consumption. Thus, the building control problem can
be formulated as a standard MPC problem of the form (4.8) with structure (4.19), (4.20). Splitting of the problem is as described in Section 4.2, with the extra $u_m$ stored at the central node since it physically corresponds to the central cooling.

The standard building communication protocol is BACnet (Building Automation and Control networks) [63]. We consider the BACnet local area network (LAN) option of Attached Resource Computer Network (ARCnet). ARCnet’s communication in a building is a star network, with one node acting as a central hub for communication, and the rest acting as leaf nodes [64] (c.f. Figure 4.2). This central node can use BACnet service ReadProperty-Multiple which can read a data property from multiple nodes at a time. This occurs on the uplink pictured in Figure 4.2. Similarly, the central hub can use WriteProperty-Multiple to write data on the downlink pictured in Figure 4.2.

Studies done to estimate delay time $\Delta t_{delay}$ for the ReadProperty-Multiple service on an ARCNET network come up with delay times of about 0.0317 seconds per double precision floating point to 0.39155 seconds per double precision floating point from a study of up to 30 nodes [64]. Therefore, we assume that the delay in such a communication network can range from approximately 0.03 seconds per point to almost 0.4 seconds per point.

4.5 Results

In this section we compare numerical results for total computation time and communication time of the distributed primal-dual active-set method with distributed algorithms ADMM and dual decomposition. We also compare the performance of the active-set algorithm relative to another popular active-set algorithm. Numerical experiments were conducted in Matlab on an Ubuntu desktop with a 3.6 GHz Intel Core i7 processor. The primal-dual active-set algorithm is coded in Matlab and thus utilizes the MA57 LDL solver provided by the Harwell Subroutines Library (HSL) [90] to compute the linear algebra in the subspace minimization problem (4.5).

For our comparisons, we examine two test MPC problems of the form (4.8). In the first test problem, the dynamic coupling in the $B$ matrix matches building HVAC dynamics as described in Section 4.4. The state is a deviation from a nominal setpoint temperature. In the second test problem, we construct a random network flow system where the $B$ matrix is randomly generated with condition numbers varying from 1 to 1000. We use the best-case communication delay time of $\Delta t_{delay} = 0.03$ seconds per floating point to estimate communication delay. The rest of the problem parameters are summarized in Table 4.2. Note that the $A$ matrix, the disturbance $d$ and the initial state are randomized to test the limits of the algorithm. The disturbance forecast is assumed to be known.

Numerical Comparison of Distributed Algorithms

In this section we present experimental results to compare the execution time $T$ for the primal-dual active-set method, ADMM, and dual decomposition. All algorithms converge
A
\[ d_i(k) \]
\[ [−2, 2] \]
\[ [−5, 5] \]
\[ [−3, 3] \]
\[ P, Q \]
\[ 0.1I_n \]
\[ R \]
\[ 10I_n \]
\[ N \]
\[ 5 \]

Table 4.2: Experimental Problem Parameters

Table 4.3: Numerical Performance for HVAC problem (times values are in units of seconds)

to the same optimal solution, given enough iterations.

Each result is averaged from a set of 10 trials, with the difference between trials being
different initial conditions and disturbance profiles. The max number of iterations allowed
is 10000. The precision required for convergence is \( \epsilon = 1 \times 10^{-8} \). Note that the precision of
the primal-dual active-set algorithm cannot be limited in the same way as ADMM and dual
decomposition. The precision for the active-set algorithm is the precision used for the matrix
inverse in the subspace minimization problem. It is hard to assert that this precision is at
all similar to the stopping criterion precision we specify for ADMM and dual decomposition.

The results are presented in Tables 4.3, 4.4 where

\[ T_{\text{comp}} = n_{\text{iters}} (\Delta t_{\text{hub}} + \Delta t_{\text{leaf}}) \]
\[ T_{\text{delay}} = n_{\text{iters}} (m_{\text{up}} + m_{\text{down}}) \Delta t_{\text{delay}} \]

and all time measurements are presented with the unit of seconds. The total problem size
\( N(n + m) \) is enumerated in the left hand column. We do not include the time to compute
the \( LDL \) factorization and the \( \rho \) computation in the ADMM computation time since this
can be done offline before the solver starts. We use the heuristic proposed for \( \rho \) in [30]. We
abridge our results for the dual decomposition algorithm, as more detailed results can be
found in [53]. Note that the dual decomposition algorithm reaches the maximum number of
iterations for the larger problems; we do not present the numerical results for these larger
problems for dual decomposition since the solution is not optimal.

The data for \( \Delta t_{\text{hub}} \) and \( \Delta t_{\text{leaf}} \) are also presented in Tables 4.3 and 4.4 for the primal-
dual active-set and ADMM algorithms in order to support the theoretical analysis presented
in the previous subsection. Even though \( \Delta t_{\text{hub}} \) is significantly smaller for ADMM, the net
CHAPTER 4. A PRIMAL-DUAL ACTIVE-SET METHOD FOR DISTRIBUTED MODEL PREDICTIVE CONTROL

Table 4.4: Numerical Performance for Random problem (times values are in units of seconds)

<table>
<thead>
<tr>
<th>Problem size</th>
<th>PDAS</th>
<th>ADMM</th>
<th>DD</th>
</tr>
</thead>
<tbody>
<tr>
<td>N(n+m)</td>
<td>T</td>
<td>n_{iters}</td>
<td>\Delta t_{hub}</td>
</tr>
<tr>
<td>50</td>
<td>3.76</td>
<td>1.1</td>
<td>0.002</td>
</tr>
<tr>
<td>100</td>
<td>9</td>
<td>14.0045</td>
<td>10.1</td>
</tr>
<tr>
<td>500</td>
<td>90</td>
<td>2.1</td>
<td>0.042</td>
</tr>
<tr>
<td>1000</td>
<td>157.9</td>
<td>2</td>
<td>0.18</td>
</tr>
<tr>
<td>5000</td>
<td>776.77</td>
<td>2</td>
<td>10.5</td>
</tr>
<tr>
<td>10000</td>
<td>1753</td>
<td>2</td>
<td>62.7</td>
</tr>
</tbody>
</table>

Figure 4.3: Communication and Execution Times vs Problem Size

Computation time \( T_{\text{comp}} \) is much larger than that of the primal-dual active-set method simply because \( n_{\text{iters}} \) is much larger.

In practice, the number of iterations \( n_{\text{iters}} \) is many orders of magnitude less for the primal dual active-set than for ADMM. From these results, it is quite clear that the impact of communication on solver time is not negligible – even in the best case scenario for communication delay \( \Delta t_{\text{delay}} \). The worst case result for communication delay for ADMM is on the order of days for the HVAC problem and the order of hours for the random problem. This result would be dramatically worse for the worst case communication latency of \( \Delta t_{\text{delay}} = 0.4 \) seconds per point.

Comparison of Active-Set Algorithms

We compare the primal-dual active-set algorithm to another active-set algorithm, qpOASES [91]. The comparison done here is completely centralized since the qpOASES algorithm is centralized.

First, we present \( n_{\text{iters}} \), the average number of iterations, and \( T_{\text{comp}} \), the average solver time for qpOASES in Table 4.5. We employ the Algorithm 4 to initialize the active sets.
Table 4.5: Numerical Performance of qpOASES (times values are in units of seconds)

for both algorithms for Table 4.5 results. This method is employed to showcase the worst-case performance of the algorithm. The qpOASES algorithm is considerably slow for the large-scale problems.

In addition, we compute a closed-loop profile for the HVAC problem over one day. The sampling time is 15 minutes for this system, and the prediction horizon is \( N = 20 \). We include the closed-loop profile for reference in Figure 4.4, where the temperature is the deviation from setpoint, and the input is a power input. Both active-set algorithms compute this same closed-loop profile.

For the closed-loop numerical results, we employ a warm-starting active-set initialization method beyond the first time step. The thermal load was computed from data from the Doe Library at the University of California, Berkeley in the month of December. Numerical results are summarized in Table 4.6, where \( T_{avg} \) is the centralized computation time averaged over all time steps. We find that on average, with warm-starting, the primal-dual active-set method outperforms the qpOASES algorithm by one to many orders of magnitude, even though the number of iterations to find the new active-set is comparable. The improved performance of the proposed active-set method relative to qpOASES is linked to sparse linear algebra vs. dense linear algebra. We give a detailed explanation next.

The linear algebra used in qpOASES solves a dense version of the KKT matrix (analogous to (4.5)) where the state variables have been eliminated by substitution; this results in fill-in of the Hessian and the constraints. The linear algebra in the initialization of qpOASES is a null-space method which requires a dense TQ factorization and a dense Cholesky factorization of reduced sized matrices [85], [91]. Each of these factorizations cost \( O(n_{null}^3) \), where \( n_{null} \approx \frac{1}{3}n_{large} \) and \( n_{large} = N(2n + m) \) is the original size of the KKT matrix in (4.5). Subsequent iterations of qpOASES compute factorization updates at only quadratic cost, \( O(n_{reduced}^2) \), where \( n_{reduced} \approx \frac{2}{3}n_{large} \).

On the other hand, the main linear algebra of the PDAS algorithm is the LDL factorization of a very sparse KKT matrix. The complexity of the dense computation of an LDL factorization is \( O(\frac{1}{3}n_{large}^3) \), but permutations are used to preserve sparsity of the LDL factorization and therefore achieve computation that is \( \ll O(\frac{1}{3}n_{large}^3) \). Many sources in the literature point to the practical advantages of sparse linear algebra. For reference, [92] cites an example where sparse Cholesky takes about 1% the number of flops of dense Cholesky.
Numerical performance of the sparse LDL factorization method provided by MA57 is reported in [90]. A similar note about the practical advantages of sparse LDL factorization and solve time is included in [87].

Comparing the two active-set algorithms, the initial factorizations in qpOASES are computationally intensive at $O(n_{\text{reduced}}^3)$ because the reduced problem size $n_{\text{reduced}}$ is still large; the sparse LDL factorization, $\ll O(\frac{1}{3}n_{\text{large}}^3)$, easily outperforms these dense factorizations. Additionally, the closed loop results shown in Table 4.6 suggest that in large-scale problems, the sparse LDL factorization, $\ll O(\frac{1}{3}n_{\text{large}}^3)$, also outperforms the factorization updates, $O(n_{\text{reduced}}^2)$, used in qpOASES. Thus, sparse linear algebra is a large factor affecting our numerical results. Adding a sparse linear algebra kernel to qpOASES may improve its performance.

Figure 4.4: Closed Loop Profiles

<table>
<thead>
<tr>
<th>N(n+m)</th>
<th>PDAS $n_{\text{iters}}$</th>
<th>PDAS $T_{\text{avg}}$</th>
<th>qpOASES $n_{\text{iters}}$</th>
<th>qpOASES $T_{\text{avg}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>340</td>
<td>1.5395</td>
<td>0.0025</td>
<td>1.5658</td>
<td>0.1162</td>
</tr>
<tr>
<td>2020</td>
<td>2.25</td>
<td>0.0098</td>
<td>2.3816</td>
<td>48.23</td>
</tr>
</tbody>
</table>

Table 4.6: Numerical Comparison of Active-Set Methods in Closed Loop
4.6 Conclusions

In our work, we have developed a novel distributed model predictive control algorithm that utilizes much less communication than other proposed distributed algorithms. The main conclusion of this research is that communication has a large impact on distributed algorithms. This idea is not new to the parallel computing community, and the work done here shows the need for careful consideration of algorithms in practical distributed control.

Moreover, this research indicates the importance of utilizing sparse linear algebra for large-scale problems. Although the proposed active-set algorithm had a similar number of algorithm iterations as the qpOASES algorithm, the numerical computation performance of the proposed algorithm was faster than that of qpOASES. The main computational bottleneck of both active-set algorithms was the linear algebra; the dense linear algebra of a reduced KKT system for qpOASES did not compute as quickly as the sparse linear algebra of the non-reduced KKT system in the primal-dual active-set method. The reason for dense linear algebra in qpOASES is that it is designed for small to medium scale systems; sparse linear algebra can greatly improve algorithm speed for large-scale problems.

The primal-dual active-set method described in this chapter shows impressive performance both in distributed form and in comparison to other active-set solvers. In a distributed setting, the primal-dual active-set method fits a star communication network with embedded processors at the leaf nodes and a larger centralized computer at the hub node, an architecture that is prominent amongst large-scale systems. Dual decomposition and ADMM are both commonly proposed for such systems, but the number of iterations required to communicate information is much too large to achieve practical performance on an HVAC building as well as on other systems with large communication delay times.
Chapter 5

Freeway Ramp Metering Systems

The objective of this chapter is to describe the way that ramp metering systems work in practice. We introduce the asymmetric cell transmission model to describe the dynamics of the freeway system. Then, we describe the sensors, computation platforms, and communication systems typically found in a ramp metering system. We summarize some ramp metering algorithms which are used in practice. We then describe how model predictive controllers are typically designed for ramp metering systems and discuss the previously proposed distributed control solutions for ramp metering.

5.1 The Asymmetric Cell Transmission Model

In this section, we describe the Asymmetric Cell Transmission Model (ACTM) [3] which is adopted throughout this thesis for modeling freeway traffic. For ACTM, the freeway is divided into $I$ segments where each segment can have at most one onramp and one offramp as depicted in Figure 5.1. The index set of all the segments is denoted as $\mathcal{I} \triangleq \{1, \ldots, I\}$.

For each segment $i$, the following quantities are defined:

- $n_i(k)$: Number of vehicles in segment $i$ at time step $k$. This is also referred to as the density of vehicles in segment $i$.

- $f_i(k)$: Number of vehicles leaving segment $i$, moving to segment $i + 1$ during time step $k$.

![Figure 5.1: Schematic of Freeway Segmentation](image)
CHAPTER 5. FREEWAY RAMP METERING SYSTEMS

- \( l_i(k) \): Number of vehicles queuing on the onramp of segment \( i \) at time step \( k \).
- \( r_i(k) \): Number of vehicles entering segment \( i \) through its onramp during time step \( k \).
- \( d_i(k) \): Number of vehicles entering onramp of segment \( i \) during time step \( k \), also known as demand.
- \( s_i(k) \): Number of vehicles leaving segment \( i \) through its offramp during time step \( k \).

In this model, mainline and onramp densities \((n_i(k) \text{ and } l_i(k), i \in \mathcal{I})\) are the states of the system. The following parameters of the model are assumed to be known a priori through a calibration process [93]. The parameters are positive, unless otherwise denoted.

- \( v_i \): Normalized free-flow speed \( \in (0, 1] \).
- \( \beta_i \): Split ratio of the offramp in segment \( i \in [0, 1] \).
- \( w_i \): Normalized congestion wave speed \( \in (0, 1] \).
- \( \gamma \): Blending coefficient of onramp flows \( \in [0, 1] \).
- \( n_c^i \): Critical density of segment \( i \), the density above which segment \( i \) is considered as congested.
- \( f_0(k) \): The exogenous demand entering link 1 (the most upstream segment) at time step \( k \).

The dynamics of the system states are obtained through mass conservation equations:

\[
\begin{align*}
    n_i(k+1) &= n_i(k) + f_{i-1}(k) + r_i(k) - s_i(k), \\
    l_i(k+1) &= l_i(k) + d_i(k) - r_i(k).
\end{align*}
\] (5.1) (5.2)

In ACTM, the parameter \( \beta \) is used to describe how much flow exits segment \( i \) through the offramp. Thus, \( s_i(k) = \beta_i(s_i(k) + f_i(k)) \), and the update rule of mainline densities can be rewritten as:

\[
n_i(k+1) = n_i(k) + f_{i-1}(k) + r_i(k) - f_i(k) \tilde{\beta}_i
\] (5.3)

where \( \tilde{\beta}_i = 1 - \beta_i \). Note that the onramp flows \( r_i(k) \) are determined by a controller (assuming that all onramps are actuated). The last piece of the traffic evolution model is the mapping function between mainline flows \( f_i(k) \) and densities \( n_i(k) \).

In freeway first order models, the flow in each segment \( i \) is restricted by the number of vehicles available to leave segment \( i \), \((\tilde{\beta}_i v_i(n_i(k) + \gamma r_i(k)))\), mainline capacity \((\bar{f}_i)\) and the available space in the downstream segment \( i+1 \), \((w_{i+1}(\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)))\). In other words, the flow in segment \( i \in \{1, \ldots, I-1\} \) is computed by (Figure 5.2):

\[
f_i(k) = \min\{\tilde{\beta}_i v_i(n_i(k) + \gamma r_i(k)), \ w_{i+1}(\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)), \bar{f}_i\}.
\] (5.4)
Note that the edge flows of the freeway system are special cases. The incoming flow \( f_0(k) \) is an exogenous input. The outgoing flow \( f_I(k) \) is simplified since \( u_{I+1} \) is undefined:

\[
f_I(k) = \min\{\bar{\beta}_i v_i (n_i(k) + \gamma r_i(k)) , \bar{f}_i\}.
\]  

(5.5)

Without loss of generality, we assume that:

\[
\bar{f}_i = \bar{\beta}_i v_i n_i^c = w_{i+1}(\bar{n}_{i+1} - n_{i+1}^c)
\]

(5.6)

where \( n_i^c \) is the critical density of segment \( i \) which is the density at which segment \( i \) transitions between congested and uncongested states. This is a common assumption in freeway modeling [94].

![Figure 5.2: Demand and Supply for Segment i](image)

The densities \( n_i(k) \), queue lengths \( l_i(k) \), flows \( f_i(k) \), and onramp inputs \( r_i(k) \) are bounded by box constraints. The states \( n_i(k), l_i(k), f_i(k) \) are non-negative since a negative density, queue length, or flow between segments are not physically possible. Similarly, the onramp inputs \( r_i(k) \) are restricted to be non-negative. In addition, each state and input has an upper bound, defined here:

- \( \bar{n}_i \): Jam density of segment \( i \), the maximum number of vehicles that can fit in segment \( i \).
- \( \bar{l}_i \): Max queue length of segment \( i \).
- \( \bar{f}_i \): Main-line capacity defined as the maximum number of vehicles that can leave segment \( i \).
- \( \bar{r}_i \): Maximum allowable ramp flows.

The density and queue lengths are restricted simply by space limitations on the freeway and onramp, respectively. The flows \( f_i \) and the onramp inputs \( r_i \) are restricted by the capacity of the freeway; only a certain amount of cars can move forward on the given freeway lanes in the sampling time of the model.
We note that other freeway models exist in the literature. For the most part, they are similarly based on mass conservation laws. ACTM is one of the most popular modeling options; METANET [95] is also fairly popular. The main difference between the models is that METANET possesses a higher order nonlinearity than ACTM.

5.2 Ramp Metering Systems

Ramp metering systems are composed of distributed sensors and controllers. In this section, we summarize the main sensors, actuators, computing platforms, and communication systems of a typical ramp metering system. More detailed descriptions of ramp metering system design can be found in [96, 97, 98].

![Ramp Metering Schematic](image)

A schematic of an on-ramp is shown in Figure 5.3. A queue sensor exists where the cars enter the on-ramp to detect cars entering the ramp queue. A passage sensor exists just after the stop line and ramp signal to measure when cars leave the ramp queue and enter the freeway. Algorithms use these two sensors to estimate the length of the queue [99, 100]. On the freeway, there is a mainline sensor located just upstream of the ramp that counts the cars that pass by it. This data is used to estimate the density of the freeway [101].

Each of these sensors and the ramp signal are connected to a controller box that is located near the ramp. A typical ramp controller has a 25 MHz Microprocessor as well as 8 MB
CHAPTER 5. FREEWAY RAMP METERING SYSTEMS

dynamic random access memory, 8 MB Flash memory, and 1 MB of static random access memory [102]. This controller is able to control multiple traffic devices and can manage multiple CPUs if desired. Each controller is connected by some form of communication to a centralized Traffic Management Center. The Traffic Management Center is a location in which the full ramp metering system can be monitored. In addition, computing resources for centralized ramp metering algorithms are available at the Traffic Management Center.

Communication Systems

The ramp controllers are connected to the Traffic Management Center by telephone line, fiber optics, microwave, radio frequency, or possibly a wireless communication system. Fiber-optic communication is highly favored as the choice of communication [97] because it has very high bandwidth capabilities. Fiber-optic technologies have been able to achieve an effective speed of 1.05 Petabit/s and have many advantages over electrical wiring present in the other suggested wired communication systems [103]. However, telephone lines are already present in cities and are cheaper than fiber-optic communications.

It is suggested in ramp metering handbooks that communications go directly between the controller boxes for coordinated control design. It is not clear how widely prevalent this communication design choice is in reality. Communication failure between the ramp metering controller boxes and the central traffic management center has been a challenge for some ramp metering implementations [104]. However, it is not clear what the communication system choice was in those implementations, or why the communication failures occurred.

5.3 Current Ramp Metering Practices

In practice, there are three categories of ramp metering strategies: fixed time metering, local ramp metering, and system wide metering. Fixed time metering is a schedule based on historical data and does not utilize feedback information. Local ramp metering adjusts the ramp metering signal based on local sensor information. Local ramp metering cannot account for information from other parts of the freeway, but it can be a good back up controller in the case of communication system failure. System wide metering is usually done by collecting information for the full freeway system at the Traffic Management Center and responding to that information. A table listing ramp metering algorithms used in practice is included in [98].

ALINEA (Asservissement Linéaire d’Entrée Autoroutière) is a well-known local ramp metering strategy that has been implemented in Paris and Amsterdam [105]. The control law is a simple proportional feedback law:

\[ r(k) = r(k - 1) + K_R(\hat{o} - o_{out}(k)) \]  

(5.7)

where \( K_R \) is a tuning parameter, \( o_{out}(k) \) is the occupancy of the freeway immediately downstream of the ramp, and \( \hat{o} \) is a desired occupancy of the freeway. The value of \( \hat{o} \) may be
changed by a higher level controller or user. A common suggestion for $\hat{o}$ is to use the value of critical occupancy $o_{cr}$ which corresponds to the maximum flow of the freeway. This $o_{cr}$ is the same as $n^c$ defined in the ACTM model in Section 5.1. A suggested value for $K_R$ is 70vph. Results show that this value worked well across multiple highway ramps [105]. Moreover, ALINEA worked better than feedforward strategies which do not utilize feedback and ALINEA performed no worse than other simple feedback strategies.

5.4 Model Predictive Control

Model predictive control is a widely favored strategy in the transportation research community due to its ability to directly optimize an objective function as well as directly incorporate demand predictions and constraints. Both system models of ACTM and METANET have been widely incorporated into model predictive control schemes [106, 3, 107, 108]. In most model predictive controllers for ramp metering, a metric for freeway efficiency is used as the cost function. We define some of these metrics next.

The first commonly used metric is the Total Travel Time (TTT). This metric encapsulates the total travel time of all the vehicles on the network. This metric is equivalent to the sum over densities in all links and queues at all time steps:

$$J_{TTT} = \sum_{i=1}^{I} \sum_{k=0}^{\infty} n_i(k) + l_i(k).$$  \hfill (5.8)

The Total Travel Time metric should be minimized in an MPC problem.

A second commonly used metric is the Total Travel Distance (TTD). The total travel distance is a measurement of throughput and is equal to the sum of the segment flows and on-ramp flows over all time:

$$J_{TTD} = \sum_{i=1}^{I} \sum_{k=0}^{\infty} f_i(k) + r_i(k).$$  \hfill (5.9)

A large total travel distance is desired.

Next, we formulate the linear program that is typically used in an optimal control framework [3]. The objective for ramp metering is a weighted sum of the Total Travel Time and Total Travel Distance metrics. This choice of cost $J_{LP}$ is:

$$J_{LP} = \sum_{i=1}^{I} \sum_{k=0}^{H} (n_i(k) + l_i(k) - \eta (f_i(k) + r_i(k))) .$$  \hfill (5.10)

where $H$ is the prediction horizon. Minimizing $J_{LP}$ is analogous to minimizing total time spent in network and maximizing throughput where $\eta$ in Equation (5.10) is a tuning parameter used to determine the weight assigned to throughput in the optimization problem.
Minimizing $J_{LP}$ subject to the dynamics defined by (5.3),(5.2),(5.4),(5.5) leads to a non-convex optimization problem since Equations (5.4) and (5.5) are non-linear. It is shown in [3] that in lieu of this non-convex optimization problem, a linear program with relaxed flow equations works well with the relaxation being exact under certain conditions. The main idea of relaxing the flow equations is to replace equation (5.4) with the following inequalities:

\[
\begin{align*}
    f_i(k) &\leq \bar{\beta}_i v_i (n_i(k) + \gamma r_i(k)) \\
    f_i(k) &\leq w_{i+1} (\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)) \\
    f_i(k) &\leq \bar{f}_i.
\end{align*}
\]

The equation for the last link (5.5) is similarly relaxed.

Then, the following linear program (LP) is solved in a receding horizon fashion:

\[
\begin{align*}
    \min \quad & \sum_{i=1}^{I} \sum_{k=1}^{H} (n_i(k) + l_i(k) - \eta (f_i(k) + r_i(k))) \\
    \text{s.t.} \quad & n_i(k+1) = n_i(k) + f_{i-1}(k) + r_i(k) - \frac{f_i(k)}{\beta_i(k)} & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & l_i(k+1) = l_i(k) + d_i(k) - r_i(k) & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & f_i(k) \leq \bar{\beta}_i v_i (n_i(k) + \gamma r_i(k)) & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & f_i(k) \leq w_{i+1} (\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)) & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & f_i(k) \leq \bar{f}_i & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & r_i(k) \leq \bar{r}_i & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & r_i(k) \geq 0 & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & n_i(k) \geq 0 & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & l_i(k) \geq 0 & \forall i = \{1, \ldots, I\}, k = \{1, \ldots, H\} \\
    & n_i(0) = n_i|t \\
    & l_i(0) = n_i|t 
\end{align*}
\]

In this thesis, this problem will be referred to as the relaxed LP formulation. A model predictive control solution solves the problem (5.12) at each time step $k$ with current state feedback $n_i(0)$, $l_i(0)$ for all segments $i$ in the network. The first ramp metering input $r_i(0)$ is applied at all ramp meters $i \in \{1, \ldots, I\}$ and the problem is resolved at the following time step $k + 1$ with new feedback information. The problem (5.12) can be quite large if the network has a large number of segments, $n_s$, and/or if a large prediction horizon $H$ is utilized. We note that persistent feasibility and stability is not proven to be guaranteed for this control design.
As discussed in this chapter, ramp metering systems are large-scale networked distributed systems. As a result, some distributed controllers have been proposed in the literature, including a game theoretic controller [109] and a robust decentralized controller [110]. A few distributed model predictive controllers have also been proposed. A distributed ramp metering and variable speed limit MPC is proposed in [111] where only one communication cycle is required. However, its performance is suboptimal compared to a centralized controller, and the results are verified for a network of containing at most three links. An asynchronous ADMM algorithm is utilized in [112] splitting the computation where the optimization problem solved at every time step is a linear program. This algorithm appears to be considerably slow, requiring 10 minutes to compute a solution in parallel and allowing control updates only every 17 minutes.

In this thesis, we propose a stable hybrid model predictive controller for ramp metering. This differs from the relaxed LP formulation described in Section 5.4 because this method guarantees persistent feasibility and stability of the MPC problem. This results in a mixed integer quadratic program. We propose a smoothing method in order approximate the problem as a smooth nonlinear problem. This problem formulation is non-convex. We develop and apply a novel distributed inexact interior point method to the problem which is guaranteed to converge to a KKT point for smooth nonlinear problems under mild assumptions.
Chapter 6

Stable Hybrid Model Predictive Control for Ramp Metering

Model predictive control is heavily favored as a control solution for ramp metering systems. In [113], a nonlinear model predictive controller minimizing a weighted sum of Total Travel Time (TTT) and Total Time Spent (TTS) was introduced. In [108], variable speed limits are taken into account in a nonlinear model predictive controller with a TTS cost function. In [114], ramp metering flows are assumed to be obtained through an existing strategy, and the speed limits are attained through a model predictive controller. In [3], a weighted sum of TTT and Total Travel Distance (TTD) is optimized at every time step, where the optimization problem of interest is formulated as a linear program. In [115], speed limits are considered in addition to ramp metering while optimizing for the same performance measure as [3].

Despite the popularity and practicality of model predictive control, stability and feasibility properties are often not addressed in practice. The existing work in the traffic network control literature for handling infeasibility conditions include [116] which proposes to minimize the amount of constraint violation of temporal logic specifications for signalized intersections and [117], where persistent feasibility for a target set of states is guaranteed by constructing invariant sets of a finite abstraction of the system (an approximation of the system evolution). Both [116] and [117] are for urban arterial control; consequently, for freeways, the issues of persistent feasibility and stability under model predictive control are still not addressed. Hence, the focus of this work is to design model predictive controllers for traffic networks with guarantees on persistent feasibility and closed-loop stability of the controller.

The organization of this chapter is as follows. First, we show how to describe the Asymmetric Cell Transmission Model (ACTM) as a piecewise affine system over regions of the state and input. Next, we show how to develop a hybrid model predictive controller for this system. The computation of an equilibrium point is analyzed in detail. To stabilize the system, we show how to design a terminal set and terminal cost for the system. Then, we compare our proposed strategy to the popular linear programming approach which min-
imizes the total travel time subject to a relaxed version of ACTM. Finally, we conclude the chapter and discuss some ideas subject to future work.

6.1 Piecewise Affine Model

In this section, we show how the Asymmetric Cell Transmission Model can be written as a piecewise affine system model. The minimum operator in Equation (5.4) implies that freeway can be described by a piecewise affine model: in free flow, \( f_i(k) = \beta_i v_i(n_i(k) + \gamma r_i(k)) \); otherwise in congestion, \( f_i(k) = w_{i+1} (\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)) \). This piecewise affine system is described by the Equations (5.3), (5.2) and (5.4). Next, we specify the state space partitions affiliated with each piece of the system. The dynamics of segment \( i \) are dependent on segment \( i \)'s current density \( n_i(k) \) and the density of the segment directly downstream \( n_{i+1}(k) \). Define the uncongested flow \( f_{v,i}(k) \) of segment \( i \) and the congested flow \( f_{w,i}(k) \):

\[
\begin{align*}
    f_{v,i}(k) &= \beta_i v_i(n_i(k) + \gamma r_i(k)) \\
    f_{w,i}(k) &= \begin{cases} 
        w_{i+1}(\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)) & \text{if } i < I \\
        \bar{f}_i & \text{if } i = I 
    \end{cases}
\end{align*}
\]  

The last segment, when \( i = I \), is different from other segments as there is no downstream segment imposing restrictions to its flow. Hence, the flow \( f_i(k) \) can be characterized by three regions:

\[
    f_i(k) = \begin{cases} 
        f_{v,i}(k) & \text{if } f_{v,i}(k) \leq f_{w,i}(k), f_{w,i}(k) \leq \bar{f}_i \\
        f_{w,i}(k) & \text{if } f_{w,i}(k) \leq f_{v,i}(k), f_{w,i}(k) \leq \bar{f}_i \\
        \bar{f}_i & \text{if } \bar{f}_i \leq f_{v,i}(k), \bar{f}_i \leq f_{w,i}(k).
    \end{cases}
\]  

In general, the regions for flow \( f_i(k) \) defined in (6.3) depend on the state \( n_i(k) \) and \( n_{i+1}(k) \) (if \( i < I \)) as well as the input \( r_i(k) \). If \( \gamma = 0 \), then the regions defining \( f_i(k) \) are not dependent on the input \( r_i(k) \).

**Remark 10.** By definition (6.3), the flow \( f_i(k) \) is continuous at its region boundaries. This is because \( f_{v,i}(k) = f_{w,i}(k) \) at the boundaries, which is coincidentally the function value of both regions.

The dynamics of \( n_i(k+1) \) rely on both \( f_{i-1}(k) \) and \( f_i(k) \) resulting in 6 equations for the density of segment \( i \). The exception to this is for the first segment \( i = 1 \) and the last segment \( i = I \). The first segment includes \( f_0(k) \) as an external demand, so there are only three equations for \( n_1(k+1) \). The density of the last segment is simplified to six equations since \( f_I(k) \) is only defined by two regions (5.5).

These equations hold for a network of any size. The flow variables \( f_i(k) \) can be eliminated by substituting the Eq. (6.3) into (5.3) to achieve a piecewise controller over state space regions \( \mathcal{R}_i \). The dynamics are then defined by equation (5.2) and

\[
    N(k+1) = A_i N(k) + B_i R(k) + g_i \text{ if } (N(k), R(k)) \in \mathcal{R}_i
\]  

(6.4)
where $N(k) = [n_1(k), n_2(k), \ldots, n_I(k)]^T$ and $R(k) = [r_1(k), r_2(k), \ldots, r_I(k)]^T$. Denoting the vector $L(k) = [l_1(k), \ldots, l_I(k)]^T$, $D(k) = [d_1(k), \ldots, d_I(k)]^T$ the vector form of equation (5.2) is

$$L(k+1) = L(k) + D(k) - R(k). \quad (6.5)$$

Note that in this notation, we have assumed that all segments have onramps, and all onramps are actuated. If this is not the case, the equations can easily be modified so that the ramp queues of actual onramps appear in the equations.

For example, the piecewise affine equations for a two segment freeway are defined by six pieces:

$$A_1 = \begin{bmatrix} 1 - v_1 & 0 \\ \bar{\beta}_1 v_1 & 1 - v_2 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 1 - \gamma v_1 & 0 \\ \gamma \bar{\beta}_1 v_1 & 1 - \gamma v_2 \end{bmatrix}, \quad g_1 = \begin{bmatrix} f_0(k) \\ 0 \end{bmatrix} \quad (6.6a)$$

$$A_2 = \begin{bmatrix} 1 & \frac{1}{\beta_1} w_2 \\ 0 & 1 - v_2 - w_2 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1 & \frac{1}{\bar{\beta}_1} \gamma w_2 \\ 0 & 1 - \gamma v_2 - \gamma w_2 \end{bmatrix},$$

$$g_2 = \begin{bmatrix} f_0(k) - \frac{1}{\beta_1} w_2 \bar{n}_2 \\ w_2 \bar{n}_2 \end{bmatrix} \quad (6.6b)$$

$$A_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 - v_2 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 - \gamma v_2 - \gamma w_2 \end{bmatrix},$$

$$g_3 = \begin{bmatrix} f_0(k) - \frac{1}{\bar{\beta}_1} \bar{f}_1 \end{bmatrix} \quad (6.6c)$$

$$A_4 = \begin{bmatrix} 1 - v_1 & 0 \\ \bar{\beta}_1 v_1 & 1 \end{bmatrix}, \quad B_4 = \begin{bmatrix} 1 - \gamma v_1 & 0 \\ \gamma \bar{\beta}_1 v_1 & 1 \end{bmatrix}, \quad g_4 = \begin{bmatrix} f_0(k) \\ -\frac{\bar{f}_1}{\bar{\beta}_1} \end{bmatrix} \quad (6.6d)$$

$$A_5 = \begin{bmatrix} 1 & \frac{w_2}{\beta_1} \\ 0 & 1 - w_2 \end{bmatrix}, \quad B_5 = \begin{bmatrix} 1 & \frac{\gamma w_2}{\bar{\beta}_1} \\ 0 & 1 - \gamma w_2 \end{bmatrix},$$

$$g_5 = \begin{bmatrix} f_0(k) - \frac{w_2 \bar{n}_2}{\bar{\beta}_1} \\ w_2 \bar{n}_2 - \frac{\bar{f}_1}{\bar{\beta}_1} \end{bmatrix} \quad (6.6e)$$

$$A_6 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B_6 = \begin{bmatrix} 1 & 0 \\ 0 & 1 - \gamma w_2 \end{bmatrix}, \quad g_6 = \begin{bmatrix} f_0(k) - \frac{\bar{f}_1}{\bar{\beta}_1} \\ \bar{f}_1 - \frac{\bar{f}_2}{\bar{\beta}_2} \end{bmatrix} \quad (6.6f)$$

with regions $R_i$:

$$R_1 \equiv \begin{bmatrix} \bar{\beta}_1 v_1 & w_2 & \bar{\beta}_1 v_1 \gamma & w_2 \gamma \\ \bar{\beta}_1 v_1 & 0 & \bar{\beta}_1 v_1 \gamma & 0 \gamma \\ 0 & \bar{\beta}_2 v_2 & 0 & \bar{\beta}_2 v_2 \gamma \end{bmatrix} \begin{bmatrix} n_1(k) \\ n_2(k) \\ r_1(k) \\ r_2(k) \end{bmatrix} \leq \begin{bmatrix} w_2 \bar{n}_2 \\ \bar{f}_1 \\ \bar{f}_2 \end{bmatrix} \quad (6.7a)$$

$$R_2 \equiv \begin{bmatrix} -\bar{\beta}_1 v_1 & -w_2 & -\bar{\beta}_1 v_1 \gamma & -w_2 \gamma \\ 0 & -w_2 & 0 & -w_2 \gamma \\ 0 & \bar{\beta}_2 v_2 & 0 & \bar{\beta}_2 v_2 \gamma \end{bmatrix} \begin{bmatrix} n_1(k) \\ n_2(k) \\ r_1(k) \\ r_2(k) \end{bmatrix} \leq \begin{bmatrix} -w_2 \bar{n}_2 \\ \bar{f}_1 - w_2 \bar{n}_2 \\ \bar{f}_2 \end{bmatrix} \quad (6.7b)$$
CHAPTER 6. STABLE HYBRID MODEL PREDICTIVE CONTROL FOR RAMP METERING

The shifted horizon problem is then solved at the next time step using the current state feedback. This is solved at each time step. The first input is applied, then the horizon is receded and the equilibrium point. Then, we can apply hybrid model predictive control design techniques to synthesize the terminal set and terminal cost to guarantee persistent feasibility and stability of the piecewise affine system.

In this section, we construct a hybrid model predictive controller for the ramp metering system. The goal of ramp metering is to make the freeway system as efficient as possible.

Our approach is to design a model predictive controller whose objective is to track the equilibrium point. Then, we can apply hybrid model predictive control design techniques to synthesize the terminal set and terminal cost to guarantee persistent feasibility and stability of the piecewise affine system.

For hybrid model predictive control, a constrained finite time optimal control problem is solved at each time step. The first input is applied, then the horizon is receded and the shifted horizon problem is then solved at the next time step using the current state feedback. The $H$-step constrained finite time optimal control problem at time step $t$ is:

\[
\min_{\tilde{X}(k), \tilde{U}(k)} \tilde{X}(H)^T P \tilde{X}(H) + \sum_{k=0}^{H-1} \tilde{X}(k)^T Q \tilde{X}(k) + \tilde{U}(k)^T R \tilde{U}(k) \tag{6.8a}
\]

s.t. \quad (6.4), (6.5) \tag{6.8b}
\[
0 \leq N(k) \leq \bar{N} \tag{6.8c}
\]
\[
0 \leq L(k) \leq \bar{L} \tag{6.8d}
\]
\[
0 \leq R(k) \leq \bar{R} \tag{6.8e}
\]
\[
\tilde{X}(H) \in \mathcal{X}_f \tag{6.8f}
\]

where the regions $\mathcal{R}_i$ are dependent on both state and input.

6.2 Hybrid Model Predictive Control

In this section, we construct a hybrid model predictive controller for the ramp metering system. The goal of ramp metering is to make the freeway system as efficient as possible.

Our approach is to design a model predictive controller whose objective is to track the equilibrium point. Then, we can apply hybrid model predictive control design techniques to synthesize the terminal set and terminal cost to guarantee persistent feasibility and stability of the piecewise affine system.

For hybrid model predictive control, a constrained finite time optimal control problem is solved at each time step. The first input is applied, then the horizon is receded and the shifted horizon problem is then solved at the next time step using the current state feedback. The $H$-step constrained finite time optimal control problem at time step $t$ is:

\[
\min_{X(k), U(k)} \tilde{X}(H)^T P \tilde{X}(H) + \sum_{k=0}^{H-1} \tilde{X}(k)^T Q \tilde{X}(k) + \tilde{U}(k)^T R \tilde{U}(k) \tag{6.8a}
\]

s.t. \quad (6.4), (6.5) \tag{6.8b}
\[
0 \leq N(k) \leq \bar{N} \tag{6.8c}
\]
\[
0 \leq L(k) \leq \bar{L} \tag{6.8d}
\]
\[
0 \leq R(k) \leq \bar{R} \tag{6.8e}
\]
\[
\tilde{X}(H) \in \mathcal{X}_f \tag{6.8f}
\]
CHAPTER 6. STABLE HYBRID MODEL PREDICTIVE CONTROL FOR RAMP METERING

\[ N(0) = N|_t, \quad L(0) = L|_t \]  

(6.8g)

where \( \bar{X}(k) = [N(k)^T - N_{ss}^T, L(k)^T - L_{ss}^T]^T, \) \( \bar{U}(k) = R(k) - R_{ss}, \) \( \bar{N} = [\bar{n}_1, \ldots, \bar{n}_I]^T, \) \( \bar{L} = [\bar{l}_1, \ldots, \bar{l}_I]^T, \) \( \bar{R} = [\bar{r}_1, \ldots, \bar{r}_I]^T. \) The steady state or equilibrium points are denoted as \( N_{ss}, L_{ss}, R_{ss}. \) The cost is quadratic with state penalty \( Q = Q^T > 0, \) input penalty \( R = R^T > 0, \) and terminal cost penalty \( P = P^T > 0. \) The final state is restricted to be in the terminal set \( \mathcal{X}_f. \) The current state feedback at time \( t, N|_t \) and \( L|_t \) are used. The MPC control law for ramp metering at time \( t \) is then:

\[ R(t) = \bar{U}^*(0) + R_{ss} \]  

(6.9)

where \( \bar{U}^*(0) \) is the optimal input for the first step computed by (6.8).

We next show how to compute the equilibrium of the system for a constant demand profile. Then, we design a stable and persistently feasible Hybrid MPC by applying the design methodology proposed in [118]-[119] to our problem.

Equilibrium Computation

The equilibrium of the piecewise system (6.4) is dependent on the current demand. In order to evaluate an equilibrium of the system, we assume the demand is stationary or constant over time, i.e. \( d_i(k) = d_i \forall k \) and \( f_0(k) = f_0 \forall k. \) Denote the constant demand vector \( D = [d_1, \ldots, d_I]^T. \)

First, without knowing which region \( \mathcal{R}_i \) of the model (6.4) the equilibrium point lies in, the steady state queue length \( L_{ss} \) and ramp metering input \( R_{ss} \) can be determined by computing the steady state of (6.5):

\[ L_{ss} = L_{ss} + D - R_{ss}. \]  

(6.10)

This equations tells us that the steady state ramp metering input \( R_{ss} = D \) and that \( L_{ss} \) is a constant value.

Remark 11. A steady state ramp metering input \( R_{ss} = D \) is feasible if \( 0 \leq D \leq \bar{R} \) where \( \bar{R} \) is the vector of upper bound limits for \( R, \) i.e. \( \bar{R} = [\bar{r}_1, \ldots, \bar{r}_I]^T. \)

It can be verified by examining equation (6.5) that if \( D \) exceeds the feasible values for \( R, L \) does not have an equilibrium point. That is, if \( d_i > \bar{r}_i, \) then queue \( i \) is receiving more cars than it can release onto the freeway. If the demand is stationary, then the queue length \( l_i \) will grow without bound and is therefore unstable.

Remark 12. A steady state queue length vector \( L_{ss} \) is feasible if \( 0 \leq L_{ss} \leq \bar{L} \) where \( \bar{L} \) is the vector of upper bound limits for \( L, \) i.e. \( \bar{L} = [\bar{l}_1, \ldots, \bar{l}_I]^T. \) The steady state queue length for segment \( i, [l_i]_{ss} = l_i(0) \) if \( d_i = \bar{r}_i. \) Otherwise, the steady states \( [l_i]_{ss} \in [0, \bar{l}_i] \) are reachable.
CHAPTER 6. STABLE HYBRID MODEL PREDICTIVE CONTROL FOR RAMP METERING

Remark 13. Among a continuum of equilibria for $L$, the desired equilibrium point is $L_{ss} = 0$. This corresponds to empty queues at steady state, which guarantees that no cars are getting stuck in a queue (cars are stuck if $L > 0$ and $R = 0$ for a long period of time). However, in order to prove stability of the hybrid MPC (see the details in the next section), the equilibrium point must be interior to the constraint set. Therefore, we choose an equilibrium point of $L_{ss} = \epsilon$, where $\epsilon$ is a positive number close to zero. In practice, a controller that brings the queue length to $\epsilon$ effectively brings $L$ to zero.

Next, we review some results from the literature regarding the steady state density vector $N_{ss}$. In [94], the equilibria of the Cell Transmission Model (CTM) under stationary demand is studied. The CTM is similar to ACTM except that $\gamma = 0$ and there are no ramp queues $L(k)$. Equilibrium results are established under the assumption of feasible demand. We provide a definition of a feasible demand that is similar to that from [94] in Definition 1. Under the assumption of feasible demand, the work in [94] proves that there exists a unique uncongested equilibrium that is stable in the sense of Lyapunov. An uncongested equilibrium is an equilibrium point in which $N_{ss}$ is in the set $[0, N^c]$ where $N^c = [n^c_1, \ldots, n^c_I]^T$.

Definition 1 (cf. [94]). A constant demand $D = [d_1, \ldots, d_I]^T$ is feasible if there exists an equilibrium point of the system defined by (5.2), (5.3), (5.4), the resulting flows are feasible, i.e. $[f_i]_{ss} \in [0, \bar{f}_i] \forall i \in I$, and the ramp metering inputs are nonnegative, i.e. $[r_i]_{ss} \geq 0 \forall i \in I$.

The following lemma and proof are similar to Lemma 4.1 of [94] and its corresponding proof. We include Lemma 1 and proof here for the more general ACTM model ($\gamma$ is not restricted to be zero) for completeness.

Lemma 1. For a stationary demand $D$ that is feasible in the sense of Definition 1, there exists an uncongested equilibrium of the ACTM model.

Proof. Recall that from Remark 11, we know that the equilibrium point of $[r_i]_{ss} = d_i$. If there is an equilibrium point of $n_i$ in equation (5.3), then the following equations compute the equilibrium flow $f_i$:

$$[f_i]_{ss} = \tilde{\beta}_i (\bar{f}_{i-1} + [r_i]_{ss}).$$

We drop the steady state subscript $[\cdot]_{ss}$ for the remainder of the proof.

Suppose the equilibrium flow is uncongested, i.e. $f_i = f_{v,i}$ for all $i \in I$. Thus, we can find an $n_i$ by solving:

$$f_i = f_{v,i} = \tilde{\beta}_i v_i (n_i + \gamma r_i).$$

This gives:

$$n_i = (\tilde{\beta}_i v_i)^{-1} f_i - \gamma r_i.$$
We propose that (6.12) is the uncongested equilibrium of \( n_i \) with \( f_i \) defined by (6.11). The proposed \( n_i \) is an equilibrium point of (5.3) by construction.

Next, we must show that our proposed equilibrium \( n_i \in [0, n^c_i] \forall i \in \mathcal{I} \). Notice that:

\[
n_i \leq (\bar{\beta}_i v_i)^{-1} \bar{f}_i - \gamma r_i
\]

\[
\leq n^c_i - \gamma r_i
\]

\[
\leq n^c_i.
\]

Thus, all densities \( n_i \leq n^c_i \) and are therefore considered uncongested. We also need to prove that such an \( n_i \) is not negative. We prove by contradiction. Suppose \( n_i < 0 \). Then,

\[
n_i = (\bar{\beta}_i v_i)^{-1} f_i - \gamma r_i < 0
\]

The equilibrium computation (6.11) ascertains that

\[
n_i = (\bar{\beta}_i v_i)^{-1} \bar{\beta}_i (f_{i-1} + r_i) - \gamma r_i < 0
\]

which implies:

\[
f_{i-1} < r_i (\gamma v_i - 1).
\]

However, since \( v_i \in [0,1] \), \( \gamma \in [0,1] \) by definition and \( r_i \geq 0 \) by assumption, the last line implies that \( f_{i-1} < 0 \) which is clearly a contradiction as it indicates that the vehicles must move in the opposite direction of traffic. Hence, \( n_i \geq 0 \).

For the general freeway, let \( R_1 \) be the uncongested region, i.e.

\[
R_1 = \{n_i(k), r_i(k) | f_{v,i}(k) \leq f_{w,i}, f_{v,i}(k) \leq \bar{f}_i \forall i \in \mathcal{I}\}. \tag{6.13}
\]

Note that this corresponds to the example equations in (6.6). We next show that the uncongested equilibrium \( N_{ss} \) necessarily lies in \( R_1 \). That is, an uncongested equilibrium implies that all flows are in free flow, or \( f_i(k) = f_{v,i}(k) \) for all segments \( i \in \mathcal{I} \).

**Lemma 2.** If there is an uncongested equilibrium point \( N_{ss} \), then it is in \( R_1 \).

**Proof.** Recall that for our proposed uncongested equilibrium, \( n_i \leq n^c_i - \gamma r_i \) from the previous proof. Then,

\[
f_{v,i} = \bar{\beta}_i v_i (n_i + \gamma r_i)
\]

\[
\leq \bar{\beta}_i v_i (n^c_i - \gamma r_i + \gamma r_i)
\]

\[
\leq \bar{\beta}_i v_i n^c_i.
\]

Recall also that \( \bar{\beta}_i n^c_i = w_{i+1}(\bar{n}_{i+1} - n^c_{i+1}) \) from (5.6). Then,

\[
f_{v,i} \leq \bar{\beta}_i v_i n^c_i
\]
= w_{i+1}(\bar{n}_{i+1} - n_{c,i+1}) \\
\leq w_{i+1}(\bar{n}_{i+1} - n_i - \gamma r_i) \\
= f_{w,i}.

The second-to-last line is true because \( n_i \leq n_{c,i} - \gamma r_i \) \( \forall i \in I \), so \( n_{i+1} \geq n_i + 1 \). Thus, \( f_{v,i} \leq f_{w,i} \) \( \forall i \in I \). This is the definition of \( R_1 \), so the uncongested equilibrium must lie in \( R_1 \).

The piecewise affine model defined in Section ?? enables the development of a set definition assumption about the demand. The steady state density vector \( N_{ss} \) can be found by computing the equilibrium of the affine system defined over \( R_1 \), or by solving the following equation:

\[ N_{ss} = A_1 N_{ss} + B_1 R_{ss} + g_1. \]

(6.14)

In order for the steady state equation 6.14 to hold, the resulting solution \( N_{ss}, R_{ss} \) must lie in the uncongested region \( R_1 \). This brings us to the following theorem.

**Theorem 3.** Assume the following about the demand \( D \):

- \( 0 \leq D \leq \bar{R} \)
- The initial state \( L(0) \) is feasible
- Demand \( D \) and external flow \( f_0 \) satisfy \( ((I - A_1)^{-1}(B_1 D + g_1), D) \in R_1 \).

Then, there exists a feasible equilibrium \( N_{ss}, R_{ss}, L_{ss} \) of the system (6.4)-(6.5) such that \( N_{ss} \in [0, N^c], R_{ss} \in [0, \bar{R}], L_{ss} \in [0, \bar{L}] \). Moreover, there exists an equilibrium that is uncongested, i.e. an equilibrium lies in \( R_1 \).

**Proof.** The first two conditions give us feasible equilibrium vectors \( R_{ss} \) and \( L_{ss} \), following Remarks 11 and 12. The last condition gives a steady state \( N_{ss} \) that lies in \( R_1 \). Note that the matrix \( (I - A_1) \) is invertible with eigenvalues \( \{v_i\} \) (e.g., see (6.6a)).

Theorem 3 differs from the previous work [94] in that it precisely defines the set of demands that result in having an equilibrium point in the uncongested region, \( R_1 \). The particularly powerful result of Lemmas 1 and 2 is that a search through other regions \( R_i \) with \( i \neq 1 \) is not necessary. This is because an uncongested equilibrium must lie in \( R_1 \).

**Lemma 3.** If the assumptions of Theorem 3 hold, and the demands \( D \) and \( f_0 \) are such that \( 0 < D < \bar{R} \) and \( (I - A_1)^{-1}(B_1 D + g_1) > 0 \), then, the equilibrium \( N_{ss}, R_{ss}, L_{ss} \) of the system (6.4)-(6.5) is interior to the constraint set, i.e. \( N_{ss} \in (0, \bar{N}), R_{ss} \in (0, \bar{R}), L_{ss} \in (0, \bar{L}) \).

**Proof.** The equilibrium \( R_{ss} \in (0, \bar{R}) \) by Remark 11. The equilibrium \( L_{ss} \in (0, \bar{L}) \) by Remark 12. The equilibrium \( N_{ss} > 0 \) since \( N_{ss} = (I - A_1)^{-1}(B_1 D + g_1) > 0 \). The equilibrium \( N_{ss} < \bar{N} \) since by Lemma 1, the equilibrium point \( N_{ss} \leq N^c \) and by definition \( N^c < \bar{N} \).
CHAPTER 6. STABLE HYBRID MODEL PREDICTIVE CONTROL FOR RAMP METERING

Design of Terminal Set and Terminal Cost

It is well known that careful design of the terminal cost and terminal constraint of a receding horizon controller yields persistent feasibility and stability results (see Theorem 13.2 of [7]). In this section, we develop a stable and persistently feasible hybrid model predictive controller based on the methods of [118] and [119]. In particular, we follow the design of Algorithm 3.1 in [118]. We note that a similar algorithm is proposed in [120].

The design of a model predictive controller requires that the equilibrium point must lie in the interior of the constrained state space, i.e. $0 < N_{ss} < \bar{N}$, $0 < L_{ss} < \bar{L}$, $0 < R_{ss} < \bar{R}$ (see Assumption (A1) in Theorem 13.2 of [7]; also see [119]). We make the assumptions about demand in Lemma 3.

First, we identify the regions of the dynamics in which the equilibrium lies. In many cases, the equilibrium lies in the interior of region $R_1$, but it may generally lie on the border of multiple regions. Define the error dynamics of the regions $S_{eq}$ which contain the equilibrium point:

$$\tilde{X}(k+1) = \tilde{A}_i \tilde{X}(k) + \tilde{B}_i \tilde{U}(k) \quad i \in R_i, R_i \subseteq S_{eq}. \quad (6.15)$$

Note that the error dynamics are in general a piecewise linear system.

Then, a search is done to find stabilizing piecewise linear feedback controllers $\tilde{U}(k) = K_i \tilde{X}(k)$ for all regions in $S_{eq}$ and an associated Lyapunov function $V(\tilde{X}) = \tilde{X}^T P \tilde{X}$. This search can be done by solving the following semidefinite program:

$$\begin{align*}
\min & \quad \gamma \\
\text{s.t.} & \quad Z \succ 0 \quad (6.16b) \\
& \quad \begin{bmatrix}
Z & (\tilde{A}_i Z + \tilde{B}_i Y_i) & Q^{1/2} Z & R^{1/2} Z \\
(\tilde{A}_i Z + \tilde{B}_i Y_i)^T & Z & 0 & 0 \\
Q^{1/2} Z & 0 & \gamma I & 0 \\
R^{1/2} Z & 0 & 0 & \gamma I
\end{bmatrix} \succeq 0 \\
& \quad \forall i \in R_i \subseteq S_{eq} \quad (6.16c)
\end{align*}$$

where $Y_i = K_i Z$, $Z = P^{-1}/\gamma$

If the equilibrium lies in the interior of $R_1$, then the terminal cost $P$ can be designed by solving the Algebraic Riccati Equation for the $\tilde{A}_1$ and $\tilde{B}_1$ matrices defined in $R_1$ (cf. Remark 2 in [118]). Moreover, the terminal set can be found by computing the maximal positive invariant set for the system in $R_1$ subject to the LQR controller found for $\tilde{A}_1, \tilde{B}_1, Q, R$.

In general, the maximal positive invariant set for the system under the controllers $K_i$ found by (6.16) is used as the terminal set $\mathcal{X}_f$. In general, the maximal positive invariant set is intensive to compute as it is comprised of a union of polyhedron; see [119] for how to compute the maximal positive invariant set. We use tools from the multi-parametric toolbox [19] to compute the maximal positive invariant set.
By Theorem 3.1 of [118], the closed loop system is asymptotically stable under the hybrid model predictive controller (6.8)-(6.9) with the terminal cost $P$ determined by (6.16) and the terminal set the maximal positive invariant set of this system under piecewise linear feedback controllers $K_i$.

### Practical Considerations

We have designed a persistently feasible and stable controller. To the best of our knowledge, no other proposed controller for this problem has any feasibility or stability guarantees. There are however, a few practical considerations that must be taken into account.

The first consideration is how to determine the prediction horizon $N$ such that the set of feasible initial states covers the full state space. The initial feasible state can be computed for an $N$ step horizon by computing the controllable pre-sets of the piecewise affine system (6.4), (6.5). The desired prediction horizon $N$ would correspond to the $N$-step controllable pre-set that covers the full state space. However, these pre-sets are in general a union of polyhedrons and thus are hard to compute [119] since a union of polyhedrons is not necessarily convex and the union may consist of many polyhedrons. It may be easier in some cases to determine $N$ in simulation by checking how big $N$ must be to be feasible for the worst case initial feasible states.

The second consideration is that the assumptions about $D$ in Theorem (3) may be too restrictive. In reality, it is possible to allow the queues to spillover into the urban arterial networks. This effectively allows the upper bound constraint on $L$ to be a soft constraint. Simply adding a soft constraint extends the feasibility of the problem, but loses any guarantees about stability. In general, stability of soft constrained MPC is understood for linear systems [121], [122]. Unfortunately, designing stable MPC controllers for soft constrained piecewise affine systems has not yet been studied to the best of our knowledge.

### 6.3 Case Studies

In this section, we present simulation results comparing our hybrid MPC approach with the linear program (LP) formulation of optimal freeway ramp metering [3] for a two link freeway example with a sampling time of 3 seconds. The prediction horizon for both controllers is $H = 25$. Figure 6.1 demonstrates the trajectory of states and inputs when our hybrid MPC framework is utilized for ramp metering whereas Figure 6.2 shows the same plots where the linear programming formulation is used.

We evaluate the efficiency of the two controllers using two metrics: total travel time (TTT) and total travel distance (TTD). These metrics were defined in Section 5.4. Recall that it is desirable to minimize the total travel time of all vehicles and maximize the total travel distance.

As seen in figures 6.1 and 6.2, the initial condition is in congestion. Both controllers can successfully steer the freeway to the uncongested regime and an uncongested equilibrium is
achieved. The total travel time (TTT) of the hybrid trajectory over the simulation period is 648.3 vehicles while the total travel time of the trajectory resulting from linear program formulation is 638.3 vehicles. Thus, the linear program formulation is 1.55\% more efficient in terms of total travel time. However, the total travel distance (TTD) is the same for both controllers at 353.1 vehicles.

The trajectories of the two controllers are quite different qualitatively. Figures 6.1 and 6.2 show that the hybrid MPC trajectory is much smoother than the LP trajectory, and the variations of the control input are much less for the hybrid controller. This smoothness is advantageous in a real-time implementation since rapidly changing ramp flows cannot be achieved in practice; this is because traffic response time is longer than the agility required
We note that our proposed controller has guarantees by design whereas the linear programming method does not. Our controller has a terminal set and terminal cost design which guarantee the stability and persistent feasibility of the controller. For the linear program, a cool-down period is used to encourage stability. A cool-down period is a period at the end of the prediction horizon in which demands are assumed to be zero. The idea of the cool-down period is to imitate the role of a terminal cost; it penalizes vehicles that remain on the freeway at the end of the horizon. The cool-down period should be long enough such that all remaining vehicles leave the freeway. The duration of this cool-down period is decided heuristically, and there is no formal way of computing when this cool-down period should begin or how long it should last. Additionally, there are no guarantees of persistent feasibility attached to the linear programming controller.

In addition, in order for the linear program formulation to work properly, the length of the prediction horizon must be fairly large. The prediction horizon of the linear program is essentially designed through trial and error observations. The controller must be tested in various scenarios to check that the controller eventually moves toward an equilibrium point. Our prediction horizon is determined by reachability analysis and is guaranteed to reach the terminal set within the prediction horizon length. Moreover, our controller is guaranteed to reach the equilibrium point whereas the convergence of the linear program must be checked for various prediction horizons and various initial conditions.

Another issue with the linear programming approach is the exactness of the model relaxation. Even if the horizon length and cool down period length are properly designed, there are scenarios where the relaxation of the ACTM model does not match the nonlinear model exactly. For example, if the ramp flows $R$ are not strictly less than the available mainline space allocated to onramps, the relaxations introduced in [3] do not necessarily hold. This results in model mismatch between the predicted model of the controller and the evolution of the actual network. Hence, in such cases, the linear program does not perform optimally, while the hybrid formulation is not restricted to certain conditions on the mainline densities.

We show an example where the relaxed LP controller is short-sighted in Figure 6.3. The controller is not able to steer the states to the uncongested equilibrium. In addition, there are steps in which the ACTM relaxation is not exact; we propagate the relaxation solution through the ACTM model. On the other hand, Figure 6.4 shows the hybrid MPC controller is smooth and is able to converge to the desired stable uncongested equilibrium point with the same short prediction horizon. For the relaxed LP controller, the TTT is 436 vehicles and the TTD is 130.8 vehicles. For the hybrid controller, the TTT is 219.4 vehicles and the TTD is 167 vehicles.

### 6.4 Conclusions and Future Work

In conclusion, we have designed the first stable model predictive controller for the ramp metering problem. To the best of our knowledge, no other ramp metering model predictive
controller has rigorous guarantees of stability and persistent feasibility. We compare our controller to the widely proposed linear programming solution and show that there is no significant performance degradation of our design. We show that our controller provides a smooth trajectory and converges quite quickly to the equilibrium point.

We acknowledge that in general the demand may vary throughout the course of a day. However, real data shows that the demand is constant for long periods of the day. In ramp metering research, the demand is often assumed to be constant over a prediction horizon. Nevertheless, the handling of time varying demand is of interest. It is common practice in the model predictive control literature to utilize an observer and constant demand model. This design is able to achieve desired steady state convergence (see Theorem 13.4 in [7]).

Finally, the scalability of a hybrid model predictive control approach is subject to future
work. However, the control design done in this work makes large-scale methodology such as decentralized, hierarchical, and distributed model predictive control more easily applicable. In particular, it is now possible to design and analyze stability of large-scale algorithms for the ramp metering problem.
Chapter 7

An Inexact Interior Point Algorithm for DMPC with Application to Ramp Metering

Model Predictive Control requires the solution of a potentially nonlinear, non-convex constrained finite time optimal control problem at every time step. Interior point algorithms are able to solve nonlinear problems and have been shown to be effective for many MPC applications ranging from small to large control problems [123, 124, 125, 126, 45]. The purpose of this chapter is to explore methods that promote the scalability of interior point methods, particularly for nonlinear MPC.

For large-scale nonlinear MPC problems, the most popular centralized approaches are interior point or sequential quadratic programming methods. Interior point method is highly favored due to its fast performance on a centralized platform [127, 45]. However, for large-scale control problems, the computation can become prohibitive in real-time on a centralized computational platform.

Distributed MPC solutions are proposed for the solution of large-scale model predictive control problems. In particular, two popular solutions are dual decomposition and ADMM, as discussed in the introduction of this thesis. However, dual decomposition and ADMM are not guaranteed to converge for general non-convex problems. In some cases, ADMM may converge for non-convex problems if the nonlinearities are polynomial functions [128]. However, there are many nonlinearities in MPC problems that are not polynomial functions.

The main contribution of this chapter is to propose a distributed interior point method for general nonlinear problems. In particular, the main linear algebra in the interior point method is parallelized by use of iterative linear algebra methods. This is attractive for speeding up the linear algebra since the main computational bottleneck of the interior point method is the solving of one or two large matrix solve operations at every iteration of the interior point algorithm. This matrix operation is used to solve a linear system of Karush-Kuhn-Tucker conditions to find the next step direction (a second matrix solve operation may be used to provide a corrector step, as in the Mehrotra corrector-predictor algorithm).
In centralized solutions, a direct factorization of an indefinite matrix is done [127], such as the $LDL^T$ factorization. However, iterative methods can quickly solve sparse linear systems approximately [92],[129] and are trivially parallelizable. In particular, Krylov Subspace methods have good convergence properties, with the conjugate gradient method being the most well-known Krylov subspace method.

In this chapter, we propose a distributed interior point method which utilizes a Krylov subspace method, MINRES or SYMMLQ [130], in combination with an inexact interior point method [131]. This inexact interior point method is designed to allow inexact linear algebra computations. In addition, the inexact interior point method has global convergence guarantees and exhibits a local superlinear convergence rate under mild assumptions. This contrasts with other distributed methods such as dual decomposition and ADMM which have not been proven to converge for general non-convex problems.

One challenge in implementing a distributed interior point method is that the iterative numerical linear algebra solver needs to converge quickly. We use preconditioners to improve the condition number of the KKT matrix and thus speed up the performance of MINRES or SYMMLQ. In particular, we propose the use of preconditioners presented by [132] that are particularly tailored for optimal control problems. Note that a similar idea of solving an inexact interior point method in a distributed manner is proposed in [35]. This work differs from our chapter since it uses a different inexact interior point algorithm and does not use preconditioners that are tailored for optimal control.

Finally, we study the application of the inexact interior point method to the ramp metering problem. The ramp metering problem is a large-scale control problem, ranging from 1,000s to 100,000s of variables in a problem at a single time step [3]. A stable hybrid model predictive controller was proposed in Chapter 6 for control of ramp metering systems. Such a control law requires solving a mixed integer quadratic program at every time step. We use a conjugate smoothing method [133, 134] to smooth the problem. In previous work, ADMM was proposed for the distributed solution of ramp metering problems [112]. Unfortunately this does not apply to the smoothed model predictive controller due to non-convexity. Therefore, we use the inexact interior point method developed in this chapter. In the results, we show that the inexact interior point method is able to compute the solution of this smoothed ramp metering problem. Moreover, the trajectories provided by the smoothed hybrid model predictive controller are nearly identical to the hybrid model predictive control trajectories.

### 7.1 Nonlinear Model Predictive Control

A model predictive controller solves a constrained finite time optimal control (CFTOC) problem at each time step in a receding horizon fashion. Consider the following CFTOC problem at time step $t$:

\[
\min_{X,U} \ p(x(N)) + \sum_{k=0}^{N-1} q(x(k), u(k))
\]  

(7.1a)
CHAPTER 7. AN INEXACT INTERIOR POINT ALGORITHM FOR DMPC WITH APPLICATION TO RAMP METERING

\[
\begin{align*}
\text{s.t. } x(k + 1) &= f(x(k), u(k), \hat{d}(k)) \quad (7.1b) \\
x(k) &\in \mathcal{X}, \ k = 1, \ldots, N \quad (7.1c) \\
u(k) &\in \mathcal{U}, \ k = 0, \ldots, N - 1 \quad (7.1d) \\
x(N) &\in \mathcal{X}_f \quad (7.1e) \\
x(0) &= x|_t \quad (7.1f)
\end{align*}
\]

where the state at time step \(k\) is \(x(k) \in \mathbb{R}^n\) with \(X = [x(0)^T, \ldots, x(N)^T]^T\), the input at time step \(k\) is \(u(k) \in \mathbb{R}^m\) with \(U = [u(0)^T, \ldots, u(N - 1)^T]^T\), \(\hat{d}(k)\) is the predicted load at time step \(k\), the prediction horizon length is \(N\), and the current state feedback \(x|_t\) is used. The dynamics are encoded by \(f(\cdot)\) and are generally nonlinear. We assume that \(f(\cdot)\) is twice differentiable with its first and second derivatives being continuous.

The constraints are \(\mathcal{X} = \{x \in \mathbb{R}^n | H_x x \leq h_x\} a\) polyhedron encoding state constraints, and \(\mathcal{U} = \{u \in \mathbb{R}^m | H_u u \leq h_u\}, a\) polyhedron encoding actuator constraints. The objective of the control problem is encoded in the cost function as a summation of stage costs \(q(x(k), u(k))\) and a terminal cost \(p(x(N))\). We assume that the cost functions are also twice differentiable with first and second derivatives being continuous.

An optimization algorithm such as the interior point method searches for the solution to the Karush-Kuhn-Tucker optimality conditions. In order to write these conditions concisely, we lump the equality constraints together:

\[
g(X, U) = \begin{bmatrix} 
    x(1) - f(x(0), u(0), \hat{d}(0)) \\
    x(2) - f(x(1), u(1), \hat{d}(1)) \\
    \vdots \\
    x(N) - f(x(N - 1), u(N - 1), \hat{d}(N - 1))
\end{bmatrix}
\]

such that the equality constraint (7.1b) is simply \(g(X, U) = 0\). The total number of equality constraints is \(n_{eq} = N n\). Similarly we lump the inequality constraints:

\[
h(X, U) = \begin{bmatrix} 
    H_u u(0) - h_u \\
    H_x x(1) - h_x \\
    \vdots \\
    H_u u(N - 1) - h_u \\
    H_f x(N) - h_f
\end{bmatrix}
\]

such that the inequality constraints (7.1c),(7.1d),(7.1e) are summarized as \(h(X, U) \leq 0\).

Let \(H_x \in \mathbb{R}^{n_x \times n}, H_u \in \mathbb{R}^{n_u \times m}, H_f \in \mathbb{R}^{n_f \times n}\). Then, the number of inequality constraints \(n_{in} = (N - 1)n_x + N n_u + n_f\). Finally, we write the cost function as:

\[
J(X, U) = p(x(N)) + \sum_{k=0}^{N-1} q(x(k), u(k)).
\]
and formulate the Lagrangian:

$$\mathcal{L}(X, U, \nu, \lambda) = J(X, U) + \nu^T g(X, U) + \lambda^T h(X, U)$$ (7.5)

where $\nu$ is the dual variable associated with equality constraints $g(X, U) = 0$ and $\lambda$ is the dual variable associated with inequality constraints $h(X, U) \leq 0$.

The KKT conditions can then be written as follows:

$$\nabla J(X, U) + \nu^T (\nabla g(X, U)) + \lambda^T (\nabla h(X, U)) = 0$$ (7.6a)

$$g(X, U) = 0$$ (7.6b)

$$h(X, U) \leq 0$$ (7.6c)

$$\lambda \geq 0$$ (7.6d)

$$\lambda^T h(X, U) = 0$$ (7.6e)

where the first equation (7.6a) is the stationarity condition, the next conditions (7.6b), (7.6c) are the primal feasibility conditions, the inequality (7.6d) is the dual feasibility condition, and (7.6e) is the complementary slackness condition. For interior point methods, it is common to introduce a slack variable $s$ to move the inequalities in $h(X, U)$ into the equality constraints:

$$\nabla J(X, U) + \nu^T (\nabla g(X, U)) + \lambda^T (\nabla h(X, U)) = 0$$ (7.7a)

$$g(X, U) = 0$$ (7.7b)

$$h(X, U) + s = 0$$ (7.7c)

$$\lambda \geq 0$$ (7.7d)

$$s \geq 0$$ (7.7e)

$$\lambda^T s = 0.$$ (7.7f)

This addition of a slack variable $s$ makes the complementary slackness (7.7f) condition much cleaner than its previous form (7.6e).

### 7.2 Inexact Interior Point Method

In this section, we describe an inexact interior point method proposed by Bellavia [131]. The main idea of an interior point method is to use a Newton method for the equality constraints (7.7a),(7.7b),(7.7d),(7.7f) with a backtracking linesearch to satisfy inequality constraints (7.7d),(7.7e) and improvement of a merit function. This approach iteratively solves for the solution to the nonlinear KKT equations (7.7) except that the complementary slackness condition (7.7f) is set equal to a centering parameter $\mu$:

$$\lambda^T s = \mu.$$ (7.8)
The parameter $\mu$ tends to zero over the iterations of the interior point algorithm in order to approach a solution to the KKT conditions (7.6).

In an inexact interior point method, the Newton step direction is computed inexactly. That is, there is room for some error in the computation of the step direction. With careful design of the interior point method, the inexact interior point method is guaranteed to converge. A globally convergent inexact interior point method is formulated and proven to converge in [131]. We summarize the steps of this method next.

First, define the KKT system of (7.7) with the variable $Z = [X^T \ U^T]^T$:

$$
\begin{bmatrix}
\nabla^2 \mathcal{L}(Z) & \nabla g(Z)^T & \nabla h(Z)^T & 0 \\
\nabla g(Z) & 0 & 0 & I \\
\nabla h(Z) & 0 & 0 & S \\
0 & 0 & S & \Lambda
\end{bmatrix}
\begin{bmatrix}
\Delta Z \\
\Delta \nu \\
\Delta \lambda \\
\Delta s
\end{bmatrix} =
\begin{bmatrix}
-(\nabla J(Z) + \nu^T \nabla g(Z) + \lambda^T \nabla h(Z)) \\
-g(Z) \\
-(h(Z) + s) \\
-(\lambda^T s) + \mu
\end{bmatrix}
$$

(7.9)

where $\nabla$ indicates a partial derivative with respect to $Z$ and $S = \text{diag}(s)$ is a matrix with the vector $s$ on its diagonal and similarly, $\Lambda = \text{diag}(\lambda)$. We also define the KKT residual,

$$H(Z, \nu, \lambda, s) =
\begin{bmatrix}
\nabla J(Z) + \nu^T (\nabla g(Z)) + \lambda^T (\nabla h(Z)) \\
g(Z) \\
h(Z) + s \\
\lambda^T s
\end{bmatrix},$$

(7.10)

which we would like to see go to zero as the algorithm progresses. It will also be useful to encapsulate the stationarity constraint and primal feasibility equality constraints separately from the complementary slackness constraint:

$$F(Z, \nu, \lambda, s) =
\begin{bmatrix}
\nabla J(Z) + \nu^T (\nabla g(Z)) + \lambda^T (\nabla h(Z)) \\
g(Z) \\
h(Z) + s
\end{bmatrix}.$$

(7.11)

The algorithm begins with an initial guess at iteration $t = 0$: $(Z^0, \mu^0, \lambda^0, s^0)$. Some linesearch parameters are also initialized: $\bar{\tau}_1, \bar{\tau}_2, \gamma_{t-1} \in [0.5, 1), \eta_{\text{max}} \in (0, 1), \beta \in (0, 1), \theta \in (0, 1)$. The first two parameters are defined as follows:

$$\begin{align*}
\bar{\tau}_1 &= \min(\lambda^0, * s^0)/(\lambda^0 s^0/n_{\text{in}}) \\
\bar{\tau}_2 &= (\lambda^0 s^0/\|F(Z, \nu, \lambda, s)\|)
\end{align*}
$$

(7.12a)

(7.12b)

where $\lambda^0, * x^0$ is the element-wise product of the elements of vectors $\lambda^0, x^0$ and $\min(\lambda^0, * x^0)$ is the minimum value of any element of the vector argument.

Given the latest iterate of $(Z^t, \nu^t, \lambda^t, s^t)$, the centering parameter $\mu$ and the parameter $\bar{\eta}^t$ are updated:

$$\begin{align*}
\sigma^t + \bar{\eta}^t &\in (0, \eta_{\text{max}}) \\
\gamma^t &\in [1/2, \gamma^{t-1}]
\end{align*}
$$

(7.13a)

(7.13b)
\[ \mu^t = \sigma^t(s^t)^T(\lambda^t)/n_{in} \] 
\[ \tilde{\eta}^t = \sigma^t + \tilde{\eta}^t. \] 

Next, the KKT system is solved inexactly. That is, the following linear system is solved:

\[ \nabla H(Z^t, \nu^t, \lambda^t, s^t) \begin{bmatrix} \Delta Z \\
\Delta \nu \\
\Delta \lambda \\
\Delta s \end{bmatrix} = -H(Z^t, \nu^t, \lambda^t, s^t) + \begin{bmatrix} 0 \\
0 \\
0 \\
\mu^t \end{bmatrix} + \hat{r}^t \] \hspace{1cm} (7.14)

where \( \hat{r}^t \) is the inaccuracy or error that is permitted by the linear solver at the \( t \)-th step of the interior point algorithm. This error must be bounded by

\[ \| \hat{r}^t \| \leq \tilde{\eta}^t(s^t)^T \lambda^t/n_{in}. \] \hspace{1cm} (7.15)

This means that the KKT system (7.9) can be solved inexactly with error bounded by (7.15). Thus, an iterative algorithm attempting to solve (7.9) can terminate once this accuracy has been reached.

Next in the interior point method, a backtracking linesearch is done. First, a step size \( \bar{\alpha}_1 \in (0, 1) \) is chosen to be as large as possible while satisfying:

\[ \min((s^t + \bar{\alpha}_1^t \Delta s) \cdot (\lambda^t + \bar{\alpha}_1^t \Delta \lambda) \geq \bar{\tau}_1 \gamma^t(s^t + \bar{\alpha}_1^t \Delta s)^T(\lambda^t + \bar{\alpha}_1^t \Delta \lambda)/n_{in}. \] \hspace{1cm} (7.16)

The condition set by (7.16) is a typical path following constraint, which helps the elements of the complementary slackness condition \( s_i \lambda_i \) approach zero at approximately the same rate (see [135] Chapter 14.1). This condition can be solved for analytically by finding the roots of the following equations:

\[ p_{2,i}(\bar{\alpha}_1^t)^2 + p_{1,i}(\bar{\alpha}_1^t) + p_{0,i} = 0 \hspace{1cm} \forall \hspace{1cm} i = 1, \ldots, n_{in}, \] \hspace{1cm} (7.17a)

\[ p_{2,i} = (\Delta s(i))(\Delta \lambda(i)) - \bar{\tau}_1 \gamma^t(\Delta s)^T(\Delta \lambda)/n_{in} \] \hspace{1cm} (7.17b)

\[ p_{1,i} = (\Delta s(i))(\lambda^t(i)) + (s^t(i))(\Delta \lambda(i)) - \bar{\tau}_1 \gamma^t((\Delta s)^T(\lambda^t) + (s^t)^T(\Delta \lambda))/n_{in} \] \hspace{1cm} (7.17c)

\[ p_{0,i} = (\lambda^t(i))(s^t(i)) - \bar{\tau}_1 \gamma^t(\Delta s)^T(\Delta \lambda)/n_{in} \] \hspace{1cm} (7.17d)

and then finding the minimum \( \bar{\alpha}_1^t \) that is non-negative. Each polynomial (7.17a) should return at least one positive \( \bar{\alpha}_1 \) if the parameters \( \sigma^t \) and \( \tilde{\eta}^t \) are chosen to satisfy certain assumptions (see Theorem 4).

Next, the step size \( \bar{\alpha}_2 \in (0, 1) \) is chosen to be as large as possible while satisfying:

\[ \min((s^t + \bar{\alpha}_2^t \Delta s)^T(\lambda^t + \bar{\alpha}_2^t \Delta \lambda) \geq \bar{\tau}_2 \gamma^t\|F((Z^t + \bar{\alpha}_2^t \Delta Z), (\nu^t + \bar{\alpha}_2^t \Delta \nu), (\lambda^t + \bar{\alpha}_2^t \Delta \lambda), (s^t + \bar{\alpha}_2^t \Delta s))\|. \] \hspace{1cm} (7.18)

This is also a central path following constraint, similar to (7.16) [135]. Both (7.16) and (7.18) are important for establishing global convergence of the algorithm [136, 131]. It is generally
not possible to solve for step size $\bar{\alpha}_2$ analytically. We implement a simple incremental backtrack linesearch, starting from $\bar{\alpha}_2$, initially decreasing by 0.1 for a few iterations and then dividing by 2 until the condition (7.18) is satisfied.

The minimum of these two step sizes is determined: $\tilde{\alpha}^t = \min(\bar{\alpha}^t_1, \bar{\alpha}^t_2)$. The search direction is updated to $p^t = \tilde{\alpha}^t[\Delta Z^T, \Delta \nu^T, \Delta \lambda^T, \Delta s^T]^T$ and one more backtracking line search is run. This last search makes sure that the KKT residual decreases from the previous step to the next step. This is a typical merit function filter, where the merit function is the norm of the KKT residual $\|H(Z, \nu, \lambda, s)\|^2_2$. The search direction $p^t$ and parameter $\eta^t$ are iteratively update by:

$$p^t \leftarrow \theta p^t \quad (7.19a)$$
$$\eta^t \leftarrow 1 - \theta(1 - \eta^t) \quad (7.19b)$$

until the following condition is satisfied

$$\|H([(Z^t)^T, (\nu^t)^T, (\lambda^t)^T, (s^t)^T]^T + p^t)\| \leq (1 - \beta(1 - \eta^t))\|H([(Z^t)^T, (\nu^t)^T, (\lambda^t)^T, (s^t)^T]^T)\|. \quad (7.20)$$

The algorithm then goes back to (7.13) and repeats through all of the following steps until the norm of the KKT residual is small enough $\|H(Z', \nu', \lambda', s')\| < \epsilon$ or the maximum number of iterations $\text{maxiters}$. The steps of the inexact interior point algorithm are summarized in 7.

**Algorithm 7 Inexact Interior Point Method**

1: Choose algorithm parameters $\gamma_{t-1} \in [0.5, 1), \eta_{\text{max}} \in (0, 1), \beta \in (0, 1), \theta \in (0, 1)$.
2: Compute $\bar{\gamma}_1, \bar{\gamma}_2$ by equations (7.12).
3: while $\|H(Z', \nu', \lambda', s')\|^2_2 > \epsilon$ OR $t < \text{maxiters}$ do
4: Update the centering parameter $\mu^t$ and $\eta^t$ by (7.13).
5: Compute the step direction inexactly (7.14) with error bound (7.15).
6: Compute step sizes $\bar{\alpha}^t_1$ and $\bar{\alpha}^t_2$ that satisfy (7.16) and (7.18), respectively.
7: Find the minimum step size $\tilde{\alpha}^t = \min(\bar{\alpha}^t_1, \bar{\alpha}^t_2)$.
8: Update the search direction to $p^t = \tilde{\alpha}^t[\Delta Z^T, \Delta \nu^T, \Delta \lambda^T, \Delta s^T]^T$.
9: Complete a backtracking linesearch to improve the merit function $\|H(Z, \nu, \lambda, s)\|$ by adjusting $p^t, \eta^t$ by (9) until condition (7.20) is met.
10: $t \leftarrow t + 1$.
11: end while

We now summarize some useful results from [131] and in particular highlight the assumptions that must be placed on the MPC problem (7.1).

**Theorem 4.** Assume the following properties:

1. The KKT residual $H(Z, \nu, \lambda, s)$ (7.10) is continuously differentiable.
2. The iteration sequence of \((Z^t, \nu^t, \lambda^t, s^t)\) produced by Algorithm 7 is bounded.

3. The Jacobian of \(H(Z, \nu, \lambda, s)\) (7.10) is nonsingular along iterates \((Z^t, \nu^t, \lambda^t, s^t)\) of the algorithm.

4. The derivative \(\nabla F\) is Lipschitz continuous.

5. The parameters \(\sigma^t\) and \(\hat{\eta}^t\) satisfy:

\[
\sigma^t > \hat{\eta}^t \max \left( \frac{\sqrt{n_{in}} + \bar{\tau}_1 \gamma^t}{\sqrt{n_{in}(1 - \bar{\tau}_1 \gamma^t)}}, \frac{\bar{\tau}_2 \gamma^t + \sqrt{n_{in}}}{n_{in}} \right).
\] (7.21)

Then, Algorithm 7 is a globally convergent inexact interior point method to a point \((Z^*, \nu^*, \lambda^*, s^*)\) where \(\|H(Z^*, \nu^*, \lambda^*, s^*)\| \to 0\), provided that maxiters is sufficiently large. The step sizes \(\bar{\alpha}_1^t\) and \(\bar{\alpha}_2^t\) are bounded away from zero. Moreover, the inexact interior point method exhibits superlinear local convergence to the solution.

Proof. A detailed proof is provided in [131]. We remark that the last assumption can be guaranteed by design of \(\sigma^t\) and \(\hat{\eta}^t\).

We note that there are variants of Theorem 4 presented in [131] that use less strong assumptions than those stated here. We simplify some of the assumption statements since they are met by the MPC problem formulation in Section 7.1. We discuss some assumptions in context of Nonlinear MPC next.

Remark 14. Under the assumptions that the MPC problem (7.1) has \(f(\cdot)\) is twice differentiable with its second derivative being continuous, and the cost functions are twice differentiable with the second derivative being continuous, the Assumptions 1, 4 of Theorem 4 are satisfied.

Proof. Recall the KKT residual from (7.10):

\[
H(Z, \nu, \lambda, s) = \begin{bmatrix}
\nabla J(Z) + \nu^T(\nabla g(Z)) + \lambda^T(\nabla h(Z)) \\
g(Z) \\
h(Z) + s \\
\lambda^T s
\end{bmatrix}.
\]

By assumption, \(J(Z)\) is continuously differentiable with its second derivative being Lipschitz continuous. Similarly, \(g(Z)\) is an affine function of \(f(\cdot)\) and is thus also twice continuously differentiable with \(\nabla g(z)\) being continuously differentiable and therefore Lipschitz continuous. The inequalities encoded in \(h(Z)\) are affine in \(Z\) since they consist of polyhedrons (see (7.3)). Thus, \(h(Z)\) also is continuously differentiable with its derivative being Lipschitz continuous. The terms in the first three block rows are linear combinations of \(\nabla J(Z), \nabla g(Z), \nabla h(Z)\), so they satisfy assumptions 1 and 4. The last block row is continuously differentiable in both \(\lambda\) and \(s\) as it is linear in each variable.
Remark 15. The iterates of the algorithm \((Z^t, \nu^t, \lambda^t, s^t)\) are bounded if \(\|H(Z, \nu, \lambda, s)\|\) is radially unbounded.

Proof. The function \(\|H(Z, \nu, \lambda, s)\|\) is radially unbounded if \(\|(Z^t, \nu^t, \lambda^t, s^t)\| \rightarrow \infty \implies \|H(Z, \nu, \lambda, s)\| \rightarrow \infty\). By definition, the iterates of the algorithm have bounded \(\|H(Z, \nu, \lambda, s)\|\) because \(\|H(Z^t, \nu^t, \lambda^t, s^t)\| \leq \|H(Z^0, \nu^0, \lambda^0, s^0)\|\) for all \(t \geq 0\). If \(\|H(Z, \nu, \lambda, s)\|\) is not infinite, then \(\|(Z^t, \nu^t, \lambda^t, s^t)\|\) cannot be infinite by the contrapositive of the radially unbounded assumption. 

Assumption 3 of Theorem 4 is satisfied for specific forms of problem (7.1); one example is MPC with quadratic cost and linear constraints that are linearly independent. However, in general, it is difficult to guarantee the assumption 3. In some cases, the singularity of the KKT system can be modified by adding \(\delta I\) to some diagonal block components of the matrix to improve conditioning (see [135] Chapter 19.3 or [127]).

7.3 Iterative Linear Algebra Methods

The inexact interior point algorithm allows an inexact computation of the KKT system (7.14). This inexact computation enables the usage of iterative linear algebra methods including the Jacobi method, the Gauss-Seidel method, and Krylov Subspace methods [76, 92]. While methods such as the Jacobi method or the Gauss-Seidel method are intuitive and simple to code, the Jacobi method only works for linear systems that have a diagonally dominant matrix and the Gauss-Seidel method is only guaranteed to converge for linear systems where the matrix is diagonally dominant or symmetric positive definite [92]. Moreover, the Jacobi method and the Gauss-Seidel method are particularly slow to converge, especially when compared with Krylov subspace methods.

Krylov subspace methods are the most successful iterative methods for numerical linear algebra [92]. The most well-known algorithm is the conjugate gradient method. The conjugate gradient method is particularly efficient because it uses very little memory storage over the iterations of the algorithm. Its convergence properties are particularly well understood; convergence depends on the condition number of the matrix in the linear system. However, conjugate gradient specifically applies to linear systems with a positive definite matrix.

The KKT system for a nonlinear optimization problem (7.9) is not positive definite. In fact, it is indefinite, meaning it has both positive and negative eigenvalues. One approach for dealing with an indefinite matrix is to solve the normal equations with the conjugate gradient method. That is, instead of solving \(Ax = b\), the normal equations \(A^T A x = A^T b\) are solved instead. This works well if the condition number of \(A\) is good, but if the condition number of \(A\) is bad, then the condition number of \(A^T A\) will be worse.

For solving linear systems with nonsymmetric indefinite matrices, the existing iterative linear algebra methods include the generalized minimum residual algorithm GMRES [137] and quasi-minimum residuals [138]. These methods can require substantially more memory as well as more complex computations than the conjugate gradient method uses. If the matrix
is indefinite, but symmetric, there are two Krylov subspace methods that operate more closely to the efficiency of the conjugate gradient method: the minimum residual algorithm MINRES and the symmetric LQ method SYMMLQ [130]. The MINRES algorithm aims to minimize \( \|r^t\| \) (7.14) whereas SYMMLQ ensures that the residual \( r^t \) is orthogonal to the Krylov subspace at time step \( t \).

The KKT system (7.9) consists of a symmetric indefinite matrix if the last line of the matrix is eliminated and if \( \nabla^2 L(Z) \) is symmetric. The last line of the matrix bears the following equation:

\[
S \Delta \lambda + \Lambda \Delta s = -s^T \lambda + \mu \tag{7.22}
\]

Since both \( S \) and \( \Lambda \) are a diagonal matrix of strictly positive elements, they are invertible and therefore we can easily write \( \Delta \lambda \) in terms of \( \Delta s \) or vice versa. Eliminating \( \Delta s \) is a standard approach since its only coefficient in the KKT system (7.9) is an identity matrix. Thus \( \Delta s \) can be substituted out by using the following equation:

\[
\Delta s = -\Lambda^{-1} S \Delta \lambda - \Lambda^{-1} s^T \lambda + \mu \tag{7.23}
\]

where \( 1 \) is a vector of ones of appropriate size. Then the KKT system can be written in a condensed form:

\[
\begin{bmatrix}
\nabla^2 L(Z) & \nabla g^T(Z) & \nabla h^T(Z) \\
\nabla g(Z) & 0 & 0 \\
\nabla h(Z) & 0 & -\Lambda^{-1} S
\end{bmatrix}
\begin{bmatrix}
\Delta Z \\
\Delta \nu \\
\Delta \lambda
\end{bmatrix}
= \begin{bmatrix}
- (\nabla J(Z) + \nu^T \nabla g(Z) + \lambda^T \nabla h(Z)) \\
-g(Z) \\
-h(Z) - \Lambda^{-1} \mu
\end{bmatrix}
\tag{7.24}
\]

This condensed form (7.24) now consists of a symmetric indefinite matrix if \( \nabla^2 L(Z) \) is symmetric.

For the nonlinear MPC problem (7.1), \( \nabla^2 L(Z) \) is symmetric if \( \nabla^2 J(Z) + \nu^T \nabla^2 g(Z) \) is symmetric. Note that \( \nabla^2 L(Z) \) of the MPC problem does not include \( \nabla^2 h(Z) \) because \( h(Z) \) is an affine equation (since inequalities are polyhedrons, c.f. (7.3)) and thus the second derivative of \( h(Z) \) is zero. A theorem from analysis states that if the second partial derivatives of a function are continuous, then the Hessian of that function is symmetric. This theorem is known as Schwarz’s theorem or Clairaut’s theorem and is summarized here [139].

**Theorem 5** (Schwarz’s theorem or Clairaut’s theorem). If a function \( f(Z) : \mathbb{R}^{nz} \to \mathbb{R} \) has continuous second partial derivatives at any point \( Z \in \mathbb{R}^{nz} \), then the second partial derivatives are symmetric, i.e. \( \frac{\partial^2}{\partial Z_i \partial Z_j} f(Z) = \frac{\partial^2}{\partial Z_j \partial Z_i} f(Z) \). Moreover, the Hessian \( \nabla^2 f(Z) \) is symmetric for all \( Z \).

**Proof.** See an advanced mathematical analysis book (such as [139, 140]) for a detailed discussion and/or proof.

Since it is assumed that both the cost function \( J(Z) \) and the dynamics \( f(Z) \) are twice differentiable with continuous second derivatives, we can use Schwarz’s theorem to claim that \( \nabla^2 L(Z) \) is symmetric and therefore we can use iterative linear algebra methods for symmetric matrices, such as MINRES or SYMMLQ.
7.4 Preconditioners

The performance of iterative linear algebra methods depend heavily on the condition number of the matrix. If the $A$ matrix of $Ax = b$ has a bad condition number, the matrix itself can be *preconditioned* in an effort to improve the condition number. The main idea of preconditioning is that instead of solving $Ax = b$, a transformed problem is instead solved:

$$M^{-1}Ax = M^{-1}b$$

where $M$ is an invertible matrix. Then, the condition number of $M^{-1}A$ is what affects the speed of the iterative solver. This equation (7.25) is when the linear system is left conditioned. In some cases, the system may be *right* conditioned:

$$AM^{-1}x = b$$

and then the linear system $AM^{-1}y = b$ is solved where $y = Mx$. Sometimes, the linear system is both left and right preconditioned:

$$M^{-1}AM^{-T}M^Tx = M^{-1}b.$$  

However, eigenvalues of $M^{-1}AM^{-T}$ are the same as the eigenvalues of $(MM^T)^{-1}A$ and $A(MM^T)^{-1}$. Thus, a left and right preconditioner can be recast as a left or right preconditioner $(MM^T)^{-1}$.

In the literature, there are suggested preconditioners $M$, most of which are fairly problem specific. Of particular relevance to this work are preconditioners for interior point methods [141], preconditioners for indefinite KKT systems [142], and preconditioners for KKT systems arising from optimal control problems [132].

We consider preconditioners for optimal control from [132]. The preconditioners are designed for a KKT matrix of the following form:

$$A_{KKT} = \begin{bmatrix} H_x & 0 & A^T \\ 0 & H_u & B^T \\ A & B & 0 \end{bmatrix}$$

(7.28)

where $A$ is an invertible matrix. We show that the nonlinear MPC problem has this KKT matrix under a few small assumptions. For the nonlinear MPC problem, we have reduced the KKT matrix to:

$$A_{KKT} = \begin{bmatrix} \nabla^2 \mathcal{L}(Z) & \nabla g^T(Z) & \nabla h^T(Z) \\ \nabla g(Z) & 0 & 0 \\ \nabla h(Z) & 0 & -\Lambda^{-1}S \end{bmatrix}$$

(7.29)
where the second matrix separates $Z$ into $X$ and $U$. We would like to show that (7.29) is the same form as (7.28).

First, note that the bottom left block matrix of (7.29) needs to be square and invertible. Both $\nabla_X h(X,U)$ and $\nabla_U h(X,U)$ are constant matrices since they are derived from polyhedrons (7.3). However, neither of these matrices are necessarily square. On the other hand, the matrix $\nabla_X g(X,U)$ is square as it is defined by dynamics (7.2), and in many cases it is invertible. Thus, we eliminate the last row of the (7.28) by eliminating $\Delta \lambda$ as a variable. The procedure is similar to that shown for eliminating $\Delta s$ (as done in (7.23)), so we skip these steps and write the further reduced KKT system:

\[
\begin{bmatrix}
\nabla_{XX} \mathcal{L}(X,U) + \nabla_X h^T(Z) \Lambda X S_X^{-1} \nabla_X h(X,U) & \nabla_{XU} \mathcal{L}(X,U) & \nabla_{Xg}(X,U) \\
\nabla_{UX} \mathcal{L}(X,U) & \nabla_{UU} \mathcal{L}(X,U) + \nabla_U h^T(Z) \Lambda U S_U^{-1} \nabla_U h(X,U) & \nabla_{Ug}(X,U) \\
\nabla_{Xg}(X,U) & \nabla_{Ug}(X,U) & 0
\end{bmatrix}
\begin{bmatrix}
\Delta X \\
\Delta U \\
\Delta \nu
\end{bmatrix} =
\begin{bmatrix}
-(\nabla_X f(X,U) + \nu X \nabla_X g(X,U) + \lambda X \nabla_X h(X,U)) - \nabla_X h^T(Z) \Lambda X S_X^{-1} h(X,U) - \nabla_X h^T(Z) S_X^{-1} \mu \\
-(\nabla_U f(X,U) + \nu U \nabla_U g(X,U) + \lambda U \nabla_U h(X,U)) - \nabla_U h^T(Z) \Lambda U S_U^{-1} h(U,X,U) - \nabla_U h^T(Z) S_U^{-1} \mu \\
-g(X,U)
\end{bmatrix}
\] (7.30)

This further reduced KKT system (7.30) matches (7.28) if the following assumptions are made.

**Assumption 1.** Assume for the KKT system (7.30) the following:

1. $\nabla_{XU} \mathcal{L}(X,U) = (\nabla_{UX} \mathcal{L}(X,U))^T = 0$.

2. $\nabla_X g(X,U)$ is invertible.

The first assumption holds if the cost function is separable in $X$ and $U$ and the dynamics are in the following form:

\[x(k+1) = f_X(x(k)) + f_U(x(k))u(k)\] (7.31)

This form of nonlinear dynamics is commonly used for nonlinear systems and is known as an affine system since it is affine in the control input. In some cases, a first-order filter can be added to a general nonlinear system to obtain an affine system [143]. The second assumption holds if $\nabla_X f(x(k))$ is invertible. This is usually true if the dynamics are linear and the state space is of minimum representation, but may not be true in general. If these assumptions do not hold, we recommend investigating another preconditioning option, such as those mentioned earlier.

We make the assumptions defined by Assumption 1 and introduce the first of three preconditioners proposed in [132]. The other preconditioners are valid, but the first preconditioner is significantly simpler than the other two. The preconditioner is defined by equation (7.27) where $M^{-1}$ is:

\[
M^{-1} = \begin{bmatrix}
H_x^{-1/2} & 0 & 0 \\
0 & H_x^{-1/2} & 0 \\
0 & 0 & H_x^{1/2} A^{-1}
\end{bmatrix}
\] (7.32)
where \( A = \nabla_X g(X,U) \), \( H_x = \nabla_{XX} \mathcal{L}(X,U) \) and \( H_u = \nabla_{UU} \mathcal{L}(X,U) \). The first two block diagonal entries clearly cancel the ill-conditioning that may be present in \( H_x \) and \( H_u \). The last block diagonal entry reduces the effect of the first block matrices of \( M^{-1} \) on \( A \) and \( B \). This preconditioner requires the square root inverse of \( H_x, H_u \), and the inverse of \( A \). Usually, an approximation is used to avoid too much computational overhead. Such methods include the truncated Neumann series or an incomplete Cholesky factorization [144, 92, 132].

The preconditioner can be applied directly to the \( A,b \) matrices before the iterative linear algebra method commences, or the preconditioner can be directly handed to a preconditioned MINRES or preconditioned SYMMLQ method. In the latter case, the matrix \((MM^T)^{-1}\) is not handed to the preconditioner, but \( MM^T \) is handed instead. This is particularly useful because then inverses of \( H_u \) and \( A \) need not be computed or otherwise approximated. For example, for the preconditioner (7.32), we have

\[
MM^T = \begin{bmatrix}
H_x & 0 & 0 \\
0 & H_u & 0 \\
0 & 0 & AH_x^{-1}A^T
\end{bmatrix}.
\] (7.33)

This latter option is especially recommended if \( MM^T \) is sparse and linear systems of the form \( MM^T x = b \) can be solved easily. The system \( MM^T x = b \) is easy to compute when direct factorization of \( MM^T \) is fast and results in a sparse solution. This is generally much faster than computing the inverse of \( MM^T \).

### 7.5 Distributed Computation

In the distributed implementation of the inexact interior point method, the main step to be parallelized is the iterative numerical linear algebra that is used to compute step 5 of Algorithm 7. The backtrack search in steps 6 and 9 of Algorithm 7 can also be implemented in parallel in the same way. In this section, we describe how these steps can be parallelized.

Iterative linear algebra is trivial to parallelize. The biggest operation that occurs in Krylov subspace methods such as SYMMLQ and MINRES is the multiplication of the KKT matrix with a vector, e.g. the computation of \( A_{KKT}v \) where \( v \) is some vector of appropriate dimension. Consider the following structure of the problem:

\[
A_{KKT}v = \begin{bmatrix} A_1 & A_2 & \ldots & A_{np} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{np} \end{bmatrix}.
\] (7.34)

Then each \( A_iv_i \) operation can be computed completely separately and in parallel. In this case, there are clearly \( np \) matrix vector operations which can occur in parallel.

The parallel implementation of SYMMLQ and MINRES has two components: a central coordinator handles some very simple computations and distributed processors handle parallel matrix-vector multiplication. Thus, this will require scatter and gather operations over
a star communication network. This is similar to the distributed algorithm proposed for the primal-dual active-set method in Chapter 4. In a single iteration, the iterative algorithm comes up with a vector \(v\), scatters it to the distributed processors, the processors compute \(A_{KKT}v\) in parallel, and then the resultant vector is gathered centrally for the next iteration.

The beauty of this trivial parallelization is that the parallel computation of \(A_{KKT}v\) can be done for any splitting of the vector \(v\). For example, the matrix vector operation can be split temporally, e.g. \(A_{i}\) are the columns of \(A_{KKT}\) that apply to the variables \([x(i)^Tu(i)^Tu(i)^Tu(i)^Tu(i)^T]\) in (7.30). This option is popularly used by algorithms tailored for MPC because of its temporal multi-stage nature [125].

Additionally, for large-scale systems, the problem data may be spread out spatially, as shown in the introduction of this thesis. In that case, each subsystem may know only its dynamics and constraints. Thus, it may be desirable to split the matrix vector computation spatially with \(A_{i}\) consisting of the columns of \(A_{KKT}\) that apply to \([x(1)^Tu(0)^Tu(0)^Tu(0)^Tu(0)^T]\) in (7.30). This is commonly proposed as the splitting for algorithms such as dual decomposition or ADMM applied to MPC [28, 16, 30].

In general, the splitting of the matrix vector operation can be optimized for the computing platform it is run on (e.g. distributed embedded processors, cloud computing, super computing, GPU, etc.). Packages such as Petsc [145] or Pardiso [146] will automatically optimize the parallel computation of the matrix-vector operations for the certain distributed computing platforms.

The backtracking searches of the inexact interior point algorithm are similarly trivially parallelizable. Specifically, the evaluation of \(F(Z, \nu, \lambda, s)\) and \(H(Z, \nu, \lambda, s)\) can be done in parallel. The iterate \((Z, \nu, \lambda, s)\) can be split spatially, temporally, or otherwise. The backtracking search to satisfy condition (7.18) updates \(\alpha_{2}\) and \((Z^{t} + \bar{\alpha}Z), (\nu^{t} + \bar{\alpha}\nu), (\lambda^{t} + \bar{\alpha}\lambda), (s^{t} + \bar{\alpha}s)\) centrally, scatters \((Z^{t} + \bar{\alpha}_{2}Z), (\nu^{t} + \bar{\alpha}_{2}\nu), (\lambda^{t} + \bar{\alpha}_{2}\lambda), (s^{t} + \bar{\alpha}_{2}s)\) to the appropriate processors where \(F((Z^{t} + \bar{\alpha}_{2}Z), (\nu^{t} + \bar{\alpha}_{2}\nu), (\lambda^{t} + \bar{\alpha}_{2}\lambda), (s^{t} + \bar{\alpha}_{2}s))\) is computed in parallel and then gathered to the central hub where condition (7.18) is checked. The merit function backtracking search in step (9) of Algorithm 7 can be done similarly.

### 7.6 Application: Freeway Ramp Metering

A main motivation of this work is the freeway ramp metering problem introduced in Chapter 6. Recall the equations of the hybrid model predictive control problem for ramp metering:

\[
J_{0}^{*}(x_{0}) = \min_{\bar{X}(k), \bar{U}(k)} \bar{X}(H)^{T}P\bar{X}(H) + \sum_{k=0}^{H-1} \bar{X}(k)^{T}Q\bar{X}(k) + \bar{U}(k)^{T}R\bar{U}(k)
\]  

\(s.t.\) \(N(k+1) = A_{i}N(k) + B_{i}R(k) + g_{i}\) if \((N(k), R(k)) \in \mathcal{R}_{i}\)

\(L(k+1) = L(k) + D(k) - R(k)\)

\(0 \leq N(k) \leq \bar{N}\)

\(0 \leq L(k) \leq \bar{L}\)
Unfortunately, the MPC design of equation (7.35) is a mixed-integer quadratic program (MIQP) (this can be more clearly seen by transforming the piecewise affine dynamics into mixed-integer logic, e.g. see [7]) which is an NP hard problem. Moreover, the ramp metering problem is very large in reality, on the order of 1,000 to 100,000 variables [3]. In order to compute a solution in real-time for this very large problem, we apply smoothing methods and the distributed interior point algorithm introduced in this chapter.

Smoothing approximation

We apply a conjugate smoothing technique [133, 134], to smooth the nonlinearity present in the ramp metering problem. The main nonlinearity in the ramp metering problem comes from the equations for flow from one link of the freeway to the next. For example, the flow from link $i$ to link $i+1$ at time step $k$ is defined by:

$$f_i(k) = \min \{ \bar{\beta}_i v_i (n_i(k) + \gamma r_i(k)), w_{i+1} (\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k)), \bar{f}_i \}. \quad (7.36)$$

This function is the minimization of three functions which are affine in $n_i(k), n_{i+1}(k), r_i(k), r_{i+1}(k)$.

Conjugate smoothing methods [133, 134] allow us to approximate the minimization of affine functions as a logarithm of a summation of exponents. In particular, the approximation of $f_i(k)$ is:

$$f_{\mu,i}(k) = -\mu \log \left( e^{\frac{-1}{\mu} (\bar{\beta}_i v_i (n_i(k) + \gamma r_i(k)))} + e^{\frac{-1}{\mu} w_{i+1} (\bar{n}_{i+1} - n_{i+1}(k) - \gamma r_{i+1}(k))} + e^{\frac{-1}{\mu} \bar{f}_i} \right) + \mu \log 3 \quad (7.37)$$

where $\mu$ is a small valued parameter which trades off between accuracy of the approximation and smoothness. Note that this approximation is infinitely continuously differentiable; it has infinite derivatives which are all continuous making it a smooth function.

Then, the piecewise affine dynamics encoded in the hybrid MPC (7.35b) are replaced by:

$$n_i(k+1) = n_i(k) + f_{\mu,i-1}(k) + r_i(k) - f_{\mu,i}(k) \quad (7.38)$$

This approximates the hybrid MPC problem as a smooth nonlinear non-convex optimization problem.
Preconditioner for Ramp Metering

In this section, we make some remarks about approximations used for the preconditioner for the ramp metering problem. Matrices $H_x$ and $H_u$ are almost completely diagonal if $P, Q, R$ are diagonal matrices and the constraints are box constraints. This is because

$$H_x = \nabla_{XX} \mathcal{L}(X,U) + \nabla_X h^T(X,U) \Lambda_X S_X^{-1} \nabla_X h(X,U)$$

$$= \begin{bmatrix} Q & & \\ Q & \ddots & \\ & \ddots & P \end{bmatrix} + \nu^T \nabla_{XX} g(X,U) + [I - I] \Lambda_X S_X^{-1} \begin{bmatrix} I \\ -I \end{bmatrix}.$$  (7.39)

The first term is clearly diagonal and the last term is clearly diagonal (recall $\Lambda$ and $S$ are both diagonal matrices). The middle term may cause some off-diagonal terms to appear. To circumvent this issue, we approximate $H_x$ by utilizing the diagonal terms of $\nu^T \nabla_{XX} g(X,U)$.

Thus, approximate $H_x$ with $\tilde{H}_x$

$$\tilde{H}_x = \nabla_{XX} J(X,U) + \text{diag}(\nu^T \nabla_{XX} g(X,U)) + \nabla_X h^T(X,U) \Lambda_X S_X^{-1} \nabla_X h(X,U)$$  (7.40)

and used in the preconditioner $M$ (7.32). The use of diagonal terms can also be done for the $P$ matrix in the case that it is not already diagonal. Analysis of the matrix $H_u$ can be done similarly to $H_x$. If $\gamma = 0$, then $H_u$ is strictly a diagonal matrix.

We remark that the $A$ matrix is a very sparse matrix. The matrix is tridiagonal with several zeros on the upper and lower off-diagonals. This makes the solution of a linear system $A^T H_x^{-1} A x = b$ very easy to compute because at tridiagonal matrix linear system can be solved in $O(n)$ operations, where $n$ is the number of variables, instead of the usual $O(n^3)$ operations that dense linear systems require [147]. Thus, the linear system $M M^T x = b$ is very easy to compute because it is made up of diagonal matrices $\tilde{H}_x$, $\tilde{H}_u$, and sparse tridiagonal matrix $A^T \tilde{H}_x^{-1} A$.

Distributed Implementation on a Ramp Metering System

This section highlights how the inexact interior point method can be computed in a distributed manner on a ramp metering system. In Chapter 5, the computational platforms are described to be in a star communication network where the central hub is the Traffic Management Center and each actuated ramp has its own controller with computation and communication capabilities.

The main distributed computation comes from the repeated matrix-vector multiplication of $A_{KKT} v$ (where $v$ are various vectors) in the iterative linear algebra, as discussed in Section 7.5. It is recommended to perform the other computations involved in the iterative linear algebra at the central hub since they are considerably fast and small operations such as vector-vector multiplication or even scalar addition, subtraction, multiplication, division. Similarly, most steps of the inexact interior point method (Algorithm 7) are simple enough
to be performed at the central hub. The backtrack search in steps 6 and 9 of Algorithm 7 can also be implemented in parallel in the same way that $A_{KKT}v$ can be implemented in parallel.

Due to the high frequency of communication required by the interior point algorithm, this distributed implementation is recommended for ramp metering systems with fast communication, such as fiber-optic communication. The algorithm speeds may suffer if the communication is slow (similar to results seen in Chapter 4). If this is the case, then the algorithm can still be implemented in a parallel manner on a centralized computing system with multiple cores. A controller box of the ramp metering system is able to hold multiple CPUs and can therefore do parallel computation. In this approach, feedback information of the system would be communicated to the box where the inexact interior point method is run. Then, the solutions would be communicated from that box to the appropriate controller boxes along the freeway. Otherwise, a multi-threaded computer at the Traffic Management Center can also compute the solution to the inexact interior point method in parallel. Similarly, this centralized computer would require communication of state feedback from the ramp metering controller boxes as well as communication back to the boxes of the control inputs.

7.7 Results

We examine several simulation examples in this results section. The first is a small toy example, similar to that of Chapter 6, to understand the difference between the hybrid MPC solution and the smoothed MPC solution. The second two examples are large-scale examples with different initial conditions. We show that the inexact interior point method can compute solutions even when the MIQP cannot be computed. The last example is a simulation of a full-scale freeway. In all examples, the model sampling time is 3 seconds. The first example is a toy example since it is only 2 links, but the other examples use the asymmetric cell transmission model of the 210 East freeway in Los Angeles, California. The first example has an actuated on-ramp at every segment, but the 210 East freeway has an actuated on-ramp only at some segments of the freeway. If segment $i$ does not have an actuated on-ramp, then it is assumed to have no on-ramp and thus $r_i(k) = d_i(k) = 0 \forall k$.

The inexact interior point method is coded in Matlab. The numerical linear algebra uses the Matlab implementation of SYMMLQ. Thus, the implementation is not parallel, but a discussion of how to parallelize the method is included in the conclusions. These results are run on a PC desktop with a 3.6 GHz Intel Core i7 processor running Windows 10.

The small scale problem is a 2 link example. The prediction horizon is 20. The hybrid MPC solution is coded using the MPT toolbox [19] which uses Gurobi [148] to solve the MIQP. The Hybrid MPC solution is provided in Figure 7.1 whereas the inexact interior point method solution for the same problem is shown in Figure 7.2.

The two simulation results are qualitatively not very different. The inexact interior point trajectory looks a somewhat smoothed version of the hybrid MPC solution. This suggests
that the conjugate smoothing presented in Section 7.6 is a fairly good approximation and using this approximation in the controller formulation is not overall detrimental to the quality of the solution.

Next, we present results from a large-scale traffic example. In this example we simulate a freeway with 20 segments and a prediction horizon of 25. We present the results of the inexact interior point method in Figure 7.3. We remark that the controller is able to steer the system to its equilibrium point by the end of the solution. The computation itself is about 400
seconds per iteration. However, the code has not been optimized (it contains cell structures and for loops which are known to be slow relative to other data structures and vectorized code [149]). Additionally, the code has not been parallelized. A parallel implementation of this algorithm for a machine learning problem showed that the parallelization resulted in a speed up of up to 100x relative to a centralized implementation [150].

For the large-scale example, we also remark on the performance of the preconditioner with respect to the iterative linear algebra iterations. With the preconditioner design in section 7.4, the number of iterations SYMMLQ used was usually about 18 iterations. Without the preconditioner, the number of iterations used by SYMMLQ varied widely from about 6-100 iterations. The consistency of iteration numbers with the preconditioner is nice for having repeatable algorithm performance. However, this result suggests that it may be a good idea to sometimes avoid using the preconditioner.

We next present another 20 segment simulation result with a different initial condition
but otherwise the same problem parameters. In particular, we present the solution provided by the relaxed linear program described in Section 5.4. This is compared with the inexact interior point solution in Figure 7.4a–7.4b. We report that the total travel time of the LP is 47266 and that of the smoothed hybrid MPC is 46750 with the LP travel time being 1% better than that of the smoothed hybrid MPC controller. Interestingly, the smoothed hybrid MPC again has no substantial performance loss relative to the relaxed LP.

Note that the examples with 20 segments were not simulated using the hybrid MPC controller proposed in Chapter 6. Such an example would have a piecewise affine model consisting of \(2 \times 3^{19}\) regions which is \(O(1e9)\) regions. To convert this to a MIQP, the piecewise affine model can be transformed into a mixed logical dynamical system using the big-M method [7]. Such a transformation would require a binary variable for each row of the linear inequality defining each region and two added mixed integer inequalities. In addition, this would add four linear inequalities for each row of the equality constrained dynamics. For example with 20 segments and 4 actuated on-ramps, this would require about 40 binary variables per region with 80 mixed-integer inequalities, and \(4 \times 24 = 96\) inequalities per region to replace the dynamics. These numbers are multiplied by the number of regions as well as the prediction horizon, resulting in \(O(1e12)\) binary variables and \(O(1e13)\) mixed integer constraints (this does not include the linear constraints on continuous variables). Clearly, this is an extremely large mixed integer quadratic program, which is not solvable even on a supercomputer. Some of the largest programs that are being studied by mixed integer programming researchers max out at 10s of millions of binary variables and are not yet feasible to solve [151].
We also note that a centralized solution of the 20-link problem using Julia JuMP [152] combined with the interior point solver Ipopt [127] was 1000 times faster than a solution using Ipopt in Matlab. The optimization modeling tool JuMP is very well developed for centralized optimization solutions. In fact, we were able to run a simulation of the smoothed ramp metering controller for the full-scale freeway using JuMP. This simulation consists of all 134 segments of the freeway 210 East. The prediction horizon was 25 steps. The terminal set $X_f$ was removed from this MPC formulation. Each MPC iteration required 2-16 seconds to compute an optimal solution on a laptop with a 2.2 Ghz Intel Core i7 processor running Mac OS X El Capitan. This is very impressive performance and certainly encourages the use of Julia for improving the code implementation of the inexact interior point algorithm. This is especially useful since the Julia language can easily interface with state-of-the-art parallel linear algebra packages such as PETSC [145] or Pardiso [146].

Figure 7.4: Trajectory using JuMP with Ipopt

7.8 Conclusions and Future Work

In conclusion, we have presented an inexact interior point method which can be computed in parallel by its use of iterative numerical linear algebra methods. The parallel implementation is flexible in how it can be parallelized – the splitting of the algorithm can be done spatially, temporally, or optimized for the parallel hardware. An interesting study that should be done
in the future is a study of these splitting options on embedded hardware that is typically found in large-scale control systems.

The method was applied to the ramp metering problem. The ramp metering problem has been proposed as a hybrid model predictive control problem which is NP hard to compute; a conjugate smoothing method was applied to transform the problem into a smooth nonlinear optimization problem. Interior point methods were a good fit because they are known to exhibit fast convergence in practice [135], especially for large-scale problems with a lot of sparsity which is generally present large-scale MPC problems.

The code implementing the inexact interior point algorithm is written in Matlab and is not optimized or parallelized. We suggest that a future implementation of the algorithm be implemented in a language such as Julia [153] or C. These languages are particularly useful because they can work with state-of-the-art parallel numerical linear algebra packages such as PETSC [145] or Pardiso [146]. Additionally, an inexact newton method has been proposed in [35]. This method uses different line-search methods and uses more general preconditioners than the preconditioner tailored for optimal control that we have used here. It would be very interesting to analyze and compare the algorithm in this chapter with the algorithm presented in [35].
Bibliography


