Optimal Control of Piecewise Affine Systems
– a Multi-parametric Approach –

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH
for the degree of
Doctor of Sciences

presented by
MATO BAOTIĆ

mr. sc., dipl. ing., University of Zagreb, Croatia
born 23.12.1973
citizen of Croatia

accepted on the recommendation of
Prof. Dr. Manfred Morari, examiner
Prof. Dr. Diethard Klatte, co-examiner
Prof. Dr. Nedjeljko Perić, co-examiner

2005
Optimal Control of Piecewise Affine Systems
– a Multi-parametric Approach –
To my parents.
Every burden is easier to carry if it can be shared (or at least discussed) with someone. Every joy is multiplied by the number of persons participating in it. I was happy enough to make my thesis at such a splendid place as Automatic Control Lab in Zürich. Yes, the weather in Zürich could have been better in those 4 years, but people could hardly be any better than the ones at IfA. Professors, secretaries, and, by my account, the best group of researchers in the world made every working and non-working hour thoroughly enjoyable.

First I would like to thank to Francesco Borrelli and Tobias Geyer. Without Francesco I don’t think I would have started anything of my research. His patience and endurance in explaining all the research stuff as well as some of the more obscure things (not least one of which is the position of Bellevue!) were most helpful. Tobias and I embarked on the quest of getting our PhD’s almost simultaneously and, by a stroke of good fortune, we’ve got it on the same day. The world is a magnificent place and to be able to discuss any given topic with him was a real pleasure.

My roommates Valentina Sartori and Michal Kvasnica deserve a special thanks. Our room rules! Among other things, Michal’s dedication to the MPT beyond my wildest dreams made it what it is – the best hybrid systems toolbox in the world, while Vale never gave up on me learning Italian by casual listening. Mille baci.

If Francesco was responsible for setting me up on the path of obtaining a PhD, then Frank J. Christophersen was responsible for getting me there. Without him I would have never finished many of the papers that are an essential part of this thesis. Yet the most amazing aspect was not related to work, nor to Frank’s vast knowledge of \LaTeX{} tricks, but to the fact that we share a similar taste in so many things in life. Thanks for existing my friend.

It was great to be taught the tricks of the trade by the likes of Alberto Bemporad, Giancarlo Ferrari-Trecate and Pablo Parrilo.

But the master of them all was by far my supervisor Professor Manfred Morari. Having a supervisor with such a vast scientific knowledge, clarity in explaining the ideas and wisdom in guiding PhD students through the valleys of ignorance and despair to the holy grail of knowledge was, for the lack of a better word, amazing. He was sending us around the globe and bringing the world to IfA for the same reason – so that we always have a full access to the vital research information while staying in touch with the reality. Overall, I think that the proverbial IfA saying “it is always good to talk to MM” says it all. From the bottom of my heart I thank him for the privilege of adding one more leaf (for the time being at least) to that “tree” of his PhD students.

Thanks to Fabio D. Torrisi for showing me that the clear thinking is essential for getting the job done and to Pascal Grieder for bringing a new vigor in our Lab and spurring a plethora of research activity. Thanks to Alice, Martine, Danielle and Esther for helping
me find the path through the maze of the ETH administration. Andrea B., Andrea G.,
Domenico, Dominik, Eleonora B., Eleonora Z., Gabi, George, Gültekin, Helfried, Ioannis,
Jan, Johan, Kristian, Marc, Miloš, Rainer, Raphael, Sašo, Urban - you make IfA the greatest
working environment in the world. One has to experience it to believe it.

I would also like to thank Prof. Nedjeljko Perić, Prof. Ivan Petrović, Miroslav Barić and
Mario Vašak from University of Zagreb for their encouragement and help in all the phases of
my PhD. Mario was instrumental in getting the electronic throttle to behave as the control
theory said it should.

Last but not least, this work would never have been possible if I was not set to life by my
parents Pavo and Manda. They and my brother Marinko have thought me by example the
most valuable lesson: with a little bit of patience and willingness to work one can indeed
move mountains with his own hands. Hvala vam za sve!

Mato Baotić
Zagreb, July 2005
Abstract

This thesis addresses the problem of constrained optimal control of discrete-time linear hybrid systems.

In their most general form hybrid systems are characterized by the interaction of continuous-valued components (governed by differential or difference equations) and logic rules. Among the variety of equivalent descriptions of discrete-time hybrid systems reported in the literature we focus on the class of constrained piecewise affine (PWA) systems. Discrete-time PWA models can describe a large number of processes. Moreover, they can approximate nonlinear discrete-time dynamics via multiple linearizations at different operating points.

Even though PWA systems are a special class of nonlinear systems most of the nonlinear system and control theory does not apply because it requires certain smoothness assumptions. For the same reason we also cannot simply use linear control theory in some approximate manner to design controllers for PWA systems. In the past most tools for the analysis and control of hybrid systems were ad hoc supported by extensive simulation.

The aim of this thesis is to further advance systematic procedures and develop algorithmic implementations that give the exact solution to the optimal control problems. A recurring theme in the thesis is the construction of efficient algorithms for solving various instances of optimal control problems. An instrumental tool in the development of such algorithms is the concept of multi-parametric programming, where a quadratic (or linear) optimization problem is solved off-line for a range of parameters.

Specifically, an efficient implementation of a general multi-parametric quadratic program is described together with the in-depth analysis of the properties of the solution. The optimal control problem of a constrained linear discrete-time system can now be formulated as a multi-parametric quadratic program by treating the state vector as a parameter. The optimal solution is a piecewise affine state-feedback control law that is defined over a polyhedral partition of the feasible state-space. This allows users to carry out most of the time-consuming/complex computation off-line, while on-line implementation (control action computation) reduces to a simple set-membership test.

By exploiting the properties of the value function and the optimal control law, new algorithms are developed that avoid storing the polyhedral regions. The new algorithms significantly reduce the on-line storage demands and computational complexity during evaluation of the PWA feedback control law.

Next, the finite time optimal control problem for constrained discrete-time linear hybrid systems based on quadratic or linear performance criteria is tackled. Basic theoretical results on the structure of the optimal state-feedback solution and of the value function are given. An algorithm for construction of the solution – a piecewise affine state-feedback control law defined over possibly non-convex regions – combines multi-parametric programming,
dynamic programming and basic polyhedral manipulation.

Similar ideas are extended to the infinite time optimal control problems with linear performance index. A novel algorithm solves the Hamilton-Jacobi-Bellman equation by using the multi-parametric linear programming solver in a dynamic programming fashion. The resulting solution when applied in a receding horizon fashion guarantees stability of the closed-loop system. The important issue of stability guarantees is also addressed with the introduction of sub-optimal control strategies that generate solutions of lower complexity.

Most of the developed algorithms were tested in two real-life automotive applications: electronic throttle control and adaptive cruise control. The electronic throttle is used in automotive applications to control the inflow of air to the vehicle engine by positioning a throttle plate. The nonlinearities present in the throttle body make the control of the plate position a challenging task. The electronic throttle is firstly modelled as a PWA system and then the control strategies developed in this thesis are applied to it. In a multi-object adaptive cruise control problem the optimal acceleration of the car is to be found respecting traffic rules, safety distances and driver intentions. The hybrid nature of the problem arises from the multiple objectives that introduce integer variables.

Finally, the MPT toolbox is presented. The MPT toolbox for MATLAB contains all of the algorithms presented in this thesis as well as a wide range of additional algorithms and tools developed by the academic community.
Diese Dissertation behandelt die Thematik optimaler Regelungsprobleme mit Beschränkungen für die Klasse der zeitdiskreten, linearen Hybriden Systeme.


Durch die Nutzung der Eigenschaften des optimalen Kostenfunktionsals und des optimalen Regelgesetzes konnten neue Algorithmen entwickelt werden, die die Speicherung der polyhe-
Zusammenfassung

Durch Regionalen und linearen Gütekriterien, für beschränkte, zeitdiskrete, lineare Hybride Systeme behandelt; es werden grundlegende theoretische Ergebnisse bezüglich der Form und Struktur des optimalen Regelgesetzes und des optimalen Kostenfunktional aufgezeigt. Der Algorithmus zur Bestimmung der optimalen Lösung, die die Form eines abschnittsweise affinen Regelgesetzes über (möglicherweise) nichtkonvexe Regionen besitzt, basiert auf einer Kombination von multi-parametrischer Programmierung, Dynamischer Programmierung und elementarer Polyhedramanipulation.


# Contents

Acknowledgement i
Abstract iii
Zusammenfassung v
1 Introduction 1

## I MATHEMATICAL PROGRAMMING 5

2 Background 6
  2.1 Basic Terminology and Definitions 6
  2.2 Convex Optimization 10

3 Polytopes 13
  3.1 Definitions 13
  3.2 Basic Operations on Polytopes 16
    3.2.1 Minimal Representation 16
    3.2.2 Chebychev Ball 17
    3.2.3 Projection 17
    3.2.4 Set-Difference 18
    3.2.5 Convex Hull 19
    3.2.6 Envelope 19
    3.2.7 Vertex Enumeration 20
    3.2.8 Pontryagin Difference 20
    3.2.9 Minkowski Sum 21
  3.3 Polycover and Regiondiff 22
    3.3.1 Polycover: MILP formulation 23
    3.3.2 Polycover: Branch & Bound algorithm 25

4 Multi-Parametric Programming 29
  4.1 Basics of Multiparametric programming 29
  4.2 Multi-parametric Quadratic Programming 31
    4.2.1 Translation into the standard mp-QP form 32
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.2</td>
<td>Solution of the mp-QP</td>
<td>32</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Direct exploration of the parameter space</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Multi-parametric Linear Programming</td>
<td>42</td>
</tr>
</tbody>
</table>

### II OPTIMAL CONTROL

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Constrained Finite Time Optimal Control of Linear Systems</td>
<td>44</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>44</td>
</tr>
<tr>
<td>5.2</td>
<td>CFTOC, RHC and their State-feedback PWA Solution</td>
<td>45</td>
</tr>
<tr>
<td>5.2.1</td>
<td>CFTOC problem formulation</td>
<td>45</td>
</tr>
<tr>
<td>5.2.2</td>
<td>RHC strategy</td>
<td>46</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Solution of CFTOC, linear cost Case</td>
<td>46</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Solution of CFTOC, quadratic cost Case</td>
<td>48</td>
</tr>
<tr>
<td>5.3</td>
<td>Efficient on-line Algorithms</td>
<td>49</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Efficient Implementation, linear cost Case</td>
<td>50</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Efficient Implementation, quadratic cost Case</td>
<td>51</td>
</tr>
<tr>
<td>5.4</td>
<td>Example</td>
<td>59</td>
</tr>
<tr>
<td>5.4.1</td>
<td>CFTOC, linear cost Case</td>
<td>60</td>
</tr>
<tr>
<td>5.4.2</td>
<td>CFTOC, quadratic cost Case</td>
<td>60</td>
</tr>
<tr>
<td>5.5</td>
<td>Conclusion</td>
<td>61</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Constrained Finite Time Optimal Control of PWA Systems</td>
<td>62</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>62</td>
</tr>
<tr>
<td>6.2</td>
<td>Piecewise Affine Systems</td>
<td>63</td>
</tr>
<tr>
<td>6.3</td>
<td>Problem Formulation</td>
<td>64</td>
</tr>
<tr>
<td>6.4</td>
<td>Solution Properties</td>
<td>65</td>
</tr>
<tr>
<td>6.5</td>
<td>Computation of the Optimal Control Law via Dynamic Programming</td>
<td>71</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Preliminaries and Basic Steps</td>
<td>72</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Multiparametric Programming with Multiple Quadratic Functions</td>
<td>74</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Algorithmic Solution of the HJB Equations</td>
<td>77</td>
</tr>
<tr>
<td>6.6</td>
<td>Discontinuous PWA systems</td>
<td>80</td>
</tr>
<tr>
<td>6.7</td>
<td>Conclusion</td>
<td>81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>CFTOC and CITOC of PWA Systems with a Linear Performance Index</td>
<td>82</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>82</td>
</tr>
<tr>
<td>7.2</td>
<td>Linear Hybrid Systems</td>
<td>83</td>
</tr>
<tr>
<td>7.3</td>
<td>Constrained Finite Time Optimal Control</td>
<td>85</td>
</tr>
<tr>
<td>7.3.1</td>
<td>The CFTOC Solution via mp-MILP</td>
<td>86</td>
</tr>
<tr>
<td>7.3.2</td>
<td>The CFTOC Solution via Dynamic Programming</td>
<td>87</td>
</tr>
<tr>
<td>7.3.3</td>
<td>An Efficient Algorithm for the CFTOC Solution</td>
<td>88</td>
</tr>
<tr>
<td>7.3.4</td>
<td>Comments on the Dynamic Programming Approach for the CFTOC Problem</td>
<td>89</td>
</tr>
<tr>
<td>7.3.5</td>
<td>Receding Horizon Control</td>
<td>90</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7.3.6</td>
<td>Example: Constrained PWA System</td>
<td>90</td>
</tr>
<tr>
<td>7.4</td>
<td>Constrained Infinite Time Optimal Control</td>
<td>93</td>
</tr>
<tr>
<td>7.4.1</td>
<td>The CITOC Solution via Dynamic Programming</td>
<td>96</td>
</tr>
<tr>
<td>7.4.2</td>
<td>An Efficient Algorithm for the CITOC Solution</td>
<td>106</td>
</tr>
<tr>
<td>7.4.3</td>
<td>Alternative Choices of the initial Value Function $J_0$</td>
<td>109</td>
</tr>
<tr>
<td>7.4.4</td>
<td>Examples</td>
<td>112</td>
</tr>
<tr>
<td>7.5</td>
<td>Software Implementation</td>
<td>114</td>
</tr>
<tr>
<td>8</td>
<td>Stabilizing Low Complexity Control of PWA Systems</td>
<td>115</td>
</tr>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>115</td>
</tr>
<tr>
<td>8.2</td>
<td>Problem Description and Properties</td>
<td>116</td>
</tr>
<tr>
<td>8.3</td>
<td>Computation of Stabilizing Controllers for Piecewise Affine Systems</td>
<td>118</td>
</tr>
<tr>
<td>8.4</td>
<td>Computation of Low Complexity Controllers for Piecewise Affine Systems</td>
<td>120</td>
</tr>
<tr>
<td>8.4.1</td>
<td>Computation of a Minimum Time Controller</td>
<td>121</td>
</tr>
<tr>
<td>8.4.2</td>
<td>Controller with Reduced Number of Switches</td>
<td>124</td>
</tr>
<tr>
<td>8.4.3</td>
<td>One-Step Controller</td>
<td>126</td>
</tr>
<tr>
<td>8.5</td>
<td>Numerical Examples</td>
<td>130</td>
</tr>
<tr>
<td>8.5.1</td>
<td>Controller Computation</td>
<td>130</td>
</tr>
<tr>
<td>8.5.2</td>
<td>Stability Analysis</td>
<td>133</td>
</tr>
<tr>
<td>8.6</td>
<td>Conclusion</td>
<td>134</td>
</tr>
<tr>
<td>9</td>
<td>Stability Analysis of PWA Systems with a Linear Performance Index</td>
<td>136</td>
</tr>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>136</td>
</tr>
<tr>
<td>9.2</td>
<td>Constrained Finite Time Optimal Control</td>
<td>137</td>
</tr>
<tr>
<td>9.3</td>
<td>Closed-Loop Stability &amp; Feasibility</td>
<td>139</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Computation of the Maximum Positive Invariant Set</td>
<td>141</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Computation of the Lyapunov Stability Region</td>
<td>145</td>
</tr>
<tr>
<td>9.4</td>
<td>Examples</td>
<td>149</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Linear System: Constrained Double Integrator</td>
<td>149</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Constrained PWA System</td>
<td>149</td>
</tr>
<tr>
<td>9.4.3</td>
<td>Constrained PWA System</td>
<td>153</td>
</tr>
<tr>
<td>9.4.4</td>
<td>3-dimensional Constrained PWA System</td>
<td>155</td>
</tr>
<tr>
<td>9.5</td>
<td>Conclusion</td>
<td>156</td>
</tr>
<tr>
<td>III</td>
<td>CASE STUDIES</td>
<td>157</td>
</tr>
<tr>
<td>10</td>
<td>Multi-object Adaptive Cruise Control</td>
<td>158</td>
</tr>
<tr>
<td>10.1</td>
<td>Introduction</td>
<td>158</td>
</tr>
<tr>
<td>10.2</td>
<td>Modelling and Estimation</td>
<td>159</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Sensors</td>
<td>159</td>
</tr>
<tr>
<td>10.2.2</td>
<td>Lane Recognition</td>
<td>160</td>
</tr>
<tr>
<td>10.2.3</td>
<td>Object Recognition</td>
<td>161</td>
</tr>
<tr>
<td>10.2.4</td>
<td>Control Model</td>
<td>162</td>
</tr>
</tbody>
</table>
10.3 Problem Formulation .................................. 164
10.4 PWA Model of the System ............................ 165
10.5 Optimal State Feedback Control Law for PWA Systems . 167
10.6 Simulation Results .................................. 167
10.7 Experimental Results ................................. 168
10.8 Conclusion ........................................ 170

11 CFTOC of an Electronic Throttle ........................ 173
11.1 Introduction ....................................... 173
11.2 Electronic Throttle Model ............................ 174
   11.2.1 Static Friction Model ........................... 176
   11.2.2 Limp-Home Nonlinearity ......................... 177
11.3 PWA Model of the Throttle .......................... 177
11.4 Constrained Optimal Control of the Throttle ............ 179
   11.4.1 Simulation Results ............................. 179
   11.4.2 Experimental Results ........................... 180
11.5 Conclusion ........................................ 180

12 Low Complexity Control of an Electronic Throttle .......... 183
12.1 Discrete-Time Piecewise Affine Model of an Electronic Throttle .... 183
   12.1.1 Electronic Throttle Model ....................... 183
   12.1.2 Dynamic Friction Model ......................... 184
   12.1.3 Discrete-Time PWA Model of an Electronic Throttle ... 186
12.2 Time-Optimal Electronic Throttle Reference Tracking ......... 186
   12.2.1 Extended System and Constraints ................ 186
   12.2.2 Off-Line Computation ........................... 187
   12.2.3 On-Line Computation ............................ 189
12.3 Experimental Results ................................ 189
12.4 Conclusion ........................................ 191

IV SOFTWARE ......................................... 193
13 The MPT Toolbox ...................................... 194
   13.1 Overview of The Toolbox ........................... 194
      13.1.1 Classes and Basic Polytope Manipulations .... 196
      13.1.2 Control Functions ............................. 199
      13.1.3 Analysis Functions ............................. 200
   13.2 Polytope Library ................................ 200
      13.2.1 Creating a polytope ............................ 201
      13.2.2 Accessing data stored in a polytope object .... 201
      13.2.3 P-collections ................................. 202
   13.3 MPT Functions and Structures ....................... 204
      13.3.1 System Structure sysStruct ..................... 205
Introduction

This thesis addresses the problem of constrained optimal control of discrete-time linear hybrid systems. We study the optimal control of linear and piecewise affine (PWA) systems with polyhedral constraints on states and inputs, consider linear and quadratic objective functions and derive solutions for finite and infinite horizon problem formulations.

Recent technological advances have caused a considerable interest in the study of dynamical processes of a mixed continuous and discrete nature, denoted as hybrid systems. In their most general form hybrid systems are characterized by the interaction of continuous-valued components (governed by differential or difference equations) and logic rules. Hybrid systems are very common in engineering and many systems encountered in everyday life can be effectively modelled as hybrid systems as well. A simple example of a hybrid system would be a car. The dynamics of the car switch when a gear shift occurs, either because the driver moves the stick shift (input event) or because the state variable “speed” exceeds a specified threshold (state event) in the case of an automatic transmission. Another special example of a hybrid system would be a linear system under feedback control with actuator constraints. When the actuator hits a constraint the dynamics change.

We focus on the class of discrete-time piecewise affine (PWA) systems that are defined by the partition of the extended state-input space into polyhedral regions and a set of different affine state update equations associated with those regions, cf. [Son81, HDB01]. Discrete-time PWA models can describe a large number of processes, such as discrete-time linear systems with static piecewise linearities, discrete-time linear systems with logic states and inputs or switching systems where the dynamic behavior is described by a finite number of discrete-time linear models, together with a set of logic rules for switching among these models. Moreover, PWA systems can approximate nonlinear discrete-time dynamics via multiple linearizations at different operating points.

Even though hybrid systems are a special class of nonlinear systems most of the nonlinear system and control theory does not apply because it requires certain smoothness assumptions. For the same reason we also cannot simply use linear control theory in some approximate manner to design controllers for PWA systems. Different methods for the analysis and design of controllers for hybrid systems have emerged over the last few years [Son81, LTS99, Bla99]. Among them, the class of optimal controllers is one of the most studied. The approaches differ greatly in the hybrid models adopted, in the formulation of the optimal control problem and in the method used to solve it.

The thesis is subdivided into four parts each of which addresses specific topic. Since most of the chapters in the thesis are based on published, accepted or submitted papers some
repetition of the key theorems was unavoidable.

Part I of the thesis summarizes some basic convex optimization concepts. Two concepts instrumental to the work carried in this thesis – that of polytopic sets and multi-parametric programming – are explained in a greater detail. As is shown in the latter chapters algorithms that solve linear or quadratic program for a range of parameters (i.e. efficient multi-parametric programming solvers) are used in deriving the closed form solutions to the optimal control problems of linear and PWA systems. The main building block in such algorithms are operations over polytopes and polytopic objects. Therefore ensuring efficient operations with such objects is crucial if we are to derive usable control algorithms. The results in this part are based on [Bao02, KGBM04, BT03, BBBM01a].

In Part II we describe how to solve constrained finite time optimal control problems for linear and PWA systems with quadratic and linear performance indices. Theoretical results on the structure of the optimal state-feedback solution and of the value function are given. It is shown that the properties of the value function and control law can be used to simplify the solution of the problem in the case of linear systems. The algorithms for the construction of the solution – a piecewise affine state-feedback control law defined over possibly non-convex regions – are given. In case of linear systems algorithms one needs to solve multi-parametric linear or quadratic program, while for PWA systems the algorithms combine multi-parametric programming, dynamic programming and polyhedral manipulation. Similar ideas are extended to the infinite time optimal control problems with linear performance index. A novel algorithm solves the Hamilton-Jacobi-Bellman equation by using the multi-parametric linear programming solver in a dynamic programming fashion. The resulting solution when applied in a receding horizon fashion guarantees stability of the closed-loop system. The important issue of stability guarantees is also addressed with the introduction of sub-optimal control strategies that generate solutions of lower complexity. Furthermore we outline procedure for stability analysis of (autonomous) PWA systems. The content in this part is based on [BBBM01a, BBBM05a, BBBM03a, BBBM05b, MBB03, BCM03a, BCM03b, GKB04, GKB05, CBM04].

Part III describes two real-life problems on which some of the developed algorithms were tested. This “reality check” was used both as a guidance in the research and as a motivation for improvement of developed algorithms. Both applications are from the area of automotive control: adaptive cruise control and electronic throttle control. In a multi-object adaptive cruise control problem the optimal acceleration of the car is to be found respecting traffic rules, safety distances and driver intentions. The hybrid nature of the problem arises from the multiple objectives that introduce integer variables. The electronic throttle is used in automotive applications to control the inflow of air to the vehicle engine by positioning a throttle plate. The nonlinearities present in the throttle body make the control of the plate position a challenging task. The electronic throttle is firstly modelled as a PWA system and then the control strategies developed in this thesis are applied to it. The content in this part is based on [MBM03, BVMP03, VBPP04, VBPP05].
In Part IV the Multi-Parametric Toolbox is presented. The MPT toolbox for MATLAB contains all of the algorithms presented in this thesis as well as a wide range of additional algorithms and tools developed by the academic community. This part of the thesis introduces the reader to MPT, describes the software framework and provides some examples. The content in this part is based on [KGBM04, KGBC04].

Contribution

Almost all of the results reported in this thesis have been obtained in close collaboration with various colleagues. This fact is also reflected in the cited references. Therefore, the contributions of my thesis should be regarded in this collaborative context. Note also that not all of the results of my research were included in the thesis. A full list of my publications can be seen in Appendix B.

The main contribution of the thesis is the development of systematic procedures and algorithmic implementations that give the explicit solution to the constrained optimal control problems for discrete-time linear and piecewise affine systems.

We address the so-called polycover problem where one needs to check if a polytope is covered by the union of other polytopes. Both polycover and related problem of computing a set difference frequently occur in the context of the optimal control and it is imperative to have efficient algorithms for them. We provide such algorithms in Chapter 3 [BT03].

We also address deficiencies of the initially proposed multi-parametric quadratic programming (mp-QP) solver from [BMDP02]. The algorithm we give in Chapter 4 [Bao02] is superior to [BMDP02] since it explores the feasible state space in a natural way – by crossing the border between two neighboring critical regions during the construction of the solution. The same approach can be used for multi-parametric linear programming (mp-LP) solvers [KGBC04].

In Chapter 5 we give a new insight into the structure of the optimal state-feedback solution and of the value function for the case of linear systems. We show how the properties of the value function and control law can be used to simplify the solution of the problem for linear systems [BBBM01a, BBBM05a].

We derive a new algorithm for the construction of the solution for PWA systems both for a linear [BCM03a, BCM03b, BCM04] and quadratic performance indices [BBBM03a, BBBM05b] in Chapters 6 and 7. The algorithms construct the solution – a piecewise affine state-feedback control law defined over possibly non-convex regions – by combining multi-parametric programming, dynamic programming strategy and polyhedral manipulation. Similar ideas are extended to the infinite time optimal control problems with linear performance index [BCM03b, BCM04]. A novel algorithm solves the Hamilton-Jacobi-Bellman equation by using the multi-parametric linear programming solver in a dynamic programming fashion. It is shown that the resulting solution, when applied in a receding horizon fashion, guarantees stability of the closed-loop system.

In Chapter 8 [GKBM04, GKB05] we extend the ideas from [GM03, GM03] where linear systems are considered to the case of PWA systems. We show that problems with simpler performance objectives can produce simpler control laws, while maintaining stability properties
1 Introduction

for the overall closed-loop system. Specifically, “minimal time” and “minimum-switching” control objectives were investigated.

The important issue of stability guarantees is also addressed in Chapter 9, where a reachability based procedure for stability analysis of (autonomous) PWA systems is outlined [CBM04].

Part III shows application of developed algorithms in the real-life automotive control.

In Chapter 10 Multi-Object Adaptive Cruise Control is discussed. In a multi-object adaptive cruise control problem the optimal acceleration of the driver’s car is to be found respecting traffic rules, safety distances and driver intentions. The hybrid nature of the problem arises from the multiple objectives which include switches. We have shown [MBM03] that the Multi-Object Adaptive Cruise Control problem can be solved via hybrid system theory. The objective function is modelled as a quadratic cost function for the discrete-time PWA system. The optimal state-feedback control law is found by solving the underlying constrained finite time optimal control problem via dynamic programming. Experimental results are reported for the car-following scenario.

An optimal control of an electronic throttle is addressed in Chapters 11 and 12. An electronic throttle is, essentially, a DC servo system used to control the inflow of air to the vehicle engine by positioning the throttle plate. Two nonlinearities in the throttle body make the control of the plate position a challenging task: i) a strong friction in the gearbox transmission mechanism, ii) the so-called Limp-Home (LH) position nonlinearity generated by an embedded mechanical safety feature that guarantees a specific level of air inflow even in the case of total power failure. In Chapter 11 we derive a state-feedback optimal control law by modeling the electronic throttle as a PWA system and applying the strategy from Chapter 6. Experimental results indicate that constrained finite time optimal control of small/medium sized PWA systems with fast sampling times can be successfully implemented [MBM03,BVMP03]. In Chapter 12 we further simplify the look-up table like control law for the throttle application by using the strategy from Chapter 8. In comparison, experimental results show that our approach outperforms a tuned PID controller that includes feedforward compensation of nonlinearities. We achieve more than two times faster transients, while preserving the quality regarding absence of overshoot and static accuracy within the measurement resolution [VBPP04, VBPP05].

The last key contribution during my Ph.D. studies is the co-development of the Multi-Parametric Toolbox which is covered in Part IV [KGBM04,KGBC04]. By developing such an extensive software tool, we have made the world of multi-parametric control accessible to a much larger audience than was previously the case. Hopefully, the easy access to the latest advances in multi-parametric theory will also lead to more practical applications in the future.
Part I

MATHEMATICAL PROGRAMMING
2

Background

Throughout this thesis we extensively use many concepts from the branch of mathematics called mathematical programming. In a mathematical programming, one seeks to minimize or maximize a real function of real or integer variables, subject to constraints on the variables. The term mathematical programming refers to the study of these problems: their mathematical properties, the development and implementation of algorithms to solve these problems, and the application of these algorithms to real world problems.

2.1 Basic Terminology and Definitions

Most of the mathematical concepts we use are well known and respective definitions can be found in the existing textbooks on applied mathematics (e.g., optimization), mathematical analysis and topology. However, for sake of completeness and to avoid misinterpretation of their meaning in the thesis, we give some of the definitions here. For a detailed reference, we refer the reader to the excellent books: Convex Analysis by R. T. Rockafellar [Roc97], and Convex Optimization by S. Boyd and L. Vandenberghe [BV04]. For a more “interactive” introduction we recommend two very helpful web based collections: Mathematical Programming Glossary created by H. Greenberg [Gre04] and MathWorld – a mathematics encyclopedia – created by E. W. Weisstein [Wei].

Definition 2.1 (Affine Set). A set $S \subseteq \mathbb{R}^n$ is affine if the line through any two distinct points in $S$ lies in $S$, i.e.,

$$x, y \in S, \quad \theta \in \mathbb{R} \quad \Rightarrow \quad \theta x + (1 - \theta) y \in S.$$  

Definition 2.2 (Affine Hull). Affine hull of a set $S \subseteq \mathbb{R}^n$ is the smallest affine set that contains $S$, i.e.,

$$\text{aff}(S) := \left\{ \sum_{i=1}^{k} \theta_i x_i \mid x_i \in S, \; \theta_i \in \mathbb{R}, \; i = 1, \ldots, k, \; \sum_{i=1}^{k} \theta_i = 1, \; k \in \mathbb{N} \right\}. \quad (2.1)$$
Note that an affine set in $\mathbb{R}^n$ containing the origin is actually a subspace of $\mathbb{R}^n$.

**Definition 2.3** (Affine Dimension). An *affine dimension* of a set $S \subseteq \mathbb{R}^n$ is the dimension of its affine hull, i.e.,

$$\text{affdim}(S) := \dim(\text{aff}(S)).$$  \hfill (2.2)

**Definition 2.4** (Convex Set). A set $S \subseteq \mathbb{R}^n$ is *convex* if the line segment connecting any pair of points of $S$ lies entirely in $S$, i.e.,

$$x, y \in S, \quad 0 \leq \theta \leq 1 \quad \Rightarrow \quad \theta x + (1 - \theta)y \in S.$$  \hfill □

**Definition 2.5** (Convex Hull). Convex hull of a set $S \subseteq \mathbb{R}^n$ is the smallest convex set that contains $S$, i.e.,

$$\text{co}(S) := \left\{ \sum_{i=1}^{k} \theta_i x_i \mid x_i \in S, \ \theta_i \geq 0, \ i = 1, \ldots, k, \ \sum_{i=1}^{k} \theta_i = 1, \ k \in \mathbb{N} \right\}. \hfill (2.3)$$  \hfill □

**Definition 2.6** (Convex Cone). A set $S \subseteq \mathbb{R}^n$ is a *convex cone* if

$$x, y \in S, \quad \theta_1, \theta_2 \geq 0 \quad \Rightarrow \quad \theta_1 x + \theta_2 y \in S.$$  \hfill □

For a real vector space $\mathbb{R}^n$ equipped with the Euclidean norm we define a particular useful convex set called *(Euclidean) ball* as follows.

**Definition 2.7** (Ball). A *ball* in $\mathbb{R}^n$ is a set of the form

$$\mathcal{B}(x, r) := \{ y \in \mathbb{R}^n \mid \|y - x\|_2 < r \},$$

where $x \in \mathbb{R}^n$ is a center and $r > 0$ is a radius of the ball. \hfill □

Note that for $x \in \mathbb{R}^n$, since $r > 0$, $\mathcal{B}(x, r)$ is a non-empty set in $\mathbb{R}^n$, i.e. the ball is a *full-dimensional* set. We use this property of a ball in $\mathbb{R}^n$ to check if other sets are full-dimensional. Furthermore, by the strict inequality in (2.4), the ball is defined as an *open set*, i.e. the boundary of the ball is not part of the ball. This allows us to describe an important concept of the *neighborhood* of a given point $x$ with a ball centered at $x$.\(^1\)

**Definition 2.8** (Full-dimensional Set). A set $S \subseteq \mathbb{R}^n$ is *full-dimensional* if

$$\exists \epsilon > 0, x \in \mathbb{R}^n : \mathcal{B}(x, \epsilon) \subseteq S.$$ \hfill (2.5)  \hfill □

\(^1\)Strictly speaking neighborhood of $x$ is any set that contains an open set around $x$, which makes the neighborhood of $x$ a broader concept than the ball centered at $x$.\[^{1}\]
Definition 2.9 (Interior). The interior of a set \( S \subseteq \mathbb{R}^n \) is the set of all points interior to \( S \), i.e.,
\[
\text{int}(S) := \{ x \in S \mid \exists \epsilon > 0 : B(x, \epsilon) \subseteq S \}.
\]  

(2.6)

According to the above definition, any non full-dimensional set in \( \mathbb{R}^n \) has an empty interior. For such sets it is useful to define a relative interior.

Definition 2.10 (Relative Interior). The relative interior of a set \( S \subseteq \mathbb{R}^n \) is the set of all points interior to \( \text{aff}(S) \), i.e.,
\[
\text{relint}(S) := \{ x \in S \mid \exists \epsilon > 0 : B(x, \epsilon) \cap \text{aff}(S) \subseteq S \}.
\]  

(2.7)

Definition 2.11 (Open Set). A set \( S \subseteq \mathbb{R}^n \) is open if it is equal to its interior, i.e.,
\[
S = \text{int}(S).
\]

\( \square \)

Definition 2.12 (Closed Set). A set \( S \subseteq \mathbb{R}^n \) is closed if every point outside \( S \) has a neighborhood disjoint from \( S \), i.e.,
\[
\forall x \notin S, \exists \epsilon > 0 : B(x, \epsilon) \cap S = \emptyset.
\]

\( \square \)

Note that the definitions of open and closed sets are not mutually exclusive, nor are they the only possibility. Indeed some sets are neither closed nor open (e.g., \( \{ x \in \mathbb{R} \mid 0 < x \leq 1 \} \)), while others are both open and closed (e.g., empty set \( \emptyset \)). When in doubt if some set is closed, open, both, or neither, the best advice is to check it with both definitions.

Definition 2.13 (Closure). Closure of a set \( S \subseteq \mathbb{R}^n \) is defined as a unique smallest closed set \( \overline{S} \) containing \( S \), or, equivalently,
\[
\overline{S} := \{ x \in \mathbb{R}^n \mid \forall \epsilon > 0, B(x, \epsilon) \cap S \neq \emptyset \}.
\]

(2.8)

\( \square \)

Definition 2.14 (Boundary). The boundary of a set \( S \subseteq \mathbb{R}^n \) is defined as
\[
\partial S := \overline{S} \setminus \text{int}(S).
\]

(2.9)

\( \square \)

Definition 2.15 (Bounded Set). A set \( S \subseteq \mathbb{R}^n \) is bounded if it is contained inside some ball of finite radius, i.e.,
\[
\exists r < \infty, x \in \mathbb{R}^n : S \subseteq B(x, r).
\]

\( \square \)
**Definition 2.16 (Compact Set).** A set \( S \subseteq \mathbb{R}^n \) is **compact** if it is closed and bounded. □

By inspection, we see that for any set \( S \subseteq \mathbb{R}^n \) the following hold:

\[
\text{int}(S) \subseteq \text{relint}(S) \subseteq S \subseteq \text{co}(S) \subseteq \text{aff}(S), \quad (2.10)
\]

and

\[
S \subseteq \bar{S} \subseteq \text{aff}(S). \quad (2.11)
\]

**Example 2.1.** Consider a two-dimensional circle, \( S = \{ x \in \mathbb{R}^2 \mid x'x = 1 \} \). According to the previous definitions set \( S \) has the following properties: i) \( S \) is not affine, ii) \( \text{aff}(S) = \mathbb{R}^2 \), iii) \( \text{affdim}(S) = 2 \), iv) \( S \) is not convex, v) \( \text{co}(S) = \{ x \in \mathbb{R}^2 \mid x'x \leq 1 \} = \overline{B(O_2, 1)} \), vi) \( S \) is not full-dimensional, vii) \( \text{int}(S) = \emptyset \), viii) \( \text{relint}(S) = S \), ix) \( S \) is not open, ix) \( S \) is closed, x) \( \bar{S} = S \), xi) \( \partial S = S \), xii) \( S \) is bounded, xiii) \( S \) is compact. □

Example 2.1 illustrates that for non-convex sets the affine dimension and the “real” (i.e. topological) dimension are not necessarily equal. This mismatch, however, does not occur when dealing with convex sets [Roc97].

**Definition 2.17 (Affine Function).** A function \( f : \mathbb{R}^n \to \mathbb{R}^m \) is **affine** if it can be written in the form

\[
f(x) = Ax + b,
\]

where \( A \in \mathbb{R}^{m \times n} \), \( b \in \mathbb{R}^m \). □

**Definition 2.18 (Convex Function).** Let \( S \subseteq \mathbb{R}^n \) be a convex set. A function \( f : S \to \mathbb{R} \) is **convex** if

\[
x, y \in S, \quad 0 \leq \theta \leq 1 \quad \Rightarrow \quad f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).
\]

□

**Remark 2.1 (Properties of vector valued functions).** Note that the convexity property is defined for the scalar valued functions. However, very often we use vector valued functions (e.g. to simplify notation) and when we talk about convexity of such functions (or any other property that is defined for scalar functions), by convention, we consider it componentwise. □

**Lemma 2.1 (First order conditions).** Let \( S \subseteq \mathbb{R}^n \) be a convex set. A differentiable function \( f : S \to \mathbb{R} \) is convex if and only if

\[
f(y) \geq f(x) + (\nabla f(x))'(y - x), \quad \forall x, y \in S.
\]

□

**Lemma 2.2 (Second order conditions).** Let \( S \subseteq \mathbb{R}^n \) be a convex set. A twice differentiable function \( f : S \to \mathbb{R} \) is convex if and only if its Hessian is positive semidefinite, i.e.

\[
\nabla^2 f(x) \succeq 0, \quad \forall x \in S.
\]

□
Corollary 2.1 (Quadratic functions). Let $S \subseteq \mathbb{R}^n$ be a nonempty convex set. Consider the quadratic function $f : S \to \mathbb{R}$ (i.e., $\text{dom}(f) = S$), given by

$$f(x) = x'Px + 2q'x + r.$$ 

with $P \in S''$, $q \in \mathbb{R}^n$, and $r \in \mathbb{R}$. Then function $f$ is convex over its domain if and only if $P \succeq 0$. Furthermore, function $f$ is strictly convex if and only if $P \succ 0$.

Proof: See [BSS93, Theorem 3.3.7 and Theorem 3.3.8].

Definition 2.19 (Mathematical Program). [Gre04, BV04] A mathematical program is an optimization problem of the form

$$\inf_z f(z) \quad (\text{MP})$$

subj. to

$$\begin{cases}
g(z) \leq 0, \\
h(z) = 0, \\
z \in Z,
\end{cases}$$

where $z \in \mathbb{R}^{n_z}$ is the optimization variable, while the functions $f : \mathbb{R}^{n_z} \to \mathbb{R}$, $g : \mathbb{R}^{n_z} \to \mathbb{R}^{n_g}$, and $h : \mathbb{R}^{n_z} \to \mathbb{R}^{n_h}$, and the set $Z \subseteq \mathbb{R}^{n_z}$ are given problem parameters.

The function $f$ is called the cost function or objective function. The relations $g(z) \leq 0$, $h(z) = 0$, and $z \in Z$ are called (resp. inequality, equality and set) constraints. The set of all points for which real valued functions $f$, $g$ and $h$ are defined is called the domain of the problem (MP), i.e. domain of (MP) = $\text{dom}(f) \cap \text{dom}(g) \cap \text{dom}(h)$. A point $z$ in the domain is feasible if it satisfies all constraints. The set of all feasible points is called the feasible set. We say that the problem (MP) is feasible if there is at least one feasible point, and infeasible if there are no feasible points. A point $z^*$ is optimal if it is feasible and if $f(z^*) \leq f(z)$ for all feasible $z$. If there are no constraints ($n_g = 0$, $n_h = 0$ and $Z = \mathbb{R}^{n_z}$) we say that the problem (MP) is unconstrained. If there are feasible points $z_k$ with $f(z_k) \to -\infty$ as $k \to \infty$ we say that the problem (MP) is unbounded (bellow).

2.2 Convex Optimization

A well known fact is that a very general problem is harder to solve then the very specific problem (belonging to the same class, of course). One might call this a curse of generality. And, the mathematical program (MP) is, indeed, a very general problem description. Luckily, as will be seen in the following chapters, it turns out that the most of the optimization problems we will encounter belong to, or can be decomposed into, a special subclass of (MP) where the objective function and constraints have convexity properties. Such convex optimization problems have been, and continue to be, extensively studied. Therefore we can rely on a well established theory when analyzing convex problems. Even more importantly, at least from a practical point of view, very efficient algorithms were developed for solving some specific instances of convex problems.
Definition 2.20 (Convex Program). A mathematical program is called the *convex program* if it has the form

\[
\min_z \quad f(z) \quad \text{(CP)}
\]

subj. to \(\begin{align*}
g(z) &\leq 0, \\
z &\in Z,
\end{align*}\)

where \(z \in \mathbb{R}^n_z\) is the optimization variable, the set \(Z \subseteq \mathbb{R}^n_z\) is convex, the cost function \(f : \mathbb{R}^n_z \to \mathbb{R}\) is convex, and all components of the constraint function \(g : \mathbb{R}^n_z \to \mathbb{R}^m_g\) are convex functions.

Although general convex optimization problems can be solved relatively efficiently it is always advantageous to use dedicated solvers for specific problems. A number of standard problems are discussed in the following.

Definition 2.21 (Linear Program). A *linear program* is a convex optimization problem that can be expressed in the form

\[
\min_z \quad c'z \quad \text{(LP)}
\]

subj. to \(G^z z \leq G,\)

where \(z \in \mathbb{R}^n_z\) is the optimization variable, and matrices \(c \in \mathbb{R}^n_z, G^z \in \mathbb{R}^{n_G \times n_z}, G \in \mathbb{R}^{n_G}\) are given problem parameters.

Theoretically every LP (with rational parameters) is solvable in polynomial time by both the ellipsoid method of Khachiyan [Kha79, Sch86] and various interior point methods [Kar84, RTV97]. A practical algorithm to solve an LP with \(n\) variables and \(m\) constraints requires roughly \(O(n^3 m^{0.5} + n^2 m^{1.5})\) operations [dH94]. In this thesis the computational expense of a single LP has similar meaning like single addition or multiplication have in algebraic computations. In many instances we will express complexity of algorithms in terms of number of LP’s one needs to solve, thus establishing the basis for relative comparison between different algorithms. Henceforth with \(lp(n, m)\) we denote the complexity of a single LP with \(n\) variables and \(m\) constraints.

Definition 2.22 (Quadratic Program). A *quadratic program* is the convex optimization problem that can be expressed in the form

\[
\min_z \quad 0.5z'Qz + c'z \quad \text{(QP)}
\]

subj. to \(G^z z \leq G,\)

where \(z \in \mathbb{R}^n_z\) is the optimization variable, while \(Q \in \mathbb{R}^{n_z \times n_z}, Q = Q' \succeq 0, c \in \mathbb{R}^n_z, G^z \in \mathbb{R}^{n_G \times n_z}, G \in \mathbb{R}^{n_G}\).

QPs can be solved with roughly the same efficiency as LPs, i.e. the solvers are approximately 5-times slower than an LP solver [Neu04, page 37].
Definition 2.23 (Semidefinite Program). A semidefinite program is the convex optimization problem that can be expressed in the form

$$\begin{array}{ll}
\min & c'z \\
\text{subj. to} & \begin{cases}
F_0 + \sum_{i=1}^{n_z} z(i)F_i \preceq 0, \\
G^z z = G,
\end{cases}
\end{array}$$

(SDP)

where $z \in \mathbb{R}^{n_z}$ is the optimization variable, while matrices $G^z \in \mathbb{R}^{n_G \times n_z}$, $G \in \mathbb{R}^{n_G}$ and $F_i \in \mathbb{S}^k$, $i = 0, \ldots, n_z$, with $k \in \mathbb{N}$, are given. The inequality

$$F_0 + \sum_{i=1}^{n_z} z(i)F_i \preceq 0$$

is called a linear matrix inequality (LMI).

Remark 2.2 (Standard Form). Various forms of LPs, QPs, LMIs and SPDs can be found in the literature. Very often the differences in the representation are motivated by the origin of a problem at hand. For example, some authors define LP in a standard form

$$\begin{array}{ll}
\min & c'z \\
\text{subj. to} & \begin{cases}
a_i' z = b_i, & i = 1, \ldots, m, \\
z \geq 0,
\end{cases}
\end{array}$$

where $z \in \mathbb{R}^{n_z}$ is the optimization variable, and vectors $a_i \in \mathbb{R}^{n_z}$, $b_i \in \mathbb{R}$, $i = 1, \ldots, m$ are given. Similarly, an SDP in a standard form is defined as

$$\begin{array}{ll}
\min & \text{Tr}(CZ) \\
\text{subj. to} & \begin{cases}
\text{Tr}(A_iZ) = b_i, & i = 1, \ldots, m, \\
Z \succeq 0,
\end{cases}
\end{array}$$

where a real square symmetric matrix $Z \in \mathbb{S}^{n_z}$ is the optimization variable, and matrices $C \in \mathbb{S}^{n_z}$, $A_i \in \mathbb{S}^{n_z}$, $b_i \in \mathbb{R}$, $i = 1, \ldots, m$, are given. Note that $\text{Tr}(CZ) = \sum_{i=1}^{n_z} \sum_{j=1}^{n_z} C_{(i,j)} Z_{(i,j)}$.}

From the standard form it is easier to see why LP is also described as an optimization over the nonnegative orthant, and SDP is described as an optimization over the semidefinite cone. Note, however, that with the appropriate modifications (introduction of new variables, substitution, etc.) standard and all other representations can be rewritten in the forms we use in our definitions (cf. [BV04, BGFB94, Roc97, WSV00]).
[3]

Polytopes

A feasible space of a linear program (see Definition 2.21) is a convex set defined by an intersection of a finite number of linear inequalities. Such convex sets are called polyhedra and they play particularly important role in the context of convex optimization. We will show later that bounded polyhedra, referred to as polytopes, and geometric operations on them are essential tools in describing and solving most constrained optimal control problems tackled in this thesis. We begin by giving definitions for polyhedral and polytopic objects commonly used throughout the thesis.

3.1 Definitions

Most of the definitions given here are standard. For additional details the reader is referred to [Zie94, Gr"u00, Fuk00].

Definition 3.1 (Hyperplane). A hyperplane in $\mathbb{R}^n$ is a set of the form

$$\{ x \in \mathbb{R}^n \mid a'x = b \},$$

where $a \in \mathbb{R}^n$, $a \neq 0$, $b \in \mathbb{R}$. □

Definition 3.2 (Half-space). A half-space in $\mathbb{R}^n$ is a set of the form

$$\mathcal{H} = \{ x \in \mathbb{R}^n \mid a'x \leq b \},$$

where $a \in \mathbb{R}^n$, $a \neq 0$, $b \in \mathbb{R}$. □

Definition 3.3 (Polyhedron). [Gr"u00] A convex set $S \subseteq \mathbb{R}^n$ given as an intersection of a finite number of closed half-spaces

$$S = \{ x \in \mathbb{R}^n \mid S^x x \leq S^c \},$$

is called polyhedron. Here, the inequality $S^x x \leq S^c$, with $S^x \in \mathbb{R}^{n_S \times n}$, $S^c \in \mathbb{R}^{n_S}$, $n_S < \infty$, is considered component-wise. □

Definition 3.4 (Polytope). [Gr"u00] A bounded polyhedron $P \subset \mathbb{R}^n$

$$P = \{ x \in \mathbb{R}^n \mid P^x x \leq P^c \},$$

is called polytope. Here, the inequality $P^x x \leq P^c$, with $P^x \in \mathbb{R}^{n_P \times n}$, $P^c \in \mathbb{R}^{n_P}$, $n_P < \infty$, is considered component-wise. □
We say that a polytope $\mathcal{P} \subset \mathbb{R}^n$, $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq Pc\}$, is full-dimensional if it is possible to fit a non-empty $n$-dimensional ball in $\mathcal{P}$, cf. Definition 2.8,

$$\exists x \in \mathbb{R}^n, \epsilon > 0 : B(x, \epsilon) \subset \mathcal{P},$$

or, equivalently,

$$\exists x \in \mathbb{R}^n, \epsilon > 0 : \|\delta\|_2 \leq \epsilon \Rightarrow P^x(x + \delta) \leq Pc.$$  

(3.4)

Otherwise, we say that polytope $\mathcal{P}$ is lower-dimensional. A polytope is referred to as empty if

$$\not\exists x \in \mathbb{R}^n : P^x x \leq Pc.$$  

(3.5)

Furthermore, if $\|P^x(i)\|_2 = 1$, where $P^x(i)$ denotes $i$-th row of a matrix $P^x$, we say that the polytope $\mathcal{P}$ is normalized. One of the fundamental properties of a polytope is that it can be described in half-space representation as in (3.2) or in vertex representation, as given below,

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{v_P} \alpha_i V^P_i, 0 \leq \alpha_i \leq 1, \sum_{i=1}^{v_P} \alpha_i = 1\},$$

(3.6)

where $V^P_i \in \mathbb{R}^n$ denotes the $i$-th vertex of $\mathcal{P}$, and $v_P$ is the total number of vertices of $\mathcal{P}$.

We will henceforth refer to the half-space representation (3.2) and vertex representation (3.6) as $\mathcal{H}$- and $\mathcal{V}$-representation respectively.

**Figure 3.1:** Illustration of a polytope in $\mathcal{H}$- and $\mathcal{V}$- representation.

**Definition 3.5 (Face).** [Zie94] A linear inequality $a^T x \leq b$ is called valid for a polyhedron $\mathcal{P}$ if $a^T x \leq b$ holds for all $x \in \mathcal{P}$. A subset of a polyhedron is called a face of $\mathcal{P}$ if it can be represented as

$$\mathcal{F} = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid a^T x = b\},$$

(3.7)

for some valid inequality $a^T x \leq b$. The faces of polyhedron $\mathcal{P}$ of dimension $0$, $1$, $(n - 2)$ and $(n - 1)$ are called vertices, edges, ridges and facets, respectively.

$\square$
We see that a set \( F \subseteq P \) is called a face of \( P \) if it is either \( \emptyset \), \( P \) itself or the intersection of \( P \) with a hyperplane derived from a valid inequality [Fuk00]. An empty set \( \emptyset \) is a face of every polyhedron and by definition it has dimension \(-1\).

According to our definition every polytope represents a convex, compact (i.e. bounded and closed) set. We will see later that polytopes – as simple objects as they are – play an instrumental role in the derivation of optimal control strategies for constrained linear and piecewise affine systems. Very often, however, we also encounter sets that are disjoint or non-convex but can be represented by the union of finite number of polytopes. Therefore, it is useful to define the following mathematical concept.

**Definition 3.6 (Polytopal Complex).** [Zie94] A *polytopal complex* \( C \) is a finite collection of polytopes in \( \mathbb{R}^n \) such that

- the empty set is in \( C \)
- if \( P \in C \), then all the faces of \( P \) are also in \( C \),
- the intersection \( P \cap Q \) of two polytopes \( P, Q \in C \) is a face both of \( P \) and \( Q \).

**Example 3.1.** Here are several polytopal complexes in \( \mathbb{R}^1 \): \( \{\emptyset, 0, 2, [0, 2]\}, \{\emptyset, 0, 2, 4, [0, 2]\} \). Contrary to those, the following sets are not polytopal complexes in \( \mathbb{R}^1 \): \( \{\emptyset, 0, [0, 2]\} \) – because of a missing polytope \( \{2\} \), and \( \{\emptyset, -2, 0, 2, [-2, 0], [-2, 2], [0, 2]\} \) – because intersection of two polytopes \([-2, 0]\) and \([-2, 2]\) is not a face of \([-2, 2]\).

Computation of a polytopal complex is quite expensive. For every polytope in the complex we need to compute all the faces (i.e. lower-dimensional polytopes). Since we usually work only with full-dimensional polytopes (cf. Remark 3.1) it is reasonable to introduce a more practical concept.

**Definition 3.7 (P-collection).** A set \( C \subseteq \mathbb{R}^n \) is called the *P-collection* (in \( \mathbb{R}^n \)) if it is collection of a finite number of \( n \)-dimensional polytopes, i.e.

\[
C = \{C_i\}_{i=1}^{N_C},
\]

where \( C_i := \{x \in \mathbb{R}^n \mid C^x_i \leq C^c_i\} \), \( \text{dim}(C_i) = n, i = 1, \ldots, N_C, \text{ with } N_C < \infty \).

**Definition 3.8 (Underlying Set).** The *underlying set* of a P-collection \( C = \{C_i\}_{i=1}^{N_C} \) is the point set

\[
C := \bigcup_{P \in C} P = \bigcup_{i=1}^{N_C} C_i.
\]

**Example 3.2.** A collection \( \mathcal{R} = \{\{-2, -1\}, [0, 2], [2, 4]\} \) is a P-collection in \( \mathbb{R}^1 \) with the underlying set \( \mathcal{R} = [-2, -1] \cup [0, 4] \). As another example, \( \mathcal{R} = \{\{-2, 0\}, [-1, 1], [0, 2]\} \) is a P-collection in \( \mathbb{R}^1 \) with underlying set \( \mathcal{R} = [-2, 2] \). Clearly, polytopes that define P-collection can overlap, while the underlying sets can be disconnected and non-convex.
Usually it is clear from the context if we are talking about the P-collection or we are referring to the underlying set of a P-collection, in which case, for simplicity, we use the same notation for both.

**Definition 3.9** (Partition). A collection of sets \( \{ R_i \}_{i=1}^{N_R} \) is a partition of a set \( \mathcal{P} \) if: (i) \( \mathcal{P} = \bigcup_{i=1}^{N_R} R_i \), and (ii) \( R_i \cap R_j = \emptyset, \forall i \neq j \), with \( i, j \in \{1, \ldots, N_R\} \).

**Definition 3.10** (Polyhedral Partition). A collection of sets \( \{ \overline{R_i} \}_{i=1}^{N_R} \) is a polyhedral partition of a set \( \mathcal{P} \) if \( \{ R_i \}_{i=1}^{N_R} \) is a partition of \( \mathcal{P} \) and the sets \( \overline{R_i} \) are polyhedra, where \( i = 1, \ldots, N_R \), and \( \overline{R_i} \) denotes the closure of \( R_i \).

### 3.2 Basic Operations on Polytopes

We will now define some basic operations and functions on polytopes. Note that although we focus on polytopes and polytopic objects most of the operations described here are directly (or with minor modifications) applicable to polyhedral objects. Additional details on polytope computation can be found in [Zie94, Grüßü00, Fuk00]. All operations and functions described in this chapter are contained in the MPT toolbox (see Part IV or [KGBM04, KGBC04]).

#### 3.2.1 Minimal Representation

We say that a polytope \( \mathcal{P} \subset \mathbb{R}^n, \mathcal{P} = \{ x \in \mathbb{R}^n \mid P^x x \leq P^c \} \) is in a minimal representation if the removal of any row in \( P^x x \leq P^c \) would change it (i.e., there are no redundant half-spaces). The computation of minimal representation (henceforth referred to as polytope reduction) of polytopes is discussed in [Fuk00] and generally requires to solve one LP for each half-space defining the non-minimal representation of \( \mathcal{P} \). We report this simple implementation of the polytope reduction in Algorithm 3.1. An improved algorithm for polytope reduction is discussed in [SLG+04] where the authors combine procedure outlined in Algorithm 3.1 with heuristic methods, such as bounding-boxes and ray-shooting, to discard redundant constraints more efficiently.

It is straightforward to see that a normalized, full-dimensional polytope \( \mathcal{P} \) has a unique minimal representation.\(^1\) This fact is very useful in practice. Normalized, full-dimensional polytopes in a minimal representation allow us to avoid any ambiguity when comparing them and very often speed-up other polytope manipulations.

**Algorithm 3.1** (Polytope in minimal representation).

**INPUT** \( \mathcal{P} = \{ x \mid P^x x \leq P^c \} \), with \( P^x \in \mathbb{R}^{n \times n_x}, P^c \in \mathbb{R}^{n_p} \)

**OUTPUT** \( Q = \{ x \mid Q^x x \leq Q^c \} := \text{minrep}(\mathcal{P}) \)

**LET** \( I \leftarrow \{1, \ldots, n_P\} \)

**FOR** \( i = 1 \) TO \( n_P \)

\(^1\)Note that ‘unique’ here means that for \( \mathcal{P} := \{ x \in \mathbb{R}^n \mid P^x x \leq P^c \} \) the matrix \( [P^x P^c] \) consist of the unique set of row vectors. The order in which those rows are stored is irrelevant.
LET $I \leftarrow I \setminus \{i\}$
LET $f^* \leftarrow \max_x P^x_{(i)} x$, subj. to $P^x_{(I)} x \leq P^c_{(i)}$, $P^x_{(i)} x \leq P^c_{(i)} + 1$
IF $f^* > P^c_{(i)}$ THEN $I \leftarrow I \cup \{i\}$
END
LET $Q^x = P^x_{(I)}$, $Q^c = P^c_{(I)}$

Remark 3.1 (Full-dimensional polytopes). Throughout this thesis we will mostly work with full-dimensional polytopes. The reason is twofold: i) numerical issues with lower-dimensional polytopes, and, more importantly, ii) full-dimensional polytopes are sufficient for describing solutions to the problems we handle in this thesis. For the same reason the MPT toolbox (see Part IV or [KGBM04]) only deals with full-dimensional polytopes. Polyhedra and lower-dimensional polytopes (with the exception of an empty polytope) are not considered.

3.2.2 Chebychev Ball

The Chebychev Ball of a polytope $\mathcal{P} = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$, with $P^x \in \mathbb{R}^{n_P \times n}$, $P^c \in \mathbb{R}^{n_P}$, corresponds to the largest radius ball $B(x_c, R)$ with center $x_c$, such that $B(x_c, R) \subset \mathcal{P}$. The center and radius of the Chebychev ball can be easily found by solving the following LP [BV04]

$$\max_{x_c, R} R$$
subj. to $P^x_{(i)} x_c + R \|P^x_{(i)}\|_2 \leq P^c_{(i)}$, $i = 1, \ldots, n_P$, (3.10b)

where $P^x_{(i)}$ denotes the $i$-th row of $P^x$. If the obtained radius $R = 0$, then the polytope is lower-dimensional; if $R < 0$, then the polytope is empty. Therefore, an answer to the question “is polytope $\mathcal{P}$ full-dimensional/empty?” is obtained at the expense of only 1 linear program. Furthermore, for a full-dimensional polytope we also get a point $x_c$ that is in the strict interior of $\mathcal{P}$. One word of caution: the center of a Chebyshev Ball in (3.10) is not necessarily unique point (e.g. when $\mathcal{P}$ is a rectangle). There are other types of unique interior points one could compute for a full-dimensional polytope, e.g., center of the largest volume ellipsoid, analytic center, etc., but those computations are formulated as SDP’s and hence they are more expensive to carry out than the Chebyshev Ball computation [BV04].

3.2.3 Projection

Given a polytope $\mathcal{P} = \{[x'y']' \in \mathbb{R}^{n+m} \mid P^x x + P^y y \leq P^c\} \subset \mathbb{R}^{n+m}$ the projection onto the $x$-space $\mathbb{R}^n$ is defined as

$$\text{proj}_x(\mathcal{P}) := \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m : P^x x + P^y y \leq P^c\}.$$ (3.11)

An illustration of a projection operation is given in Figure 3.2(a). Current projection methods that can operate in general dimensions can be grouped into four classes: Fourier elimination [Cer63, KS90], block elimination [Bal98], vertex based approaches [FLL00] and wrapping-based techniques [JKM04]. For a good introduction to projection, we refer the reader to [JKM04] and the references therein.
3 Polytopes

3.2.4 Set-Difference

The Set-Difference of two polytopes $\mathcal{P}$ and $\mathcal{Q}$

$$\mathcal{R} = \mathcal{P} \setminus \mathcal{Q} := \{x \in \mathbb{R}^n \mid x \in \mathcal{P}, x \notin \mathcal{Q}\},$$

is, in general, given as a P-collection $\mathcal{R} = \bigcup_i \mathcal{R}_i$, which can be computed by consecutively inverting the half-spaces defining $\mathcal{Q}$ as described in [BMDP02] (see Figure 3.3). Note that here we use the term P-collection in the dual context of both P-collection and its underlying set (cf. Definition 3.7 and 3.8). The precise statement would say that $\mathcal{R} = \mathcal{P} \setminus \mathcal{Q}$, where $\mathcal{R}$ is underlying set of P-collection $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^N$. However, whenever it is clear from context what are we referring to, as is the case here, we will abuse the notation and use the former, more compact form.

Figure 3.3: Illustration of the set-difference operation.

Remark 3.2. The set difference of two intersecting polytopes $\mathcal{P}$ and $\mathcal{Q}$ (or any closed sets) is not a closed set. This means that some borders of polytopes $\mathcal{R}_i$ from a P-collection $\mathcal{R} = \mathcal{P} \setminus \mathcal{Q}$
are open, while other borders are closed. Even though it is possible to keep track of the origin of particular borders of $R_i$, thus specifying if they are open or closed, we are not doing it in the algorithms described in this thesis nor in MPT [KGBM04, KGBC04], cf. Remark 3.1. In computations, we will henceforth only consider the closure of sets $R_i$.

The set difference between two P-collections $P$ and $Q$ can be computed as described in [BT03, GKB03, RKM03]. A computationally very demanding part of many algorithms reported in this thesis is an answer to the question: is a polytope $P$ fully covered by a P-collection $Q := \bigcup_i Q_i$? Because this particular computation with P-collections plays a very important role in the rest of the thesis we report the approach from [BT03] separately in Section 3.3.

### 3.2.5 Convex Hull

The convex hull of a set of points $\mathcal{V} = \{V_i\}_{i=1}^{N_V}$, with $V_i \in \mathbb{R}^n$, is a polytope defined as

$$\text{co}(\mathcal{V}) = \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{N_V} \alpha_i V_i, \ 0 \leq \alpha_i \leq 1, \ \sum_{i=1}^{N_V} \alpha_i = 1\}. \quad (3.13)$$

The convex hull operation is used to switch from a $V$-representation of a polytope to a $H$-representation. The convex hull of a union of polytopes $R_i \subset \mathbb{R}^n$, $i = 1, \ldots, N_R$, is a polytope

$$\text{co} \left( \bigcup_{i=1}^{N_R} R_i \right) := \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{N_R} \alpha_i x_i, \ x_i \in R_i, \ 0 \leq \alpha_i \leq 1, \ \sum_{i=1}^{N_R} \alpha_i = 1\}. \quad (3.14)$$

An illustration of the convex hull operation is given in Figure 3.4. Construction of the convex hull is an expensive operation which is exponential in the number of facets of the original polytope. An efficient software implementation is available from [Fuk97].

### 3.2.6 Envelope

The envelope of two $H$-polyhedra $P = \{x \in \mathbb{R}^n \mid P^x x \leq P^c\}$ and $Q = \{x \in \mathbb{R}^n \mid Q^x x \leq Q^c\}$ is an $H$-polyhedron

$$\text{env}(P, Q) = \{x \in \mathbb{R}^n \mid \bar{P}^x x \leq \bar{P}^c, \ \bar{Q}^x x \leq \bar{Q}^c\}, \quad (3.15)$$

where $\bar{P}^x x \leq \bar{P}^c$ is the subsystem of $P^x x \leq P^c$ obtained by removing all the inequalities not valid for the polyhedron $Q$, and $\bar{Q}^x x \leq \bar{Q}^c$ are defined in a similar way with respect to $Q^x x \leq Q^c$ and $P$ [BFT01]. In a similar fashion definition can be extended to the case of the envelope of a P-collection. An illustration of the envelope operation is depicted in Figure 3.5. The computation of the envelope is relatively cheap since it only requires the solution to one LP for each facet of $P$ and $Q$. Note that envelope of two (or more) polytopes is not necessarily bounded set (e.g. when $P \cup Q$ is shaped like a star).
3.2.7 Vertex Enumeration

The operation of extracting the vertices of a polytope $\mathcal{P}$ given in $\mathcal{H}$-representation is referred to as vertex enumeration. This operation is the dual to the convex hull operation and the algorithmic implementation is identical to a convex hull computation, i.e. given a set of extreme points $\mathcal{V} = \{V_i\}_{i=1}^{N_V} = \text{vert}(\mathcal{P})$ of a polytope $\mathcal{P}$ given in $\mathcal{H}$-representation it holds that $\mathcal{P} = \text{co}(\mathcal{V})$, where the operator vert denotes the vertex enumeration. The necessary computational effort is exponential in the number of input facets. Two different approaches to vertex enumeration exist: the double description method [FP96] and reverse search [Avi00]. An efficient implementation of the double description method is available in [Fuk97].

3.2.8 Pontryagin Difference

The Pontryagin difference (also known as Minkowski-Difference) of two polytopes $\mathcal{P}$ and $\mathcal{Q}$ is a polytope

$$
\mathcal{P} \ominus \mathcal{Q} := \{x \in \mathbb{R}^n \mid x + q \in \mathcal{P}, \forall q \in \mathcal{Q}\}.
$$

(3.16)

The Pontryagin difference can be efficiently computed for polytopes by solving a sequence of LPs [KG98]. For special cases (e.g. when $\mathcal{Q}$ is a hypercube), even more efficient computa-
tional methods exist [KM03]. An illustration of the Pontryagin difference is given in Figure 3.6.

![Pontryagin difference](image1)

![Minkowski sum](image2)

Figure 3.6: Illustration of the Pontryagin difference and Minkowski sum operations.

### 3.2.9 Minkowski Sum

The Minkowski sum of two polytopes $P$ and $Q$ is a polytope

$$
P \oplus Q := \{ x + q \in \mathbb{R}^n \mid x \in P, \ q \in Q \}. \quad (3.17)
$$

The Minkowski sum is a computationally expensive operation which requires either vertex enumeration and convex hull computation in $n$-dimensions or a projection from $2n$ down to $n$ dimensions. The implementation of the Minkowski sum via projection is described below.

$$
P = \{ y \in \mathbb{R}^n \mid P^y y \leq P^c \}, \quad Q = \{ z \in \mathbb{R}^n \mid Q^z z \leq Q^c \},
$$

it holds that

$$
W = P \oplus Q
= \left\{ x \in \mathbb{R}^n \mid x = y + z, \ P^y y \leq P^c, \ Q^z z \leq Q^c, \ y, z \in \mathbb{R}^n \right\}
= \left\{ x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^n, \ subj. \ to \ P^y y \leq P^c, \ Q^z (x - y) \leq Q^c \right\}
= \left\{ x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^n, \ subj. \ to \ \begin{bmatrix} 0 & P^y \\ Q^z & -Q^z \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} P^c \\ Q^c \end{bmatrix} \right\}
= \text{proj}_x \left( \left\{ [x']' \in \mathbb{R}^{n+n} \mid \begin{bmatrix} 0 & P^y \\ Q^z & -Q^z \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \leq \begin{bmatrix} P^c \\ Q^c \end{bmatrix} \right\} \right).
$$

Both the projection and vertex enumeration based methods are implemented in the MPT toolbox (see Part IV or [KGBM04]). An illustration of the Minkowski sum is given in Figure 3.6.

**Remark 3.3.** The Minkowski sum is not the complement of the Pontryagin difference. For two polytopes $P$ and $Q$, it holds that $(P \ominus Q) \oplus Q \subseteq P$. This is illustrated in Figure 3.7.
3.3 Polycover and Regiondiff

The problem of checking if some polytope is covered with the union of other polytopes emerges very often in the construction of explicit solution for the constrained finite time optimal control (CFTOC) of piecewise affine (PWA) systems. The need for solving this type of a problem is also common in the computation of the (positive) control invariant sets, infinite time solution or controllers with reduced complexity for PWA systems. Thus, it is important to have an algorithm that checks if a polytope $P$ is fully covered by a $P$-collection $Q := \bigcup_i Q_i$ in an efficient manner. We will actually differentiate between two related problems:

**polycover**: Check if $P \subseteq \bigcup_i Q_i$, and

**regiondiff**: Compute $P$-collection $R = P \setminus (\bigcup_i Q_i)$.

Clearly, **polycover** is just a special case of **regiondiff**, where resulting $P$-collection $R = \emptyset$. However, we treat **polycover** separately since it allows more efficient computation as will be shown latter.

It is straightforward to extend the above problems to the case where both $P$ and $Q$ are $P$-collections:

- Set difference $R = P \setminus Q$ can be computed by cycling through all polytopes $P_i$. We have $R = \bigcup_i (P_i \setminus (\bigcup_j Q_j))$.
- Verification of $P \subseteq Q$. We have $P \subseteq Q \iff P \setminus Q = \emptyset$.
- Verification of $P = Q$. We have $P = Q \iff (P \setminus Q = \emptyset$ and $Q \setminus P = \emptyset$).

One idea of solving the polycover problem is inspired by the following observation

$$P \subseteq \bigcup_i Q_i \iff P = \bigcup_i (P \cap Q_i).$$

Therefore, we could create $R_i = Q_i \cap P_i$ for $i = 1, \ldots, N_Q$ and then compute the union of the collection of polytopes $\{R_i\}$ by using the **polyunion** algorithm for computing the
convex union of $\mathcal{H}$-polyhedra reported in [BFT01]. If approach [BFT01] succeeds (i.e., the union is convex set) and the resulting polytope is equal to $P$ then $P$ is covered by $\{Q_i\}_i$, otherwise it is not. However, this approach is computationally very expensive. The algorithm in [BFT01] is based on the idea that whenever $P := \bigcup_i R_i$ is a polyhedron then $P = \text{env}(\{R_i\}_i)$. Unfortunately the underlying computation is very expensive, and one needs to solve $O(\Pi_{i=1}^{N_Q} n_Q_i)$ LP’s with $n_x$ variables and $\sum_{i=1}^{N_Q} n_Q_i$ constraints. Thus, polyunion computation from [BFT01] becomes quickly prohibitive with the increasing number of polytopes and constraints.

In the following sections we explore different, hopefully more efficient, ways of checking if $P \subseteq (\bigcup_{i=1}^{N_Q} Q_i)$.

### 3.3.1 Polycover: MILP formulation

When some of the optimization variables in a linear program (see Definition 2.21) are constrained to integer values the ensuing problem is called a mixed integer linear program (MILP).

**Definition 3.11 (Mixed Integer Linear Program).** A mixed integer linear program (MILP) is a non-convex optimization problem that can be expressed in the form

\[
\begin{align*}
\min_x & \quad f^\prime x \\
\text{subj. to} & \quad G^x x \leq G^c,
\end{align*}
\]

where $x \in \mathbb{R}^{n_r} \times \{0, 1\}^{n_b}$ is the optimization variable, $n_r$ is the number of real valued variables, $n_b$ is the number of binary (or, in general, integer) variables, and matrices $f \in \mathbb{R}^{n_r}$, $G^x \in \mathbb{R}^{n_G \times n}$, $G \in \mathbb{R}^{n_G}$, with $n = n_r + n_b$, are given problem parameters.

We note that $P$ is not fully covered by $Q$, i.e.

\[
P \not\subseteq (\bigcup_{i=1}^{N_Q} Q_i)
\]

if and only if there is a point $x$ inside of $P$ that violates at least one of the constraints of each $Q_i, i = 1, \ldots, N_Q$. This is equivalent to the following set of conditions

\[
\exists x \in P : \exists j_i \in \{1, \ldots, n_Q_i\}, \quad [Q_i^r]_{(j_i)} x - [Q_i^c]_{(j_i)} > 0, \quad i = 1, \ldots, N_Q.
\]

To express this violation of constraints we introduce slack variables

\[
y_{i,j}(x) = \begin{cases} 
[Q_i^r]_{(j)} x - [Q_i^c]_{(j)} & \text{if } [Q_i^r]_{(j)} x - [Q_i^c]_{(j)} \geq 0, \\
0 & \text{if } [Q_i^r]_{(j)} x - [Q_i^c]_{(j)} \leq 0,
\end{cases}, \quad j = 1, \ldots, n_Q_i, \quad i = 1, \ldots, N_Q.
\]

The expression (3.19) can now be posed as a feasibility question in $x$ and $y_{i,j}$

\[
P^x x \leq P^c, \\
\sum_{j=1}^{n_Q_i} y_{i,j} > 0, \quad i = 1, \ldots, N_Q
\]
Checking the condition (3.21) is still not possible with standard solvers, since the relation (3.20) describes a non-linear function. However, by introducing auxiliary binary variables one can rewrite (3.20) as the following equivalent set of linear inequalities (cf. [BM99])

\[
\begin{bmatrix}
0 & -m \\
0 & -M \\
1 & -M \\
1 & -m \\
1 & -M \\
\end{bmatrix}
\begin{bmatrix}
y_{i,j} \\
\delta_{i,j}
\end{bmatrix} \leq
\begin{bmatrix}
[Q^r_i]_{(j)}x - [Q^r_i]_{(j)} - m \\
-[Q^r_i]_{(j)}x + [Q^r_i]_{(j)} \\
0 \\
[Q^r_i]_{(j)}x - [Q^r_i]_{(j)} - m \\
-[Q^r_i]_{(j)}x + [Q^r_i]_{(j)} + M
\end{bmatrix}
\]

(3.22)

\[\delta_{i,j} \in \{0, 1\}, \quad j = 1, \ldots, n_Q, \quad i = 1, \ldots, N_Q,\]

where \(\delta_{i,j}\) are auxiliary binary variables and \(m, M \in \mathbb{R}\) are bounds on constraint expressions that can be pre-computed (or overestimated) beforehand

\[
m \leq \min_{x, i, j} [Q^r_i]_{(j)}x - [Q^r_i]_{(j)} \quad \text{sub. to} \quad P^x x \leq P^c \\
\text{subj. to} \quad j \in \{1, \ldots, n_Q\} \quad \text{and} \quad i \in \{1, \ldots, N_Q\}
\]

(3.23)

\[
M \geq \min_{x, i, j} [Q^r_i]_{(j)}x - [Q^r_i]_{(j)} \quad \text{sub. to} \quad P^x x \leq P^c \\
\text{subj. to} \quad j \in \{1, \ldots, n_Q\} \quad \text{and} \quad i \in \{1, \ldots, N_Q\}
\]

(3.24)

Actually, in terms of the number of inequalities that are used, (3.22) can be further simplified to

\[
\begin{bmatrix}
-1 & 0 \\
-1 & 0 \\
1 & -m \\
1 & -M
\end{bmatrix}
\begin{bmatrix}
y_{i,j} \\
\delta_{i,j}
\end{bmatrix} \leq
\begin{bmatrix}
0 \\
- [Q^r_i]_{(j)}x + [Q^r_i]_{(j)} \\
[Q^r_i]_{(j)}x - [Q^r_i]_{(j)} - m \\
0
\end{bmatrix}
\]

(3.25)

\[\delta_{i,j} \in \{0, 1\}, \quad j = 1, \ldots, n_Q, \quad i = 1, \ldots, N_Q\]

Since (3.21) and (3.25) describe a Mixed Integer Linear Programming (MILP) feasibility problem it follows that we can check if \(\mathcal{P} \not\subseteq (\bigcup_{i=1}^{N_Q} Q_i)\) by solving an MILP feasibility problem. However, instead of solving a feasibility MILP problem with (3.21) it may be more useful (and numerically robust) to solve the following optimality MILP problem

\[
\max_{\lambda, x, \delta_{i,j}, y_{i,j}} \lambda \quad \text{sub. to} \quad \begin{cases}
P^x x \leq P^c, \\
\sum_{j=1}^{n_Q} y_{i,j} \geq \lambda, \quad i = 1, \ldots, N_Q \\
\sum_{j=1}^{n_Q} d_{i,j} \geq 1, \quad i = 1, \ldots, N_Q \\
\text{Eq.(3.25)}
\end{cases}
\]

(3.26)

Effectively, the optimal value \(\lambda^*\) is related to the size of a largest non-covered part of \(\mathcal{P}\).
Theorem 3.1. Let $\lambda^*$ be the solution to the problem (3.26), then $\mathcal{P} \not\subseteq \left( \bigcup_{i=1}^{N_Q} \mathcal{Q}_i \right)$ if and only if $\lambda^* > 0$.

Proof. Follows from the construction of the MILP problem (3.26).

Remark 3.4. Strictly speaking, condition $\sum_{j=1}^{N_Q} d_{i,j} \geq 1$ in (3.26) is redundant, but it reduces the problems with the integrality tolerances in existing MILP solvers. Also note that when solving (3.26) there is no need for condition $y_{i,j} \geq 0$ (first row in Eq. (3.25)).

The MILP problem (3.26) has $n_P + 2N_Q + 3\sum_{i=1}^{N_Q} n_{Q_i}$ constraints, $n_x + 1 + \sum_{i=1}^{N_Q} n_{Q_i}$ real variables and $\sum_{i=1}^{N_Q} n_{Q_i}$ binary variables.

3.3.2 Polycover: Branch & Bound algorithm

Let us repeat problem formulation once again. Let $\mathcal{P} = \{ x \mid P^x x \leq P^c \}$ be a polyhedron in $\mathbb{R}^{n_x}$ given as the intersection of $n_P$ half-spaces and $\mathcal{Q}_i = \{ x \in \mathbb{R}^{n_x} \mid Q^e_i x \leq Q^c_i \}$ be $N_Q$ polytopes in $\mathbb{R}^{n_x}$ given as the intersection of $n_{Q_i}$ half-spaces (i.e. $Q^e_i$ is a $n_{Q_i} \times n_x$ matrix).

We want to determine if $\mathcal{P} \subseteq \left( \bigcup_{i=1}^{N_Q} \mathcal{Q}_i \right)$. Here we consider an exact solution to the problem by using the algorithm that (as an extension) computes the set $\mathcal{R} = \mathcal{P} \setminus \left( \bigcup_{i=1}^{N_Q} \mathcal{Q}_i \right)$, where $\mathcal{R}$ (if not empty) is given as a union of polyhedra. Without loss of generality we will assume that the following assumption holds:

Assumption 3.1. Let $\mathcal{P}$ and $\mathcal{Q}_i$, $i = 1, \ldots, N_Q$, be full-dimensional polyhedra in $\mathbb{R}^{n_x}$ given in the minimal $\mathcal{H}$-representation: $\mathcal{P} = \{ x \mid P^x x \leq P^c \}$, $\mathcal{Q}_i = \{ x \mid Q^e_i x \leq Q^c_i \}$, $P^x \in \mathbb{R}^{n_P \times n_x}$, $P^c \in \mathbb{R}^{n_P}$, $Q^e_i \in \mathbb{R}^{n_{Q_i} \times n_x}$, $Q^c_i \in \mathbb{R}^{n_{Q_i}}$, such that $\mathcal{P} \cap \mathcal{Q}_i \neq \emptyset$, $\forall i \in \{1, \ldots, N_Q\}$.

Note that it is always possible to obtain normalized polyhedron in minimal representation (see Section 3.2.1) and to remove those $\mathcal{Q}_i$ that do not intersect $\mathcal{P}$, e.g. by checking if Chebyshev Ball (see Section 3.2.2) of a joint polyhedra $\{ x \mid P^x x \leq P^c, Q^e_i x \leq Q^c_i \}$ is non-empty.

Algorithm 3.2 (Set Difference, $\mathcal{P} \setminus \{ \mathcal{Q}_i \}_{i=1}^{N_Q}$).

**INPUT** Polyhedron $\mathcal{P} = \{ x \mid P^x x \leq P^c \}$ and $\mathcal{Q} = \{ \mathcal{Q}_i \}_{i=1}^{N_Q}$ satisfying Assumption 3.1

**OUTPUT** $\mathcal{R} = \{ \mathcal{R}_i \}_{i=1}^{N_R} := \text{regiondiff}(\mathcal{P}, \{ \mathcal{Q}_i \}_{i=1}^{N_Q})$

1. Identify for each $\mathcal{Q}_i$, $i = 1, \ldots, N_Q$, the set of so-called active constraints

$$\mathcal{A}_i = \{ j \mid \exists x \in \mathbb{R}^{n_x} : P^x x \leq P^c, [Q^e_i]_{(j)} x > [Q^c_i]_{(j)}, j \in \{1, \ldots, n_{Q_i}\} \}. \quad (3.27)$$

2. **IF** $\exists i : \mathcal{A}_i = \emptyset$ **THEN LET** $\mathcal{R} \leftarrow \emptyset$ and **EXIT**

3. Remove non-active constraints from $\mathcal{Q}_i$: $Q^e_i \leftarrow [Q^e_i]_{(\mathcal{A}_i)}$, $Q^c_i \leftarrow [Q^c_i]_{(\mathcal{A}_i)}$, $i = 1, \ldots, N_Q$.

4. **LET** $\mathcal{R} = \text{feaspoly}(\mathcal{P}, \{ \mathcal{Q}_i \}_{i=1}^{N_Q})$

5. Remove redundant constraints from $\mathcal{R}_i$: $\mathcal{R}_i \leftarrow \text{minrep}(\mathcal{R}_i)$, $i = 1, \ldots, N_R$. 
Algorithm 3.3 \((\text{feaspoly}(P, \{Q_i\}_{i=1}^{N_Q}))\).

**INPUT**  
Polyhedron \(P = \{ x \mid P^x x \leq P^c \}\) and \(Q = \{Q_i\}_{i=1}^{N_Q}\) satisfying Assumption 3.1

**OUTPUT**  
Set of feasible polyhedra \(R = \{R_i\}_{i=1}^{N_R} := \text{feaspoly}(P, \{Q_i\}_{i=1}^{N_Q})\)

1. **LET** \(R = \emptyset\), \(k = 1\)

2. **IF** \(\exists x : P^x x < P^c, Q^x_k x < Q^c_k\) **THEN** go to Step 4, **ELSE** \(k \leftarrow k + 1\)

3. **IF** \(k > N_Q\) **THEN** \(R \leftarrow P\) and **EXIT**, **ELSE** go to Step 2

4. **FOR** \(j = 1\) TO \(n_{Q_k}\)

   4.1. **IF** \(\exists x : P^x x \leq P^c, [Q^x_k](j)x > [Q^c_k](j)\)

      4.1.1. **LET** \(\bar{P} = P \cap \{ x \mid [Q^x_k](j)x \geq [Q^c_k](j) \}\)

      4.1.2. **IF** \(N_Q > k\) **THEN** \(R \leftarrow R \cup \text{feaspoly}(\bar{P}, \{Q_i\}_{i=k+1}^{N_Q})\), **ELSE** \(R \leftarrow R \cup \bar{P}\)

   4.2. **LET** \(P \leftarrow P \cap \{ x \mid [Q^x_k](j)x \leq [Q^c_k](j) \}\)

5. **EXIT**

**Remark 3.5.** Steps 1 of Algorithm 3.2 and Step 2 and Step 4.1 of Algorithm 3.3 are simple feasibility LP’s. Note that Algorithm 3.3 would work even if Steps 2 and 3 were not present. However, those steps have huge impact on the complexity of the solution and the speed of computation since this avoids unnecessary check at later stages (i.e. lower branches) of the procedure.

**Remark 3.6.** The \text{regiondiff} algorithm (Algorithm 3.2) basically implements a branch \\& bound search for feasible constraint combinations where each branch corresponds to the inversion of a facet of \(Q_i\) obtained in Step 3 of Algorithm 3.2. The approach used to partitioning the space is based on [BMDP02].

The most demanding part of the set-difference computation lies in Algorithm 3.3 (and it’s recursive calls). Therefore in this analysis we neglect the cost of removal of the non-active constraints from \(Q_i\) or computing the minimal representation of the regions \(R_i\) that describe the set difference.

The worst case complexity of Algorithm 3.3 can be easily established from it’s tree-like exploration of the space. We see that the depth of the tree is equal to the number of regions \(Q_i, N_Q\). Furthermore every node on the level \(i\) has at most \(n_{Q_{i+1}}\) children nodes on the level \(i + 1\), where \(n_{Q_i}\) denotes the number of active constraints of a polytope \(Q_i\). Computation at every node on the level \(i\) of the tree involves solution of an LP with \(n_x + 1\) variables and
at most \( n_P + \sum_{j=1}^{i} n_{Q_j} \) constraints. Therefore, the overall complexity of Algorithm 3.3 is bounded with

\[
\sum_{i=1}^{N_Q} \left[ lp(n_x + 1, n_P + \sum_{j=1}^{i} n_{Q_j}) \prod_{j=1}^{i} n_{Q_j} \right],
\]

(3.28)

where \( lp(n, m) \) denotes the complexity of a single LP with \( n \) variables and \( m \) constraints.

To estimate the number of regions \( N_R \) generated by the set-difference computation we recall Buck’s formula [Buc43] for the hyperplane arrangement problem that gives an upper bound\(^2\) for the maximal number of cells created by \( M \) hyperplanes in \( \mathbb{R}^{n_x} \)

\[
\sum_{i=0}^{n_x} \left( \begin{array}{c} M \\ i \end{array} \right).
\]

(3.29)

By letting \( M \) be equal to the total number of active-constraints, i.e. \( M = \sum_i n_{Q_i} \), from (3.29) follows

\[
N_R \leq \sum_{i=0}^{n_x} \left( \begin{array}{c} M \\ i \end{array} \right) = O(M^{n_x}).
\]

(3.30)

**Remark 3.7.** In practice the complexity estimate given by (3.28) is very conservative. Each polytope \( R_i, i = 1, \ldots, N_R \), corresponds to the leaf (bottom) node of the exploration tree in Algorithm 3.2 which implies a substantial pruning of the tree during the execution of the algorithm. Therefore we expect the overall complexity of the set-difference computation to correlate more with the expression (3.30) than with the expression (3.28).

In a special case when we only want to check if \( P \subseteq (\cup_{i=1}^{N_Q} Q_i) \) finding any feasible \( R_j \) in Algorithm 3.3 provides a negative answer and we can abort further search as is shown in the following algorithm.

**Algorithm 3.4 (iscover(\( P, \{ Q_i \}_{i=1}^{N_Q} \))).**

**INPUT** \( P \) and \( Q = \{ Q_i \}_{i=1}^{N_Q} \) satisfying Assumption 3.1

**OUTPUT** \( R \in \{ \text{TRUE}, \text{FALSE} \} \), \( R := \text{iscover}(P, \{ Q_i \}_{i=1}^{N_Q}) \)

1. **LET** \( k = 1, \; R = \text{FALSE} \\
2. **IF** \( \exists x : P^x x < P^c, \; Q_k^x x < Q_k^c \) **THEN** go to Step 4, **ELSE** \( k \leftarrow k + 1 \)
3. **IF** \( k > N_Q \) **THEN EXIT** **ELSE** go to Step 2
4. **FOR** \( j = 1 \) **TO** \( n_{Q_k} \)
5. **IF** \( \exists x : P^x x \leq P^c, \; [Q_k^x]_{(j)} x > [Q_k^c]_{(j)} \)

---

\(^2\)The upper bound is obtained by the hyperplanes in the so-called general position.
4.1.1. **IF** \( k = N_Q \) **THEN EXIT**

4.1.2. **LET** \( \tilde{P} = \mathcal{P} \cap \{ x \mid [Q^T_k]_{(j)} x \geq [Q_k^c]_{(j)} \} \)

4.1.3. **IF** iscover\( (\tilde{P}, \{ Q_i \}_{i=k+1}^{N_Q}) = \text{FALSE} \) **THEN EXIT**

4.2. **LET** \( \mathcal{P} \leftarrow \mathcal{P} \cap \{ x \mid [Q^T_k]_{(j)} x \leq [Q_k^c]_{(j)} \} \)

5. **LET** \( R = \text{TRUE} \) and **EXIT**

Similarly to Algorithm 3.3 the worst case complexity of Algorithm 3.4 is bounded by (3.28) (see also Remark 3.7).
Multi-Parametric Programming

In this chapter, the basics of multi-parametric programming will be summarized. For a good review of standard optimization techniques, we refer the reader to [BV04]. Much of the theoretical results on parametric linear and quadratic programming can be found in [WW69, NGHB74, Gud76, Rob81, BGK+82, KT95]. A good in depth discussion of multi-parametric programs and algorithms for solving them is given in [Bor03] and [Tøn00].

We present an efficient algorithm for multi-parametric quadratic programming (mp-QP). As in [BMDP02], Karush Kuhn Tucker (KKT) conditions are used to characterize the polyhedral critical regions and the corresponding optimal solution. However, here we avoid unnecessary partitioning of the parameter space by using a direct exploration strategy. Starting from the initial critical region we explore its neighborhood by crossing each of the facets and checking if a feasible neighboring critical region exists. The procedure is then repeated in an iterative fashion with all newly generated regions. A similar approach can be utilized for solving multi-parametric linear programs (mp-LPs).

4.1 Basics of Multiparametric programming

We will use the following non-standard definitions:

**Definition 4.1.** An open set $\mathcal{R}$ whose closure $\overline{\mathcal{R}}$ is a polyhedron is called open polyhedron. A “neither open nor closed polyhedron” is a neither open nor closed set $\mathcal{R}$ whose closure $\overline{\mathcal{R}}$ is a polyhedron. A non-Euclidean polyhedron is a set whose closure equals the union of a finite number of polyhedra.

**Definition 4.2.** A function $h : \Theta \to \mathbb{R}^k$, where $\Theta \subseteq \mathbb{R}^s$, is piecewise affine (PWA) if there exists a partition $\mathcal{R}_1, \ldots, \mathcal{R}_N$ of $\Theta$ and $h(\theta) = H^i \theta + k^i$, $\forall \theta \in \mathcal{R}_i$, $i = 1, \ldots, N$.

**Definition 4.3.** A function $h : \Theta \to \mathbb{R}^k$, where $\Theta \subseteq \mathbb{R}^s$, is PWA on polyhedra (PPWA) if there exists a polyhedral partition $\mathcal{R}_1, \ldots, \mathcal{R}_N$ of $\Theta$ and $h(\theta) = H^i \theta + k^i$, $\forall \theta \in \mathcal{R}_i$, $i = 1, \ldots, N$.

Piecewise quadratic functions (PWQ) and piecewise quadratic functions on polyhedra (PPWQ) are defined analogously.

**Definition 4.4.** A function $q : \Theta \to \mathbb{R}$, where $\Theta \subseteq \mathbb{R}^s$, is a multiple quadratic function of multiplicity $d \in \mathbb{N}^+$ if $q(\theta) = \min\{q^1(\theta) := \theta^T Q^1 \theta + l^1 \theta + c^1, \ldots, q^d(\theta) := \theta^T Q^d \theta + l^d \theta + c^d\}$, $Q^i > 0$, $\forall i = 1, \ldots, d$ and $\Theta$ is a convex polyhedron.
4 Multi-Parametric Programming

Definition 4.5. A function \( q : \Theta \to \mathbb{R} \), where \( \Theta \subseteq \mathbb{R}^s \), is a multiple PWQ on polyhedra (multiple PPWQ) if there exists a polyhedral partition \( \mathcal{R}_1, \ldots, \mathcal{R}_N \) of \( \Theta \) and \( q(\theta) = \min \{q_i(\theta) := \theta'Q_i^\dagger \theta + l_i^\dagger \theta + c_i^d, \ldots, q_i^d(\theta) := \theta'Q_i^d \theta + l_i^d \theta + c_i^d\} \), \( \forall \theta \in \mathcal{R}_i \), \( i = 1, \ldots, N \). We define \( d_i \) to be the multiplicity of the function \( q \) in the polyhedron \( \mathcal{R}_i \), and \( d = \sum_{i=1}^{N} d_i \) to be the multiplicity of the function \( q \). Note that \( \Theta \) is not necessarily a convex set.

Consider the nonlinear mathematical program dependent on a parameter vector \( x \) appearing in the cost function and in the constraints

\[
J^*(x) = \inf_{z} f(z, x) \\
\text{subj. to } g(z, x) \leq 0 \\
z \in M,
\]

where \( z \in \mathbb{R}^s \) is the optimization vector, \( x \in \mathbb{R}^n \) is the parameter vector, \( f : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R} \) is the cost function, \( g : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}^m \) are the constraints and \( M \subseteq \mathbb{R}^s \).

A small perturbation of the parameter \( x \) in (4.1) can cause a variety of outcomes, i.e., depending on the properties of the functions \( f \) and \( g \) the solution \( z^*(x) \) may vary smoothly or change abruptly as a function of \( x \). We denote by \( K^* \) the set of feasible parameters, i.e.,

\[
K^* = \{ x \in \mathbb{R}^n \mid \exists z \in M, g(z, x) \leq 0 \},
\]

by \( R : \mathbb{R}^n \to 2^{\mathbb{R}^s} \), where \( 2^{\mathbb{R}^s} \) denotes the set of all subsets of \( \mathbb{R}^s \), the point-to-set map that assigns the set of feasible \( z \)

\[
R(x) = \{ z \in M \mid g(z, x) \leq 0 \}
\]

to a parameter \( x \), by \( J^* : K^* \to \mathbb{R} \cup \{-\infty\} \) the real-valued function which expresses the dependence on \( x \) of the minimum value of the objective function over \( K^* \), i.e.,

\[
J^*(x) = \inf_z \{ f(z, x) \mid x \in K^*, z \in R(x) \},
\]

and by \( Z^* : K^* \to 2^{\mathbb{R}^s} \) the point-to-set map which expresses the dependence on \( x \) of the set of optimizers, i.e., \( Z^*(\bar{x}) = \{ z \in R(\bar{x}) \mid f(z, \bar{x}) = J^*(\bar{x}) \} \) with \( \bar{x} \in K^* \).

\( J^*(x) \) will be referred to as the optimal value function or simply value function, \( Z^*(x) \) will be referred to as the optimal set. We will denote by \( z^* : \mathbb{R}^n \to \mathbb{R}^s \) one of the possible single valued functions that can be extracted from \( Z^* \), \( z^* \) will be called the optimizer function. If \( Z^*(x) \) is a singleton for all \( x \), then \( z^*(x) \) is the only element of \( Z^*(x) \).

Our interest in problem (4.1) will become clear in the next chapters. We can anticipate here that optimal control problems for nonlinear systems can be reformulated as the mathematical program (4.1) where \( z \) is the input sequence to be optimized and \( x \) the initial state of the system. Therefore, the study of the properties of \( J^* \) and \( Z^* \) is fundamental for the study of properties of state-feedback optimal controllers.

Fiacco ([Fia83, Chapter 2]) provides conditions under which the solution of nonlinear multi-parametric programs (4.1) is locally well behaved and establishes properties of the solution as a function of the parameters. In the following we report a basic result [Hog73] which focuses on a restricted set of functions \( f(z, x) \) and \( g(z, x) \):
Theorem 4.1 ([Hog73]). Consider the multi-parametric nonlinear program (4.1). Assume that $M$ is a convex and bounded set in $\mathbb{R}^s$, $f$ is continuous and the components of $g$ are convex on $M \times \mathbb{R}^n$. Then, $J^*(x)$ is continuous at each $x \in \text{relint}(K^*)$.

Unfortunately very little can be said without continuity assumption on $f$ and convexity assumption on $g$. Below we restrict our attention to two special classes of multi-parametric programming.

4.2 Multi-parametric Quadratic Programming

We investigate multi-parametric Quadratic Program (mp-QP) of the following form

$$J^*(x) = \min_{z} \left\{ J(z, x) = \frac{1}{2} z^T z \right\} \text{subj. to} \ C^z z \leq C^x x + C^c,$$  \hspace{1cm} (4.5)

where $z \in \mathbb{R}^{n_z}$ are the optimization variables, $x \in \mathbb{R}^{n_x}$ is the vector of parameters, and $C^z \in \mathbb{R}^{n_z \times n_z}$, $C^x \in \mathbb{R}^{n_x \times n_x}$, $C^c \in \mathbb{R}^{n_c}$ are matrices of polyhedral constraints in $(z, x)$ space. The number of optimization variables, number of parameters and number of constraints are denoted with $n_z$, $n_x$ and $n_c$, respectively. To clarify terminology, in this chapter the term objective function or cost function refers to $J(z, x)$, while the term value function refers to $J^*(x)$.

We denote with $C \subseteq \mathbb{R}^{n_z \times n_x}$ a so-called constraints polyhedron for the problem (4.5)

$$C := \left\{ (z, x) \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_x} \mid [C^z, -C^x] \begin{bmatrix} z \\ x \end{bmatrix} \leq C^c \right\},$$ \hspace{1cm} (4.6)

and with $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ we denote a set of feasible parameters

$$\mathcal{X} := \left\{ x \in \mathbb{R}^{n_x} \mid \exists z \in \mathbb{R}^{n_z} : C^z z \leq C^x x + C^c \right\}.$$  \hspace{1cm} (4.7)

The set $\mathcal{X}$ is just a projection of the constraints polyhedron $C$ to the parameter space $\mathbb{R}^{n_x}$. Since the projection of a polyhedron is also a polyhedron, we know that $\mathcal{X}$ can be described in the following form

$$\mathcal{X} = \left\{ x \in \mathbb{R}^{n_x} \mid H^x x \leq H^c \right\},$$ \hspace{1cm} (4.8)

where $H^x \in \mathbb{R}^{n_H \times n_x}$, $H^c \in \mathbb{R}^{n_H}$, and $n_H$ is the number of halfspaces describing the polyhedron $\mathcal{X}$.

For any given $x \in \mathcal{X}$, $J^*(x)$ denotes the minimum value of the objective function in problem (4.5). The value function $J^* : \mathcal{X} \to \mathbb{R}$ expresses the dependence on $x$ of the minimum value of the objective function over $\mathcal{X}$. The single-valued function $z^* : \mathcal{X} \to \mathbb{R}^{n_z}$ describes, for any fixed $x \in \mathcal{X}$, the optimizer $z^*(x)$ related to $J^*(x)$.

The goal is to determine the feasible region of parameters $\mathcal{X}$ (i.e., to find matrices $H^x$, and $H^c$ in (4.8)) and to find the closed-form expressions of the value function $J^*(\cdot)$ and of the optimizer function $z^*(\cdot)$. 
4.2.1 Translation into the standard mp-QP form

Any problem of the form

\[ V(u, x) = \frac{1}{2} u'Q_{uu}u + x'Q_{xx}x + Q_x'u + Q_x'x + Q_0 \]  

\[ V^*(x) = \min_{u} V(u, x) \quad \text{subj. to} \quad G^u u \leq G^x x + G_c, \]  

where \( u \in \mathbb{R}^{n_u}, Q_{uu} \in \mathbb{R}^{n_u \times n_u}, Q_{xx} \in \mathbb{R}^{n_x \times n_x}, Q_x \in \mathbb{R}^{n_u}, Q_0 \in \mathbb{R}, G^u \in \mathbb{R}^{n_G \times n_u}, G^x \in \mathbb{R}^{n_G \times n_x}, G_c \in \mathbb{R}^{n_G} \) and \( Q_{uu} = Q_{uu}' \succ 0 \), can always be transformed in the mp-QP (4.5) by using the variable substitution

\[ z = Q_{uu}^\frac{1}{2} (u + Q_{uu}^{-1}Q_x'u + Q_{uu}^{-1}Q_x). \]  

By inspection we see that the following equations hold for such a substitution

\[ n_z = n_u, \]  

\[ n_C = n_G, \]  

\[ C^z = G^u Q_{uu}^{-\frac{1}{2}} \]  

\[ C^x = G^x + G^u Q_{uu}^{-1}Q_x', \]  

\[ C_c = G_c + G^u Q_{uu}^{-1}Q_u. \]  

After solving the mp-QP (4.5) we can easily reconstruct the solution to the problem (4.10) by noting that

\[ u = Q_{uu}^{-\frac{1}{2}} z - Q_{uu}^{-1}Q_x'u - Q_{uu}^{-1}Q_u, \]  

\[ V(u, x) = J(z, x) + \Delta(x), \]  

\[ V^*(x) = J^*(x) + \Delta(x), \]  

where

\[ \Delta(x) = \frac{1}{2} x' (Q_{xx} - Q_{xx}Q_{uu}^{-1}Q_{xx}') x + (Q_x' - Q_x'Q_{uu}^{-1}Q_x') x + Q_0 - \frac{1}{2} Q_u Q_{uu}^{-1}Q_u. \]  

4.2.2 Solution of the mp-QP

To simplify the description of the algorithm we will assume that the following condition holds.

**Assumption 4.1.** The constraints polyhedron \( C \) is a full-dimensional and bounded polyhedron, i.e. \( C \) is an \((n_z + n_x)\)-polytope.
Assumption 4.1 is not as restrictive as it may seem at first sight. It is always possible to translate the mp-QP problem in some lower-dimensional \((\tilde{n}_z + \tilde{n}_x)\)-space, \(\tilde{n}_z \leq n_z, \tilde{n}_x \leq n_x\), where the constraints polyhedron \(C\) is full-dimensional, while, with slight modifications, the algorithm presented in the following sections would work even for the unbounded polyhedra. One simple consequence of Assumption 4.1 is that the feasible parameter space \(\mathcal{X}\) is an \(n_x\)-polytope.

**Definition 4.6 (Active constraint).** The \(i\)-th constraint, \(i \in \{1, \ldots, n_C\}\), of the constraints polyhedron \(C\) is said to be *active* at \(x \in \mathcal{X}\) if

\[
C^{z^*}_{(i)}(x) = C^{x}_{(i)}x + C^c_{(i)},
\]

and it is said to be *inactive* at \(x \in \mathcal{X}\) if

\[
C^{z^*}_{(i)}(x) < C^{x}_{(i)}x + C^c_{(i)}.
\]

**Definition 4.7 (Active set).** For any \(x \in \mathcal{X}\) an optimal active set of \(C\)

\[
\mathcal{AC}(x) := \{i \in \{1, \ldots, n_C\} \mid C^{z^*}_{(i)}(x) = C^{x}_{(i)}x + C^c_{(i)}\}
\]

is also called the *set of active constraints*, while an optimal inactive set

\[
\mathcal{IC}(x) := \{i \in \{1, \ldots, n_C\} \mid C^{z^*}_{(i)}(x) < C^{x}_{(i)}x + C^c_{(i)}\}
\]

is also called the *set of inactive constraints*.

It is obvious from Definition 4.7 that

\[
\forall x \in \mathcal{X} : \mathcal{AC}(x) \cup \mathcal{IC}(x) = \{1, \ldots, n_C\}
\]

The multi-parametric analysis uses the concept of *critical region*.

**Definition 4.8 (Critical region).** Let \(\mathcal{A} \subseteq \{1, \ldots, n_C\}\) be a feasible combination of active constraints, i.e., \(\exists \tilde{x} \in \mathcal{X} : \mathcal{AC}(\tilde{x}) = \mathcal{A}\). A (non-empty) set

\[
\mathcal{R}_\mathcal{A} := \{x \in \mathcal{X} \mid \mathcal{AC}(x) = \mathcal{A}\}
\]

is called *critical region* (related to the set of active constraints \(\mathcal{A}\)).

In another words \(\mathcal{R}_\mathcal{A}\) is the set of all parameters \(x\) such that the constraints indexed by \(\mathcal{A}\) are active at the optimum of problem (4.5).

**Definition 4.9 (LICQ).** For a given set of active constraints \(\mathcal{A}\) we say that *Linear Independence Constraint Qualification* (LICQ) holds if the rows of \(C^{z}_{(\mathcal{A})}\) are linearly independent.
Geometric Algorithm for mp-QP

Solving mp-QP (4.5) means determining partition of \(\mathcal{X}\) into critical regions \(\mathcal{R}_{\mathcal{A}_i}\), and finding the expression of the functions \(J^*(\cdot)\) and \(z^*(\cdot)\) for each critical region. In the following we detail the algorithm as it appears in [BMDP02].

First, an initial vector \(x_0\) inside the polyhedral set \(\mathcal{X}\) is chosen such that the QP problem (4.5) is feasible for \(x = x_0\). One simple choice is the \(x\)-component of the center of the largest ball contained in \(\mathcal{C}\) obtained by solving an LP

\[
\begin{align*}
\max_{z, x, \epsilon} & \quad \epsilon \\
\text{subj. to} & \quad C^z_{(i)} z - C^x_{(i)} x + \left\| [C^z_{(i)}, -C^x_{(i)}] \right\|_2 \leq C^c_{(i)}, \quad i = 1, \ldots, n_C 
\end{align*}
\]

In particular, \(x_0\) will be a projection of the Chebyshev center of \(\mathcal{C}\) on \(\mathbb{R}^n\) when the QP problem (4.5) is feasible. If \(\epsilon \leq 0\), then Assumption 4.1 is violated. Otherwise, we fix \(x = x_0\) and solve the QP problem (4.5), in order to obtain the corresponding optimal solution \(z^*_0\). Such a solution is unique (since the objective function is strictly convex), and therefore a set of active constraints \(\mathcal{A}_0\) is uniquely determined out of the constraints in (4.5). We have the following result (cf. [BMDP02]).

**Theorem 4.2.** Consider a combination of active constraints \(\mathcal{A}_0\), and assume that LICQ holds. Then, the optimal \(z^*\) and the associated vector of Lagrange multipliers \(\lambda^*\) are uniquely defined affine functions of \(x\) over the critical region \(\mathcal{R}_{\mathcal{A}_0}\).

**Proof** The first-order Karush-Kuhn-Tucker (KKT) optimality conditions [BSS93] for the mp-QP are given by

\[
\begin{align*}
z^* + (C^z)^T \lambda^* &= 0, \quad \lambda^* \in \mathbb{R}^{n_C}, & (4.23a) \\
\lambda^*_i (C^z_{(i)} z^* - C^x_{(i)} x - C^c_{(i)}) &= 0, \quad i = 1, \ldots, n_C, & (4.23b) \\
\lambda^* &\geq 0, & (4.23c) \\
C^z z^* &\leq C^x x + C^c, & (4.23d)
\end{align*}
\]

From (4.23a) we have

\[
z^* = -(C^z)^T \lambda^*
\]

Let \(\lambda^*_{(I_0)}\) and \(\lambda^*_{(A_0)}\) denote the Lagrange multipliers corresponding to inactive and active constraints, respectively. Substituting the result (4.24) into (4.23b) and knowing that for inactive constraints,

\[
\lambda^*_{(I_0)} = 0
\]

we get

\[
\lambda^*_{(A_0)} = -(C^z_{(A_0)}(C^z_{(A_0)})^T)^{-1}(C^x_{(A_0)} x + C^c_{(A_0)})
\]

where \((C^z_{(A_0)}(C^z_{(A_0)})^T)^{-1}\) exists because the rows of \(C^z_{(A_0)}\) are linearly independent. Thus \(\lambda^*\) is an affine function of \(x\). We can substitute \(\lambda^*_{(A_0)}\) from (4.26) into (4.24) to obtain

\[
z^* = (C^z_{(A_0)}(C^z_{(A_0)})^T)^{-1}(C^x_{(A_0)} x + C^c_{(A_0)})
\]
and note that $z^*$ is also an affine function of $x$.

Substituting expressions (4.25)–(4.27) for $\lambda^*$ and $z^*$ into (4.23c) and (4.23d) we can characterize the set of parameters where those expressions are valid, i.e., we obtain the closure of the critical region $R_{A_0}$

$$
(C_{(A_0)}^c)^{-1}(C_{(A_0)}^x x + C_{(A_0)}^c) \leq 0
$$

(4.28)

$$
C^{x}(C_{(A_0)}^x)^{-1}(C_{(A_0)}^c)^{-1}(C_{(A_0)}^x x + C_{(A_0)}^c) \leq C^{x} x + C^{c}
$$

(4.29)

Since (4.29) is trivially fulfilled for any $C_{(j)}^c$ where $j \in A_0$ we can simplify it by removing those unnecessary rows to get

$$
C_{(I_0)}^{x}(C_{(A_0)}^c)^{-1}(C_{(A_0)}^c)^{-1}(C_{(A_0)}^x x + C_{(A_0)}^c) \leq C_{(I_0)}^{x} x + C_{(I_0)}^{c}
$$

(4.30)

The region $R_{A_0}$ is a polytope in $\mathbb{R}^{n_x}$. Note that the difference between the closure $R_{A_0}$ and the critical region $R_{A_0}$ is just due to the practicality of the computation. If in expression (4.23d), (and consequently in (4.29) and (4.30)) instead of a non-strict inequality " $\leq$ " a strict inequality " $<$ " was used we would obtain the description of $R_{A_0}$. However, since it is easy to extract critical region $R_{A_0}$ once $R_{A_0}$ is computed, we will make no differentiation between the two.

**Remark 4.1.** In a more general context Theorem 4.2 is proven in [BGK+82] (cf. Theorem 5.5.2 and Equation (5.5.28) in [BGK+82]).

**Remark 4.2.** Note that the critical region $R_{A_0}$ is, in general, not closed nor open, since some of its boundaries may be defined with strict inequalities, while others might be given as non-strict inequalities. However, we see from the definition that critical region $R_{A_0}$ is always convex.

A compact representation of polyhedron $R_{A_0}$ is obtained after removing the redundant inequalities from (4.28) and (4.30). Once the critical region $R_{A_0}$ has been defined, the rest of the space $\mathcal{X} \setminus R_{A_0}$ has to be explored and new critical regions generated.

An effective approach for partitioning the rest of the space was proposed in [DP00] and formally proved in [BDMP02].

**Theorem 4.3.** Let $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ be a polyhedron, and $R_0 := \{ x \in \mathcal{X} \mid G^x x \leq G^c \}$ be a polyhedral subset of $\mathcal{X}$, $R_0 \neq \emptyset$, where $G^x \in \mathbb{R}^{n_G \times n_x}$, $G^c \in \mathbb{R}^{n_G}$. Also let

$$
R_i = \left\{ x \in \mathcal{X} \mid \begin{array}{l}
G_{(i)}^x x > G_{(i)}^c
G_{(j)}^x x \leq G_{(j)}^c \forall j \neq i
\end{array} \right\}, i = 1, \ldots, n_G
$$

Then $\{ R_0, R_1, \ldots, R_{n_G} \}$ is a polyhedral partition of $\mathcal{X}$, i.e.,

$$
\bigcup_{i=0}^{n_G} R_i = \mathcal{X}, \quad \text{and} \quad R_i \cap R_j = \emptyset, \ \forall i \neq j, \ i, j \in \{0, \ldots, n_G\}
$$

---

1It may happen that the region $R_{A_0}$ is a lower-dimensional polytope in $\mathbb{R}^{n_x}$.
Proof See [BMDP02]. □

As an illustration for the procedure proposed in Theorem 4.3 consider the two-dimensional case depicted in Figure 4.1(a). Here $\mathcal{X}$ is defined by the inequalities $\{x^-_1 \leq x_1 \leq x^+_1, x^-_2 \leq x_2 \leq x^+_2\}$, and $\mathcal{R}_0$ by the inequalities $\{g_1 \leq 0, \ldots, g_5 \leq 0\}$ where $g_1, \ldots, g_5$ are linear in $x$. The procedure consists of considering one by one the inequalities which define $\mathcal{R}_0$. Considering, for example, the inequality $g_1 \leq 0$, the first set of the rest of the region $\mathcal{X}\setminus \mathcal{R}_0$ is given by $\mathcal{R}_1 = \{g_1 \geq 0, x_1 \geq x^-_1, x^-_2 \leq x_2 \leq x^+_2\}$, which is obtained by reversing the sign of the inequality $g_1 \leq 0$ and removing redundant constraints in $\mathcal{X}$ (see Figure 4.1(b)). Thus, by considering the rest of the inequalities we get the partition of the rest of the parameter space $\mathcal{X}\setminus \mathcal{R}_0 = \bigcup_{i=1}^5 \mathcal{R}_i$, as reported in Figure 4.1(d).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4_1.png}
\caption{Two dimensional example: partition of the rest of the space $\mathcal{X}\setminus \mathcal{R}_0$.}
\end{figure}

A Summary of the mp-QP Algorithm

Based on the above discussion and results, the main steps of the off-line mp-QP solver are outlined in the following algorithm.
Algorithm 4.1 (Multi-parametric Quadratic Program).

**INPUT**  The constraints polyhedron $C$ of Problem (4.5)

**OUTPUT**  Multiparametric solution to Problem (4.5)

1. **LET** $(z_0, x_0, \epsilon)$ be the solution to the LP (4.22);

2. **IF** $\epsilon \leq 0$ **THEN EXIT**; (no full--dimensional critical region $R$ exists)

3. For $x = x_0$, compute the optimal solution $(z^*_0, \lambda^*_0)$ of the QP (4.5);

4. Determine the set of active constraints $A_0$ when $z = z^*_0, x = x_0$, and build $C^c_{(A_0)}, C^x_{(A_0)}, C^c_{(A_0)}$;

5. Determine $\lambda^*_0(x), z^*(x)$ from (4.26) and (4.27);

6. Characterize the critical region $R_{A_0}$ from (4.28) and (4.30);

7. Partition the rest of the parameter space $X \setminus R_{A_0}$ as in Theorem 4.3;

8. For each new sub--region $R_i$ execute the same Algorithm 4.1;

Algorithm 4.1 explores the set $X$ of parameters recursively: Partition the rest of the region as in Theorem 4.3 into polyhedral sets $R_i$, use the same method to partition each set $R_i$ further, and so on. This can be represented as a search tree, with a maximum depth equal to the number of combinations of active constraints.

Note that in step 7 of Algorithm 4.1 we assume that the feasible parameter space $X$ is known. It may be computed as a projection of $C$. However, in practice, we usually avoid computation of $X$ at each recursive call of Algorithm 4.1 by storing $X$ (or some outer bounds for it) and passing it to the Algorithm 4.1 as an additional input. Similarly in step 8, instead of adding the description of regions $R_i$ to the constraint matrix $C$, we just pass $R_i$ as a feasible parameter space.

**Remark 4.3** (Degeneracy). Algorithm 4.1 is sketched under the assumption that the LICQ holds. When this assumption is violated we say that the mp-QP is degenerate and KKT conditions do not lead directly to the description of the critical regions. There are several approaches in solving the degenerate mp-QP that involve either projection of the higher dimensional polyhedra or selection of the appropriate subset of active constraints. For a detailed description of how to handle degeneracy in mp-QP see [Bor03] and [TJB01a].

4.2.3 Direct exploration of the parameter space

Before outlining a novel algorithm for solving mp-QP via direct exploration of the parameter space, we show some of the properties of the mp-QP solution.

**Theorem 4.4.** Consider the multi-parametric quadratic program (4.5). The set of feasible parameters $X$ is convex polyhedron. The optimizer $z^*(x) : X \rightarrow \mathbb{R}^{n_z}$ is continuous and
piecewise affine on polyhedra, in particular it is affine in each critical region, and the optimal solution $J^*(x) : \mathcal{X} \to \mathbb{R}$ is continuous, convex and piecewise quadratic on polyhedra.

**Proof** The set of feasible parameters $\mathcal{X}$ is a projection of the constraint polyhedron $\mathcal{C}$. Hence, $\mathcal{X}$ is a convex polyhedron. To prove convexity of $J^*(x)$ consider two feasible points $x_1, x_2 \in \mathcal{X}$. Let $z_1^*$, and $z_2^*$ be corresponding optimizers, and let $x = \alpha x_1 + (1-\alpha)x_2$, $z = \alpha z_1^* + (1-\alpha)z_2^*$, $0 \leq \alpha \leq 1$. Note that $(z_3, x_3) \in \mathcal{C}$ and $J(z, x) = \frac{1}{2}z'z$ is convex in $z$. We have $J^*(x_3) := J(z_3, x_3) \leq J(z_3, x) := \frac{1}{2}z_3'z_3 \leq \frac{1}{2}\{\alpha (z_1^*)'z_1^* + (1-\alpha)(z_2^*)'z_2^*\} = \alpha J^*(x_1) + (1-\alpha)J^*(x_2)$. Thus, $J^*(x)$ is convex. Since $J^*(x)$ is convex over a convex set $\mathcal{X}$ it is continuous at every interior point of $\mathcal{X}$. The technicality of proving that $J^*(x)$ is also continuous on the border of $\mathcal{X}$ is based on the fact that $\mathcal{C}$ is closed, and both constraints and $J(z, x)$ are continuous functions, cf. [MR62].

As shown in Theorem 4.2 the optimizer $z^*(x)$ is given as an affine function over a critical region. Hence $z^*(x)$ is continuous in the interior of every critical region. Since $J(z, x) = \frac{1}{2}z'z$ is strictly convex in $z$ it follows that the optimizer $z^*(x)$ is unique. This, together with $J^*(x) = J(z^*(x), x)$ being continuous over whole $\mathcal{X}$ implies that $z^*(x)$ has to be continuous on the border of two critical regions as well as on the boundary of $\mathcal{X}$. Thus, $z^*(x)$ is continuous piecewise affine function on polyhedra. $\square$

**Remark 4.4.** The results of Theorem 4.4 can be proven by combining Theorem 5.5.1, Theorem 5.5.2 and the discussion in Section 5.5 in [BGK+82].

**Lemma 4.1.** Let $\mathcal{R}_A \subset \mathbb{R}^{n_x}$ be a non-empty critical region associated to the set of active constraints $\mathcal{A}$, and let the optimizer $z^*(x)$ be given as a function $z^*_A : x \mapsto K^x x + K^c$, $\forall x \in \mathcal{R}_A$. Then

$$\mathcal{AC}(\bar{x}) \supseteq \mathcal{A}, \quad \forall \bar{x} \in \text{aff}(\mathcal{R}_A) : z^*(\bar{x}) = K^x \bar{x} + K^c \quad (4.31)$$

where $\text{aff}(\mathcal{R}_A)$ denotes an affine subspace spanned by the set $\mathcal{R}_A$.

Although very simple, Lemma 4.1 is instrumental for the proofs and observations that follow. It claims that the constraints active in $\mathcal{R}_A$ are also active in all points $\bar{x}$ that lie in the affine subspace spanned with $\mathcal{R}_A$ if an optimizer value can be computed with the optimizer expression from $\mathcal{R}_A$. Note that we do not require the actual expression for the computation of an optimizer in $\bar{x}$ to be the same as for $\mathcal{R}_A$ ($\bar{x}$ does not have to be in $\mathcal{R}_A$). Nevertheless, as long as the computed optimizer values are the same, Lemma 4.1 holds.

**Proof** Starting from a set of active constraints $\mathcal{A}$ of the critical region $\mathcal{R}_A$

$$C^z_{(A)} z^*(x) = C^c_{(A)} x + C^c_{(A)}$$

and combining it with the expressions for the optimizer $z^*(x) = z^*_A(x) = K^x x + K^c$, we obtain the expression for the affine subspace to which $\mathcal{R}_A$ belongs to

$$\text{aff}(\mathcal{R}_A) = \{ x \in \mathbb{R}^{n_x} | (C^z_{(A)} K^x - C^c_{(A)} x = C^c_{(A)} - C^z_{(A)} K^c) \} \quad (4.32)$$

Therefore, for any point $\bar{x} \in \text{aff}(\mathcal{R}_A)$ we have

$$(C^z_{(A)} K^x - C^c_{(A)}) \bar{x} = C^c_{(A)} - C^z_{(A)} K^c$$

Therefore, for any point $\bar{x} \in \text{aff}(\mathcal{R}_A)$ we have
and if for such a point \( z^*(\bar{x}) = K^\bar{x} + K^c \) (as is assumed in Lemma 4.1) we get

\[
C^e_\mathcal{A} z^*(\bar{x}) = C^e_\mathcal{A} \bar{x} + C^e_\mathcal{A}
\]

i.e., constraints indexed with \( \mathcal{A} \) are also active in \( \bar{x} \).

Note that for the fully dimensional critical region we have \( \text{aff}(\mathcal{R}_\mathcal{A}) = \mathbb{R}^n \). Let’s see some of the interesting properties of the polyhedral partition and characteristics of the “neighboring” critical regions we can deduce from Lemma 4.1.

**Corollary 4.1.** Let \( \mathcal{R}_\mathcal{A}_i, i = 1, \ldots, m \) be non-empty critical regions with associated sets of active constraints \( \mathcal{A}_i, i = 1, \ldots, m \). Let \( \mathcal{R} := \bigcap_{i=1}^n \mathcal{R}_\mathcal{A}_i, \mathcal{R} \neq \emptyset \), then

\[
\mathcal{A} \mathcal{C}(x) \supseteq \bigcup_{i=1}^m \mathcal{A}_i, \quad \forall x \in \mathcal{R} \tag{4.33}
\]

**Proof** By definition \( \mathcal{R} \subseteq \text{aff}(\mathcal{R}_\mathcal{A}_i) \). From the continuity of the optimizer (Theorem 4.4) we see that the expression for the computation of an optimizer in \( \mathcal{R}_\mathcal{A}_1 \) gives the correct optimizer value for all points in \( \mathcal{R} \). Hence, from Lemma 4.1, we have \( \mathcal{A}(x) \supseteq \mathcal{A}_1, \forall x \in \mathcal{R} \). Since the same reasoning holds for all critical regions \( \mathcal{R}_\mathcal{A}_i, i = 1, \ldots, m \), we have completed the proof. □

In other words, if the intersection of the (closure) of two or more critical regions is non-empty then all constraints that are active in the regions that create an intersection are also active in the intersection. For example, if two “neighboring” regions “share” a facet, the set of active constraints on that facet includes sets of active constraints of both neighbors.

**Corollary 4.2.** Let \( \mathcal{R}_\mathcal{A} \) be a full dimensional critical region with associated set of active constraints \( \mathcal{A} \). Let \( \mathcal{F}_i \) be one facet of a closure \( \mathcal{R}_\mathcal{A} \) of such a critical region. Then

\[
\exists! \mathcal{A}_i \supseteq \mathcal{A} : \mathcal{A} \mathcal{C}(x) = \mathcal{A}_i, \forall x \in \text{relint}(\mathcal{F}_i) \tag{4.34}
\]

or, equivalently,

\[
\exists! \mathcal{A}_i \supseteq \mathcal{A} : \text{relint}(\mathcal{F}_i) \subseteq \mathcal{R}_\mathcal{A}_i \tag{4.35}
\]

What Corollary 4.2 says is that relative interior of a facet of a closure of a full dimensional critical region can belong to one and only one critical region.

**Proof** We will prove this corollary by contradiction. Assume that \( \text{relint}(\mathcal{F}_i) \) belongs to more than one critical region. Let \( \mathcal{R}_\mathcal{A}_1 \) and \( \mathcal{R}_\mathcal{A}_2 \) be two such critical regions that have the same dimension as \( \mathcal{F}_i \), i.e., \( \text{dim}(\mathcal{F}_i) = \text{dim}(\mathcal{R}_\mathcal{A}_1) = \text{dim}(\mathcal{R}_\mathcal{A}_2) = n_x - 1 \). Obviously \( \mathcal{R}_\mathcal{A}_1 \) and \( \mathcal{R}_\mathcal{A}_2 \) lie in the same affine subspace \( \text{aff}(\mathcal{F}_i) \). From the continuity of the optimizer in \( \mathcal{R}_\mathcal{A} \), \( \mathcal{R}_\mathcal{A}_1 \) and \( \mathcal{R}_\mathcal{A}_2 \) (Theorem 4.4) we see that expression for the optimizer used in \( \mathcal{R}_\mathcal{A}_1 \) can be used in \( \mathcal{R}_\mathcal{A}_2 \) and vice versa. Hence, by invoking Lemma 4.1, we see that all constraints active in \( \mathcal{R}_\mathcal{A}_1 \) are also active in \( \mathcal{R}_\mathcal{A}_2 \) and vice versa. Therefore, both regions have the same set of active constraints. This, by the definition of a critical region, implies that \( \mathcal{R}_\mathcal{A}_1 = \mathcal{R}_\mathcal{A}_2 \), i.e. a contradiction. □
**Remark 4.5.** Note that in the proof of Corollary 4.2 we refer to the relative interior of a facet \( \text{relint}(\mathcal{F}_i) \) and not to \( \overline{\mathcal{F}}_i \) (a closure of \( \mathcal{F}_i \)). We do this in order to exclude potential critical regions that have dimension less than \( n_x - 1 \) which may appear on the boundary of \( \overline{\mathcal{F}}_i \). In such a case we would not be able to use Lemma 4.1 in both directions since not all critical regions are in the same affine subspace.

**Remark 4.6.** Corollary 4.2 also shows that there are no critical regions in \( \text{relint}(\mathcal{F}_i) \) of dimension less than \( n_x - 1 \). This follows from the convexity of critical regions as observed in Remark 4.2, since there can be no gaps inside of a critical region.

**Direct exploration of the neighborhood**

**Assumption 4.2 (Facet-to-Facet property).** We say for the multi-parametric quadratic program (4.5) that the facet-to-facet property holds if for every two neighboring critical regions \( \mathcal{P}_i \) and \( \mathcal{P}_j \) the intersection of their closures \( \mathcal{F} = \overline{\mathcal{P}}_i \cap \overline{\mathcal{P}}_j \) is a facet of both \( \overline{\mathcal{P}}_i \) and \( \overline{\mathcal{P}}_j \).

Under Assumption 4.2 the new mp-QP algorithm that directly explores the neighborhood of a critical region can be described as follows.

**Algorithm 4.2 (mp-QP with a direct exploration of the neighborhood).**

Step 7 and step 8 of Algorithm 4.1 are replaced with the following procedure (illustrated in Figure 4.2). For each facet \( \mathcal{F}_i \) a new parameter \( x^i_\epsilon \) is generated, by moving from the center of the facet in the direction of the normal to the facet by a small step. If the parameter \( x^i_\epsilon \) is infeasible or is contained in a critical region already stored, then the exploration in the direction of \( \mathcal{F}_i \) stops. Otherwise, the set of active constraints corresponding to the critical region sharing the facet \( \mathcal{F}_i \) with the region \( \mathcal{R} \) is found by solving a QP for the new parameter \( x^i_\epsilon \).

Algorithm 4.2 was developed having in mind its practicality. However, as we will show in this section, although we work only with the closure of the critical regions and we do not construct any lower dimensional critical regions, the final solution is solving the original problem. Algorithm 4.2 relies on several known properties of the solution to the multi-parametric quadratic program that follow directly from Theorem 4.4:

- The solution of the mp-QP (optimizer as a function of the parameter) is continuous and unique. Consequently, to describe the solution it is sufficient to consider only the closure of each of the fully dimensional critical regions. Any lower dimensional critical region is just a subset of some fully dimensional region, and its solution already exists.

- The feasible space of the mp-QP is a polyhedron and it is partitioned into a finite number of polyhedral critical regions. In other words, critical regions can be considered as nodes of a finite, fully connected graph. There are no isolated regions that could not be reached by starting from any region and going from one neighbor to another neighbor. Thus we can explore the feasible space starting from anywhere, and the algorithm will terminate in finite time. Furthermore, every facet of a critical region that does not have a neighbor associated with it is also a border of the feasible space.
Figure 4.2: Two dimensional example: mp-QP algorithm that utilizes exploration of the parameter space by exploring the neighborhood.
A reasonable question is “how restrictive is Assumption 4.2?” A very recent report by Spjøtvold et.al. [Spj05] gives an example of a strictly convex, non-degenerate mp-QP that does not satisfy Assumption 4.2. When facet-to-facet property does not hold it may happen that the exploration procedure of Algorithm 4.2 does not cover the whole feasible space $\mathcal{X}$ (holes could exist in the final partition). However, one should also point out that this is very unlikely. Firstly because Algorithm 4.2 is very robust with respect to the generation of critical regions, regardless of Assumption 4.2. Namely, every critical region can be “visited”, and therefore generated, from as many sides as is the number of its facets. And secondly, Assumption 4.2 is most of the time fulfilled. In fact, for a long time Assumption 4.2 was considered as a conjecture since in all extensive simulations and practical problems we have solved we have never observed the violation of that property. One possible way of ensuring that Algorithm 4.2 works in all cases is to compute a set difference between the final partition and the feasible space $\mathcal{X}$ (by using Algorithm 3.2). If that difference is empty we have a guarantee that the solution is correct, otherwise we would treat any polytopes generated by the set difference as the initial places for continuing the exploration with Algorithm 4.2.

### 4.3 Multi-parametric Linear Programming

Consider the multi-parametric linear program (mp-LP) of the following form

$$J^*(x) = \min_z \{ J(z, x) = f'z \} \quad \text{subj. to} \quad C^z z \leq C^x x + C^c,$$

where $z \in \mathbb{R}^{n_z}$ are the optimization variables, $x \in \mathbb{R}^{n_x}$ is the vector of parameters, $C^z \in \mathbb{R}^{n_C \times n_z}$, $C^x \in \mathbb{R}^{n_C \times n_x}$, $C^c \in \mathbb{R}^{n_C}$ are matrices of polyhedral constraints in $(z, x)$ space and $f \in \mathbb{R}^{n_z}$. The number of optimization variables, number of parameters and number of constraints are denoted with $n_z$, $n_x$ and $n_C$, respectively.

There are many LP specific issues (e.g., possible existence of the primal and/or dual degenerate critical regions) that have to be taken into account when solving (4.36). We refer the reader to [Bor03] where an effective geometric algorithm for solving mp-LP is reported. The direct exploration of the parameter space described previously for the mp-QP algorithm can be easily extended to the mp-LP algorithm of [Bor03].

We give the following theorem on the properties of the solution to the mp-LP.

**Theorem 4.5.** Consider the multi-parametric linear program (4.36). The set of feasible parameters $\mathcal{X}$ is a closed polyhedron in $\mathbb{R}^{n_x}$. The function $J^*(x)$ is convex and piecewise affine over $\mathcal{X}$.

If the optimizer $z^*(x) : \mathcal{X} \to \mathbb{R}^{n_z}$ is unique for all $x \in \mathcal{X}$ then the optimizer function $z^*(x) : \mathcal{X} \to \mathbb{R}^{n_z}$ is continuous and piecewise affine. Otherwise it is always possible to define a continuous and piecewise affine optimizer function $z^*(x)$ for all $\mathcal{X}$.

**Proof** See [Gal95, p. 180].
Part II

OPTIMAL CONTROL
Constrained Finite Time Optimal Control of Linear Systems

We consider constrained finite-time optimal control problems for discrete-time linear time-invariant systems with constraints on inputs and outputs based on linear and quadratic performance indices. The solution to such problems is a time-varying piecewise affine (PWA) state-feedback law and it can be computed by means of multi-parametric programming [BMDP02, BBM00a].

By exploiting the properties of the value function and the piecewise affine optimal control law of the constrained finite time optimal control (CFTOC), we propose two new algorithms that avoid storing the polyhedral regions. The new algorithms significantly reduce the online storage demands and computational complexity during evaluation of the PWA feedback control law resulting from the CFTOC.

5.1 Introduction

Recently in [BMDP02, BBM00a] the authors have shown how to compute the solution to the constrained finite-time optimal control (CFTOC) problem as a piecewise affine state-feedback (PWA) law. Such a law is computed off-line by using a multi-parametric programming solver [BMDP02, BBM00d], which divides the state space into polyhedral regions, and for each region determines the linear gain and offset which produces the optimal control action.

This method reveals its effectiveness when a receding horizon control (RHC) strategy is used [Mac02, MRRS00]. RHC requires to solve at each sampling time an open-loop CFTOC problem. 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. Having a precomputed solution as an explicit piecewise affine function of the state vector reduces the on-line computation of the RHC control law to a function evaluation, thus avoiding the on-line solution of a quadratic or linear program.

The only drawback of such a PWA feedback control law is that the number of polyhedral regions could grow dramatically with the number of constraints in the optimal control problem. In this chapter we focus on efficient on-line methods for the evaluation of such a piecewise affine control law. The simplest algorithm would require: (i) the storage of the list of polyhedral regions and of the corresponding affine control laws, (ii) a sequential search
through the list of polyhedra for the $i$-th polyhedron that contains the current state in order to implement the $i$-th control law. By exploiting the properties of the value function and the optimal control law, for CFTOC problems based on linear programming (LP) and quadratic programming (QP), we propose two new algorithms that avoid storing the polyhedral regions. The new algorithms significantly reduce the on-line storage demands and computational complexity during evaluation of the explicit solution of the CFTOC problem. Recently the same problem has been approached in a different manner in [TJB03].

A number of contributions have proposed fast algorithms for the solution of constrained predictive control problems; these algorithms pursue the same goal of this chapter, namely to reduce the on-line computational burden in RHC, but from a different perspective, that consists of using fast LP or QP solvers (in place of a general-purpose solver) tailored to the special dynamic structure of the underlying optimal control problem [MKR00]. In this respect, the "fast algorithm" approach might be competitive with the "explicit solution approach" of this thesis. However, a comparison between these two approaches is outside the scope of this thesis.

The chapter is organized as follows. For discrete-time linear time-invariant systems the basics of CFTOC problems and of RHC are summarized in Section 5.2. In Section 5.3 for LP-based and QP-based optimal control we present two new algorithms to evaluate on-line explicit optimal control laws and we compare their complexity in terms of time and storage against the simplest algorithm mentioned above. Finally, in Section 5.4 an example is given that confirms the efficiency of the new methods.

### 5.2 CFTOC, RHC and their State-feedback PWA Solution

#### 5.2.1 CFTOC problem formulation

Consider the discrete-time linear time-invariant system

$$x(t + 1) = Ax(t) + Bu(t)$$

subject to the constraints

$$E^x x(t) + E^u u(t) \leq E$$

at all time instants $t \geq 0$.

In (5.1)-(5.2), $n_x \in \mathbb{N}$, $n_u \in \mathbb{N}$ and $n_E \in \mathbb{N}$ are the number of states, inputs and constraints respectively, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_x}$ is the input vector, $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $E^x \in \mathbb{R}^{n_E \times n_x}$, $E^u \in \mathbb{R}^{n_E \times n_u}$, $E \in \mathbb{R}^{n_E}$, the pair $(A, B)$ is stabilizable, and the vector inequality (5.2) is considered elementwise.

Let $x_0 = x(0)$ be the initial state and consider the constrained finite-time optimal control problem

$$J^*(x_0) := \min_U J(x_0, U)$$

subject to

$$\begin{align*}
    x_{k+1} &= Ax_k + Bu_k, & k \geq 0, \\
    E^x x_k + E^u u_k &\leq E, & k = 0, \ldots, N - 1,
\end{align*}$$

(5.3)
where \( N \in \mathbb{N} \) is the horizon length, \( U := [u'_0, \ldots, u'_{N-1}]' \in \mathbb{R}^{n_u N} \) is the optimization vector, \( x_i \) denotes the state at time \( i \) if the initial state is \( x_0 \) and the control sequence \( \{u_0, \ldots, u_{i-1}\} \) is applied to the system (5.1), \( J^* : \mathbb{R}^{n_x} \to \mathbb{R} \) is the value function, and the cost function \( J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u N} \to \mathbb{R} \) is given either as a linear function (i.e., sum of linear norms)

\[
J(x_0, U) = \|Q^x x_N\|_\ell + \sum_{k=0}^{N-1} \|Q^x x_k\|_\ell + \|Q^u u_k\|_\ell, \quad \ell \in \{1, \infty\}, \tag{5.4a}
\]

or as a quadratic function

\[
J(x_0, U) = x'_N Q^x x_N + \sum_{k=0}^{N-1} x'_k Q^x x_k + u'_k Q^u u_k. \tag{5.4b}
\]

In the following, we will assume that that \( Q^x, Q^u, Q^{x_N} \) are full column rank matrices when the cost function (5.4a) is used, and that \( Q^x = (Q^x)' \succeq 0, Q^u = (Q^u)' \succ 0, Q^{x_N} \succeq 0 \), when the cost function (5.4b) is used.

The optimization problem (5.3) can be translated into a linear program (LP) when the linear cost function (5.4a) is used [BBM00a] or into a quadratic program (QP) when the quadratic cost function (5.4b) is used [BMDP02]. The optimizer

\[
U^* = [u'_0, \ldots, (u'_{N-1})']'
\]

of problem (5.3)–(5.4) is a function of the initial state \( x_0 \). It can be computed by solving an LP or a QP once \( x_0 \) is fixed or it can be computed explicitly for all \( x_0 \) within a given range of values as explained in the following.

### 5.2.2 RHC strategy

Consider the problem of regulating to the origin the discrete-time linear time-invariant system (5.1) while fulfilling the constraints (5.2). The solution \( U^* \) to CFTOC problem (5.3)–(5.4) is an open-loop optimal control trajectory over a finite horizon. A receding horizon control strategy employs it to obtain a feedback control law in the following way: Assume that a full measurement of the state \( x(t) \) is available at the current time \( t \geq 0 \). Then, the CFTOC problem (5.3)–(5.4) is solved at each time \( t \) for \( x_0 = x(t) \), and

\[
u(t) = u'_0\]

is applied as an input to system (5.1).

The two main issues regarding this policy are the feasibility of the optimization problem (5.3)–(5.4) for all \( t \geq 0 \) and the stability of the resulting closed-loop system. We will assume that the matrices \( Q^x, Q^u, Q^{x_N} \), the horizon length \( N \) and the constraints in (5.3) have been chosen to guarantee the stability and the feasibility of RHC control law (5.3)–(5.5). For a detailed discussion see, e.g., [SD87, CM96, SR98, BMDP02, BBM00a, Mac02].

### 5.2.3 Solution of CFTOC, linear cost Case

Consider the problem (5.3) with the linear cost function (5.4a) and \( \ell = \infty \). Using a standard transformation [BBM00a], introducing the vector \( v := [u'_0, \ldots, u'_{N-1}, \varepsilon_1^x, \ldots, \varepsilon_N^x, \varepsilon_1^u, \ldots, \varepsilon_N^u]' \in \mathbb{R}^{n_v} \)
\[ \mathbb{R}^{n_v}, n_u := (n_u + 2)N, \varepsilon^x_k \geq \|Q^x x_k\|_\infty, \varepsilon^x_N \geq \|Q^{x_N} x_N\|_\infty, \varepsilon^u_k \geq \|Q^u u_k\|_\infty, \]
and substituting
\[ x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \] in (5.3)–(5.4), this can be rewritten as the linear program
\[
J^*(x) := \min_{v} c^t v \\
\text{subj. to } L^tv \leq L^x x
\] (5.6)
where \( x = x_0 \), and matrices \( c \in \mathbb{R}^{n_v}, L^v \in \mathbb{R}^{n_L \times n_v}, L^x \in \mathbb{R}^{n_L \times n_x}, L \in \mathbb{R}^{n_L} \) are easily obtained from \( Q^x, Q^u, Q^{x_N} \) and (5.3)–(5.4), as explained in [BBM00a].

Because the problem depends on \( x \) the implementation of RHC can be performed in two different ways: solve the LP (5.6) on-line at each time step for a given \( x \) or solve (5.6) off-line for all \( x \) within a given range of values, i.e., by considering (5.6) as a multi-parametric Linear Program (mp-LP) [Gal95].

Solving an mp-LP means computing the optimizer \( v^*(x) \) and the value function \( J^*(x) \) for all possible vectors \( x \) in a given set \( \mathcal{X} \). The solution to mp-LP problems can be simply approached by exploiting the properties of the primal and dual optimality conditions as proposed in [BBM00d, Gal95].

Once the multi-parametric problem (5.6) has been solved off-line for a polyhedral set \( \mathcal{X} \subseteq \mathbb{R}^{n_x} \) of states, the explicit solution \( v^*(x) \) of CFTOC problem (5.6) is available as a piecewise affine function of \( x \), and the receding horizon controller (5.3)–(5.5) is also available explicitly, as the optimal input \( u(t) \) consists simply of \( n_u \) components of \( v^*(x(t)) \)
\[
u(t) = [I_{n_u} \ 0 \ \ldots \ 0] v^*(x(t)).
\] (5.7)

In [Gal95] the following results about the properties of the solution are proved:

**Theorem 5.1.** Consider the multi-parametric linear program (5.6). Then the set of feasible parameters \( \mathcal{X}_f \) is convex. If the optimizer \( v^*(x) \) is unique for all \( x \in \mathcal{X}_f \), then the optimizer function \( v^*: \mathcal{X}_f \to \mathbb{R}^{n_v} \) is continuous and piecewise affine. Otherwise, it is always possible to define a continuous and piecewise affine optimizer function \( v^*(x) \) for all \( x \in \mathcal{X}_f \).

**Corollary 5.1.** The RHC (5.7), defined by the optimization problem (5.3), (5.4a) and (5.5), is a continuous and piecewise affine function, \( u: \mathbb{R}^{n_x} \to \mathbb{R}^{n_u} \), and has the form
\[
u(x) = F_i x + G_i, \ \forall x \in \mathcal{P}_i, \ i = 1, \ldots, N_p,
\] (5.8)
where \( F_i \in \mathbb{R}^{n_u \times n_x}, G_i \in \mathbb{R}^{n_u}, \) and \( \mathcal{P}_i = \{x \in \mathbb{R}^{n_x} \mid P^x_i x \leq P^x_i \}, P^x_i \in \mathbb{R}^{n_x \times n_x}, P_i \in \mathbb{R}^{n_i}, \ i = 1, \ldots, N_p \) is a polyhedral partition of \( \mathcal{X}_f (\cup_{i=1}^{N_p} \mathcal{P}_i = \mathcal{X}_f, \mathcal{P}_i, \mathcal{P}_j \) have disjoint interiors \( \forall i \neq j \).

In the following we will denote with \( N_H \) the total number of halfspaces defining the polyhedral partition of \( \mathcal{X}_f \)
\[
N_H := \sum_{i=1}^{N_p} p_i.
\] (5.9)

\(^1\)The same holds for \( \ell = 1 \) with a different optimizer vector [BBM00a].
Remark 5.1. Typically the total number of halfspaces defining polyhedral partition of feasible set $\mathcal{X}_f$ is much bigger than the number of polyhedral regions in it, i.e., $N_H \gg N_P$. The reasoning is the following. Assume, as is the case in practical applications, that all $P_i$ are bounded. Since the smallest number of halfspaces defining a bounded polyhedron in $\mathbb{R}^{n_x}$ is $n_x + 1$ (achieved by a simplex) we have $N_H \geq (n_x + 1)N_P$.

### 5.2.4 Solution of CFTOC, quadratic cost Case

Consider the problem (5.3) with the quadratic cost function (5.4b). By substituting $x_k = A^kx_0 + \sum_{j=0}^{k-1} A^jBu_{k-1-j}$ in (5.3)–(5.4), this can be rewritten as the quadratic program

$$J^*(x) = \frac{1}{2}x'Yx + \min_U \frac{1}{2}U'HU + x'FU$$

subj. to $M^U \leq M + M^x x$

(5.10)

where $x = x_0$, the column vector $U := [u'_0, \ldots, u'_{N-1}]' \in \mathbb{R}^{n_U}$, $n_U := n_uN$, is the optimization vector, $H = H' > 0$, and $H, F, Y, M^U, M^x, M$ are easily obtained from $Q^x, Q^u, Q^{xN}$ and (5.3)–(5.4) (see [BMDP02] for details).

As in the linear cost case, because the problem depends on $x$ the implementation of RHC can be performed either by solving the QP (5.10) on-line or, as shown in [BMDP02, TJB01b], by solving problem (5.10) off-line for all $x$ within a given range of values, i.e., by considering (5.10) as a multi-parametric Quadratic Program (mp-QP).

Once the multi-parametric problem (5.10) is solved off-line, i.e., the solution $U^*(x)$ of the CFTOC problem (5.10) is found, the state-feedback PWA RHC law is simply

$$u(t) = [I_{n_u} 0 \ldots 0]U^*(x(t)).$$

(5.11)

In [BMDP02] the authors prove\(^2\) the following results about the properties of the solution.

**Theorem 5.2.** Consider the multi-parametric quadratic program (5.10) and let $H > 0$. Then the set of feasible parameters $\mathcal{X}_f$ is convex, the optimizer $U^* : \mathcal{X}_f \to \mathbb{R}^s$ is continuous and piecewise affine, and the optimal solution $J^* : \mathcal{X}_f \to \mathbb{R}$ is continuous, convex and piecewise quadratic.

**Corollary 5.2.** The RHC control law (5.11), defined by the optimization problem (5.3), (5.4b) and (5.5), is continuous and piecewise affine, and has the form (5.8).

\(^2\)See Remark 4.1 and Remark 4.4.
5.3 Efficient on-line Algorithms

The on-line implementation of the control law (5.8) is simply executed according to the following steps:

Algorithm 5.1.

1. Measure the current state $x(t)$
2. Search for the $i$-th polyhedron that contains $x(t)$, $(P^x_i x(t) \leq P_i)$
3. Implement the $i$-th control law $(u(t) = F_i x(t) + G_i)$

In Algorithm 5.1, step (2) is critical and it is the only step whose efficiency can be improved. In the following for any matrix (vector) $\Theta$ with $\Theta(j)$ we denote the $j$-th row (element) of $\Theta$. A simple implementation of step (2) would consist of searching for the polyhedral region that contains the state $x(t)$ as in the following algorithm:

Algorithm 5.2.

1. $i = 0$, notfound=TRUE
2. WHILE $i \leq N_P$ AND notfound
   2.1. $j = 0$, feasible=TRUE
   2.2. WHILE $j \leq p_i$ AND feasible
      2.2.1. IF $(P^x_i(j)x(t) > (P_i(j))$ THEN feasible=FALSE
   2.3. END
   2.4. IF feasible THEN notfound=FALSE
3. END

Recalling the expression (5.9) for $N_H$ (the total number of halfspaces defining the polyhedral partition of the feasible set $X_f$), it is easy to see that Algorithm 5.2 requires $(n_x + 1)N_H$ real numbers to store all polyhedra $P_i$, and in the worst case (when the state is contained in the last region of the list) Algorithm 5.2 will give a solution after $n_x N_H$ multiplications, $(n_x - 1)N_H$ sums and $N_H$ comparisons.

Remark 5.2. In the algorithms presented in the following sections we implicitly assume that $x(t)$ belongs to the feasible set $X_f$. If this (reasonable) assumption does not hold we should include set of boundaries of feasible parameter space $X_f$ and we should (before using any of proposed algorithms) first check if the point $x(t)$ is inside the boundaries of $X_f$. Note that such a step is not needed for Algorithm 5.2 since there we automatically detect if the point $x(t)$ is outside of the feasible set $X_f$.

By using the properties of the value function, we will show how Algorithm 5.2 can be replaced by more efficient algorithms that have less computational complexity and that avoid storing the polyhedral regions $P_i$, $i = 1, \ldots, N_P$, therefore reducing the storage demand significantly.

In the following we will distinguish between optimal control based on LP and optimal control based on QP.
Table 5.1: Complexity comparison of Algorithm 5.2 and Algorithm 5.3.

<table>
<thead>
<tr>
<th></th>
<th>Algorithm 5.2</th>
<th>Algorithm 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage demand (real numbers)</td>
<td>$(n_x + 1)N_H$</td>
<td>$(n_x + 1)N_P$</td>
</tr>
<tr>
<td>Number of flops (worst case)</td>
<td>$2n_xN_H$</td>
<td>$2n_xN_P$</td>
</tr>
</tbody>
</table>

5.3.1 Efficient Implementation, linear cost Case

From Theorem 5.1, the value function $J^*(x)$ corresponding to the solution of the CFTOC problem (5.3) with the linear cost (5.4a) is convex and PWA:

$$J^*(x) = T_i x + V_i, \quad \forall x \in \mathcal{P}_i, \quad i = 1, \ldots, N_P,$$

where $T_i \in \mathbb{R}^{n_x}$, $V_i \in \mathbb{R}$.

By exploiting the convexity of the value function the storage of the polyhedral regions $\mathcal{P}_i$ can be avoided. From the equivalence of the representations of PWA convex functions [Sch87], the function $J^*(x)$ in equation (5.12) can be represented alternatively as

$$J^*(x) = \max \{T_i x + V_i\}_{i=1}^{N_P} \quad \text{for} \quad x \in \mathcal{X}_f = \bigcup_{i=1}^{N_P} \mathcal{P}_i.$$

From (5.12) and (5.13), the polyhedral region $\mathcal{P}_j$ containing $x$ can be simply identified by searching for the maximum number in the list $\{T_i x + V_i\}_{i=1}^{N_P}$.

$$x \in \mathcal{P}_j \iff T_j' x + V_j = \max \{T_i x + V_i\}_{i=1}^{N_P}.\quad (5.14)$$

Therefore, instead of searching for the polyhedron $j$ that contains the point $x$ via Algorithm 5.2, we can just store the value function and identify region $j$ by searching for the maximum in the list of numbers composed of the single affine function $T_i x + V_i$ evaluated at $x$:

Algorithm 5.3.

1. Compute the list $\mathcal{T} = \{t_i := T_i x + V_i\}_{i=1}^{N_P}$
2. Find $j$ such that $t_j = \max_{t_i \in \mathcal{T}} t_i$

For illustration see example in Figure 5.1, where we have $N_P = 4$, $f_1(x) = -0.5x + 3$, $f_2(x) = 2$, $f_3(x) = 0.5x$, and $f_4(x) = 2x - 9$.

Algorithm 5.3 requires the storage of $(n_x + 1)N_P$ real numbers and it will give a solution after $n_xN_P$ multiplications, $(n_x - 1)N_P$ sums, and $N_P - 1$ comparisons. In Table 5.1 we compare the complexity of Algorithm 5.3 against Algorithm 5.2 in terms of storage demand and number of flops.

Remark 5.3. Algorithm 5.3 will outperform Algorithm 5.2 since typically the total number of halfspaces defining the polyhedral partition of the feasible set $\mathcal{X}_f$ is much bigger than the number of polyhedral regions, i.e., $N_H \gg N_P$ (see Remark 5.1).
Figure 5.1: Example for Algorithm 5.3 in one dimension: For a given point $x \in P_3$ ($x = 5$) we have $f_3(x) = \max(f_1(x), f_2(x), f_3(x), f_4(x))$.

5.3.2 Efficient Implementation, quadratic cost Case

Consider the explicit solution of CFTOC problem (5.3) with the quadratic cost (5.4b). Theorem 5.2 states that the value function $J^*(x)$ is convex and piecewise quadratic and the simple Algorithm 5.3 described in the previous subsection cannot be used here. Instead, a modified approach is described below.

We will first establish the following general result: given a general polyhedral partition of the state space, we can locate where the state lies (i.e., in which polyhedron) by using a search procedure based on the information provided by an “appropriate” PWA continuous function defined over the same polyhedral partition. We will refer to such an “appropriate” PWA function as a PWA descriptor function. First we outline the properties of the PWA descriptor function and then we describe the search procedure itself.

**Definition 5.1.** Two polyhedra $P_i, P_j$ of $\mathbb{R}^n_x$ are called neighboring polyhedra if their interiors are disjoint and $P_i \cap P_j$ is $(n_x - 1)$-dimensional (i.e., is a common facet).

Let $\{P_i\}_{i=1}^{N_p}$ be the polyhedral partition obtained by solving the mp-QP (5.10). For each polyhedra $P_i$ we denote with $C_i$ the list of all its neighbors

$$ C_i := \{ j \mid P_j \text{ is a neighbor of } P_i, \ j = 1, \ldots, N_p, j \neq i \}, \ i = 1, \ldots, N_p. \quad (5.15) $$

Note that the list $C_i$ has at most $p_i$ elements since some of the boundaries of $P_i$ may be outer boundaries of the polyhedral partition $\{P_i\}_{i=1}^{N_p}$ and they would not introduce element in the list $C_i$. We give the following definition of a PWA descriptor function:

**Definition 5.2 (PWA descriptor function).** A scalar continuous real-valued PWA function $f : \mathbb{R}^{n_x} \to \mathbb{R}$

$$ f(x) = f_i(x) := A'_i x + B_i, \text{ if } x \in P_i, \quad (5.16) $$
is called descriptor function if
\[ A_i \neq A_j, \quad \forall j \in \mathcal{C}_i, \quad i = 1, \ldots, N_p, \]
where \( A_i \in \mathbb{R}^{n_x}, B_i \in \mathbb{R} \), and \( \mathcal{C}_i \) is the list of neighbors of \( \mathcal{P}_i \).

**Definition 5.3** (Ordering function). Let \( f(x) \) be a PWA descriptor function on the polyhedral partition \( \{ \mathcal{P}_i \}_{i=1}^{N_p} \). We define ordering function \( O_{i,j}(x) \) as
\[
O_{i,j}(x) := \begin{cases} 
+1 & \text{if } f_i(x) \geq f_j(x) \\
-1 & \text{if } f_i(x) < f_j(x)
\end{cases}
\]
with \( i \in \{1, \ldots, N_p\}, \ j \in \mathcal{C}_i \).

**Theorem 5.3.** Let \( f(x) \) be a PWA descriptor function on the polyhedral partition \( \{ \mathcal{P}_i \}_{i=1}^{N_p} \). Let
\[ \mathcal{S}_{i,j} := O_{i,j}(\xi_i), \]
with \( \xi_i \in \mathbb{R}^{n_x}, \xi_i \in \text{int}(\mathcal{P}_i), \ i = 1, \ldots, N_p, \ j \in \mathcal{C}_i \). Then the following holds
\[ x \in \text{int}(\mathcal{P}_i) \iff O_{i,j}(x) = \mathcal{S}_{i,j}, \forall j \in \mathcal{C}_i \]

**Proof** Let \( \mathcal{F} = \mathcal{P}_i \cap \mathcal{P}_j \) be the common facet of two neighboring polyhedra \( \mathcal{P}_i \) and \( \mathcal{P}_j \). Define the linear function
\[ g_{i,j}(x) = f_i(x) - f_j(x). \]
From the continuity of descriptor function \( f(x) \) it follows that \( g_{i,j}(x) = 0, \forall x \in \mathcal{F} \). As \( \mathcal{P}_i \) and \( \mathcal{P}_j \) are disjoint convex polyhedra and \( A_i \neq A_j \) it follows that \( g_{i,j}(x) = 0 \) is a separating hyperplane between \( \mathcal{P}_i \) and \( \mathcal{P}_j \).

(i) “⇒” part. Since \( g_{i,j}(x) = 0 \) is a separating hyperplane between \( \mathcal{P}_i \) and \( \mathcal{P}_j \) it follows that \( g_{i,j}(x) \) does not change sign for all \( x \in \text{int}(\mathcal{P}_i) \). Hence, we have \( O_{i,j}(x) = \mathcal{S}_{i,j} \).

(ii) “⇐” part by contradiction. Assume that \( O_{i,j}(\bar{x}) = \mathcal{S}_{i,j} \) while \( \bar{x} \notin \mathcal{P}_i \). It is easy to see that \( \forall \bar{x} \notin \mathcal{P}_i, \exists j \in \mathcal{C}_i \) such that \( (P_i^x)_{(j) \bar{x}} > (P_i)_{(j)} \). This, however, implies that \( g_{i,j}(\bar{x}) \) has different sign compared to \( g_{i,j}(\xi) \), \( \xi \in \text{int}(\mathcal{P}_i) \). Hence we have \( O_{i,j}(\bar{x}) \neq \mathcal{S}_{i,j} \), contradiction.

Theorem 5.3 states that the ordering function \( O_i(x) := [O_{i,j}(x)]_{j \in \mathcal{C}_i} \) and the vector \( \mathcal{S}_i := [\mathcal{S}_{i,j}]_{j \in \mathcal{C}_i} \) uniquely characterize \( \mathcal{P}_i \). Therefore to check on-line if the polyhedral region \( \mathcal{P}_i \) contains the state \( x \) it is sufficient to compute the binary vector \( O_i(x) \) and compare it with \( \mathcal{S}_i \). Vectors \( \mathcal{S}_i \) are calculated off-line for \( i = 1, \ldots, N_p \), by comparing the values of \( f_i(x) \) and \( f_j(x) \) for all \( j \in \mathcal{C}_i \), in a point \( \xi \) belonging to \( \text{int}(\mathcal{P}_i) \), for instance, the Chebychev center of \( \mathcal{P}_i \).

In Figure 5.2 a one dimensional example illustrates the procedure with \( N_p = 4 \) regions and for \( f_1(x) = x, f_2(x) = 2, f_3(x) = x - 3, f_4(x) = -1/3 x + 10/3 \). The list of neighboring regions \( \mathcal{C}_i \) and the vector \( \mathcal{S}_i \) can be constructed by simply looking at the figure: \( C_1 = \{ 2 \}, \ C_2 = \{ 1, 3 \}, \ C_3 = \{ 2, 4 \}, \ C_4 = \{ 3 \}, \ S_1 = -1, \ S_2 = [-1 1]', \ S_3 = [1 -1]', \ S_4 = -1 \). The point \( x = 4 \) is in region 2 and we have \( O_2(x) = [-1 1]' = S_2 \), while \( O_3(x) = [-1 -1]' \neq S_3 \), \( O_1(x) = 1 \neq S_1, O_4(x) = 1 \neq S_1 \). The failure of a match \( O_i(x) = S_i \) provides information on a good search direction(s). The solution can be found by searching in the direction where a constraint is violated, i.e., we should check the neighboring region \( \mathcal{P}_j \) for which \( O_{i,j}(x) \neq \mathcal{S}_{i,j} \).

The overall procedure is composed of two parts:
Figure 5.2: Example for Algorithm 5.4 in one dimension: For a given point $x \in \mathcal{P}_2 \ (x = 4)$ we have $O_2(x) = [-1 \ 1]' = S_2$, while $O_1(x) = 1 \neq S_1 = -1$, $O_3(x) = [-1 \ -1]' \neq S_3 = [1 \ -1]'$, $O_4(x) = 1 \neq S_4 = -1$.

1. **(off-line)** Construction of the scalar continuous real-valued PWA function $f(x)$ in (5.16) satisfying (5.17) and computation of the list of neighbors $\mathcal{C}_i$ and the vector $S_i$.

2. **(on-line)** Execution of the following algorithm

**Algorithm 5.4.**

1. $i = 1$, $\mathcal{C} = \mathcal{C}_i$
2. WHILE $\mathcal{C} \neq \emptyset$
3. 1. $j \leftarrow \mathcal{C}$, $\mathcal{C} \leftarrow \mathcal{C} \setminus \{j\}$
4. 2. compute $O_{i,j}(x)$
5. 3. IF $O_{i,j}(x) \neq S_{i,j}$ THEN $\mathcal{C} = \mathcal{C}_j$
6. END

Algorithm 5.4 does not require the storage of the polyhedra $\mathcal{P}_i$, but only the storage of one affine function $f_i(x)$ per polyhedron, i.e., $N_P(n_x + 1)$ real numbers and the list of neighbors $\mathcal{C}_i$ which demands (at most) $N_H$ integers. Algorithm 5.4 in the worst case terminates after $N_P n_x$ multiplications, $N_P (n_x - 1)$ sums and $N_H$ comparisons.

In Table 5.2 we compare the complexity of Algorithm 5.4 against the standard Algorithm 5.2 in terms of storage demand and number of flops.

**Remark 5.4.** Note that the computation of $O_i(x)$ in Algorithm 5.4 requires the evaluation of $p_i$ linear functions, but the overall computation never exceeds $N_P$ linear function evaluations. Consequently, Algorithm 5.4 will outperform Algorithm 5.2, since typically $N_H \gg N_P$. 


Table 5.2: Complexity comparison of Algorithm 5.2 and Algorithm 5.4

<table>
<thead>
<tr>
<th></th>
<th>Algorithm 5.2</th>
<th>Algorithm 5.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage demand (real numbers)</td>
<td>((n_x + 1)N_H)</td>
<td>((n_x + 1)N_P)</td>
</tr>
<tr>
<td>Number of flops (worst case)</td>
<td>(2n_xN_H)</td>
<td>((2n_x - 1)N_P + N_H)</td>
</tr>
</tbody>
</table>

Now that we have shown how to locate the polyhedron in which the state lies by using a PWA descriptor function, we need a procedure for the construction of such a function.

The image of the descriptor function is the set of real numbers \(\mathbb{R}\). In the following we will show how a descriptor function can be generated from a vector valued function \(m : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^s\). This general result will be used in the next subsections.

**Definition 5.4 (Vector valued PWA descriptor function).** A continuous vector valued PWA function \(m : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^s\)

\[
m(x) = \tilde{A}_i x + \tilde{B}_i, \quad \text{if} \quad x \in P_i,
\]

is called a vector valued PWA descriptor function if

\[
\tilde{A}_i \neq \tilde{A}_j, \quad \forall j \in C_i, \quad i = 1, \ldots, N_P.
\]

where \(\tilde{A}_i \in \mathbb{R}^{s \times n_x}, \tilde{B}_i \in \mathbb{R}^s\).

**Theorem 5.4.** Given a vector valued PWA descriptor function \(m(x)\) defined over a polyhedral partition \(\{P_i\}_{i=1}^{N_H}\) it is possible to construct a PWA descriptor function \(f(x)\) over the same polyhedral partition.

**Proof** Let \(N_{i,j}\) be the null-space of \((\tilde{A}_i - \tilde{A}_j)'\). Since by definition \(\tilde{A}_i - \tilde{A}_j \neq 0\) it follows that \(N_{i,j}\) is not full dimensional, i.e., \(N_{i,j} \subset \mathbb{R}^{s-1}\). Consequently, it is always possible to find a vector \(w \in \mathbb{R}^s\) such that \(w'(\tilde{A}_i - \tilde{A}_j) \neq 0\) holds for all \(i = 1, \ldots, N_P\) and \(\forall j \in C_i\). Clearly, \(f(x) = w'm(x)\) is then a valid PWA descriptor function.

As shown in the proof of Theorem 5.4, once we have vector valued PWA descriptor function, practically any randomly chosen vector \(w \in \mathbb{R}^s\) is likely to be satisfactory for the construction of a PWA descriptor function. From a numerical point of view, however, we would like to obtain \(w\) that is as far away as possible from the null-spaces \(N_{i,j}\). We show one algorithm for finding such a vector \(w\).

For a given vector valued PWA descriptor function we form set of vectors \(a_k \in \mathbb{R}^s, \|a_k\| = 1, k = 1, \ldots, N_H/2\), by taking and normalizing one (and only one) nonzero column from each matrix \((\tilde{A}_i - \tilde{A}_j), \forall j \in C_i, i = 1, \ldots, N_P\). The vector \(w \in \mathbb{R}^s\) satisfying the set of equations \(w'a_k \neq 0, k = 1, \ldots, N_H/2\), can then be constructed by using the following algorithm:\(^3\)

**Algorithm 5.5.**

\(^3\)Index \(k\) goes to \(N_H/2\) since the term \((\tilde{A}_j - \tilde{A}_i)\) is the same as \((\tilde{A}_i - \tilde{A}_j)\) and thus there is no need to consider it twice.
1. \( w \leftarrow [1, \ldots, 1]' \), \( R \leftarrow 1 \)
2. \[ \textbf{WHILE} \ k \leq \frac{N_h}{2} \]
2.1. \( d \leftarrow w'a_k \)
2.2. \[ \textbf{IF} \ 0 \leq d \leq R \ \textbf{THEN} \ w \leftarrow w + \frac{1}{2}(R-d)a_k , \ R \leftarrow \frac{1}{2}(R+d) \]
2.3. \[ \textbf{IF} \ -R \leq d < 0 \ \textbf{THEN} \ w \leftarrow w - \frac{1}{2}(R+d)a_k , \ R \leftarrow \frac{1}{2}(R-d) \]
3. \textbf{END}

Essentially, Algorithm 5.5 constructs a sequence of balls \( \mathcal{B} = \{ x \mid x = w + r, \| r \|_2 \leq R \} \). As depicted in Figure 5.3, we start with the initial ball of radius \( R = 1 \), centered at \( w = [1, \ldots, 1]' \). Iteratively one hyperplane \( a_k' x = 0 \) at a time is introduced and the largest ball \( \mathcal{B}' \subseteq \mathcal{B} \) that does not intersect this hyperplane is constructed. The center \( w \) of the final ball is the vector \( w \) we want to obtain, while \( R \) provides information about the degree of non-orthogonality: \( |w'a_k| \geq R, \forall k \).

In the following subsections we will show that the gradient of the value function, and the optimizer, are vector valued PWA descriptor functions and thus we can use Algorithm 5.5 for the construction of the PWA descriptor function.

**Generating a PWA descriptor function from the value function**

We will first prove that \( J^*(x) \) is continuously differentiable function and then we will show that the gradient of \( J^*(x) \) is a vector valued PWA descriptor function.

Let \( J^*(x) \) be the convex and piecewise quadratic (CPWQ) value function corresponding to the explicit solution of CFTOC (5.3) for the quadratic cost function (5.4b):

\[
J^*(x) = q_i(x) := x'Q_i x + T_i'x + V_i, \quad \text{if} \ x \in \mathcal{P}_i, \ i = 1, \ldots, N_P.
\]  

Before going further we recall the following result [TJB01b]:

![Figure 5.3: Illustration for Algorithm 5.5 in two dimensions.](image-url)
Theorem 5.5. Consider the set $A^*(x)$ of active constraints at the optimum of QP (5.10), and assume there is no degeneracy:

$$A^*(x) = \{ i \mid M^U (x) = M_i + M^F (x) \}$$ (5.25)

1. $A^*(x)$ is constant $\forall x \in \mathcal{P}_i$, $i = 1, \ldots, N_P$, i.e., $A^*(x) := A_i \forall x \in \mathcal{P}_i$.

2. If $\mathcal{P}_i$ and $\mathcal{P}_j$ are neighboring polyhedra then $A_i \subset A_j$ or $A_j \subset A_i$.

Theorem 5.6. Suppose that the QP problem (5.10) is not degenerate. Consider the value function $J^*(x)$ in (5.24) and let $\mathcal{P}_i, \mathcal{P}_j$ be two neighboring polyhedra corresponding to the set of active constraints $A_i$ and $A_j$, respectively, then

$$Q_i - Q_j \leq 0 \quad \text{or} \quad Q_i - Q_j \geq 0 \quad \text{and} \quad Q_i \neq Q_j$$ (5.26)

and

$$Q_i - Q_j \leq 0 \quad \text{iff} \quad A_i \subset A_j,$$ (5.27)

where $Q \geq 0$ denotes positive semidefiniteness (resp., $Q \leq 0$ negative semidefiniteness).

Proof Let $\mathcal{P}_i$ and $\mathcal{P}_j$ be two neighboring polyhedra and $A_i$ and $A_j$ be the corresponding sets of active constraints at the optimum of QP (5.10). Let $A_i \subset A_j$. We want to prove that the difference between the quadratic terms of $q_i(x)$ and $q_j(x)$ is negative semidefinite, i.e., $Q_i - Q_j \leq 0$ and that $Q_i \neq Q_j$.

Without loss of generality we can assume that $A_i = \emptyset$. If this is not the case a simple substitution of variables based on the set of active constraints $M^U (A_i) U = M_{(A_i)} + M^F_{(A_i)} x$ transforms Problem (5.10) into a QP in a lower dimensional space.

With the substitution $z = U + H^{-1} F' x$ Problem (5.10) can be translated into the following:

$$J^*_z (x) = \min_{z} \frac{1}{2} z' H z$$

$$\text{subj. to } G z \leq W + S x$$ (5.28)

where $G := M^U$, $W := M$, $S := M^x + M^U H^{-1} F'$ and $J^*_z (x) = J^*(x) - \frac{1}{2} x' (Y - FH^{-1} F') x$. For the unconstrained case we have $z^* = 0$ and $J^*_z (x) = 0$. Consequently

$$q_i (x) = \frac{1}{2} x' (Y - FH^{-1} F') x.$$ (5.29)

For the constrained case, as shown in [BMDP02], from the set of active constraints $G_{(A_i)} z = W_{(A_i)} + S_{(A_i)} x$ and the Karush-Kuhn-Tucker (KKT) conditions we obtain

$$z = H^{-1} G'_{(A_i)} \Gamma^{-1} (W_{(A_i)}) + S_{(A_i)} x,$$ (5.30)

$$\lambda_{(A_i)} = -\Gamma^{-1} (W_{(A_i)}) + S_{(A_i)} x,$$ (5.31)

where $\Gamma = G_{(A_i)} H^{-1} G'_{(A_i)}$, $\Gamma = \Gamma' > 0$ as the rows of $G_{(A_i)}$ are linearly independent and $\lambda_{(A_i)}$ are the Lagrange multipliers of the active constraints, $\lambda_{(A_i)} \geq 0$. The corresponding value function is

$$q_j (x) = \frac{1}{2} x' (Y - FH^{-1} F' + S'_{(A_i)} \Gamma^{-1} S_{(A_i)}) x + W'_{(A_i)} \Gamma^{-1} S_{(A_i)} x + \frac{1}{2} W'_{(A_i)} \Gamma^{-1} W_{(A_i)}.$$ (5.32)
The difference of the quadratic terms of \( q_i(x) \) and \( q_j(x) \) gives
\[
Q_i - Q_j = \frac{1}{2} S_{(A_j)}^\top \Gamma^{-1} S_{(A_j)} \preceq 0. \tag{5.33}
\]

What is left to prove is \( Q_i \neq Q_j \). We will prove it by showing that \( Q_i = Q_j \) if and only if \( \mathcal{P}_i = \mathcal{P}_j \). For this purpose we recall [BMDP02] that the polyhedron \( \mathcal{P}_j \) where the set of active constraints \( A_j \) is constant is defined as
\[
\mathcal{P}_j = \left\{ x \mid GH^{-1}G_{(A_j)}^\top \Gamma^{-1} (W_{(A_j)} + S_{(A_j)} x) \leq W + Sx, -\Gamma^{-1} (W_{(A_j)} + S_{(A_j)} x) \geq 0 \right\}. \tag{5.34}
\]

From (5.33) we conclude that \( Q_i = Q_j \) if and only if \( S_{(A_i)} = S_{(A_j)} = 0 \). The continuity of \( J^*_r(x) \) implies that \( q_i(x) - q_j(x) = 0 \) on the common facet of \( \mathcal{P}_i \) and \( \mathcal{P}_j \). Therefore, by comparing (5.29) and (5.32) we see that \( S_{(A_j)} = 0 \) implies \( W_{(A_j)} = 0 \). Finally, for \( S_{(A_i)} = S_{(A_j)} = 0 \) and \( W_{(A_j)} = 0 \), from (5.34) it follows that \( \mathcal{P}_j = \mathcal{P}_i := \{ x \mid 0 \leq W + Sx \} \).

The following property of convex piecewise quadratic functions was proved in [Sch87]:

**Theorem 5.7.** Consider the value function \( J^*(x) \) in (5.24) satisfying (5.26) and its quadratic expression \( q_i(x) \) and \( q_j(x) \) on two neighboring polyhedra \( \mathcal{P}_i, \mathcal{P}_j \) then
\[
q_i(x) = q_j(x) + (a'x - b)(\gamma a'x - \bar{b}), \tag{5.35}
\]
where \( \gamma \in \mathbb{R}/\{0\} \).

Equation (5.35) states that the functions \( q_i(x) \) and \( q_j(x) \) in two neighboring regions \( \mathcal{P}_i, \mathcal{P}_j \) of a CPWQ function satisfying (5.26) either intersect on two parallel hyperplanes: \( a'x - \bar{b} \) and \( \gamma a'x - b \) if \( \bar{b} \neq \gamma b \) (see Figure 5.4(a)) or are tangent in one hyperplane: \( a'x - b \) if \( \bar{b} = \gamma b \) (see Figure 5.4(b)). We will prove next that if the QP problem (5.10) is not degenerate then \( J^*(x) \) is a \( C^{(1)} \) function by showing that the case depicted in Figure 5.4(a) is not consistent with Theorem 5.6. In fact, Figure 5.4(a) depicts case \( Q^i - Q^j \preceq 0 \), that implies \( A_i \subset A_j \) by Theorem 5.6. However \( q_i(0) < q_j(0) \) and from the definition of \( q_i \) and \( q_j \) this contradicts the fact that \( A_i \subset A_j \).

**Theorem 5.8.** Assume that the QP problem (5.10) is not degenerate, then the value function \( J^*(x) \) in (5.24) is \( C^{(1)} \).

**Proof** We will prove by contradiction that \( \bar{b} = \gamma b \). Suppose there exists two neighboring polyhedra \( \mathcal{P}_i \) and \( \mathcal{P}_j \) such that \( \bar{b} \neq \gamma b \). Without loss of generality assume that (i) \( Q_i - Q_j \preceq 0 \) and (ii) \( \mathcal{P}_i \) is in the halfspace \( a'x \leq \bar{b} \) defined by the common boundary. Let \( \mathcal{F}_{ij} \) be the common facet between \( \mathcal{P}_i \) and \( \mathcal{P}_j \) and \( \text{int}(\mathcal{F}_{ij}) \) its interior.

From (i) and from (5.35), either \( \gamma < 0 \) or \( \gamma = 0 \) if \( Q_i - Q_j = 0 \). Take \( x_0 \in \text{int}(\mathcal{F}_{ij}) \). For sufficiently small \( \varepsilon \geq 0 \), the point \( x := x_0 - \varepsilon \) belongs to \( \mathcal{P}_i \).

Let \( J^*(\varepsilon) := J^*(x_0 - a\varepsilon), \quad q_i(\varepsilon) := q_i(x_0 - a\varepsilon) \), and consider
\[
q_i(\varepsilon) = q_j(\varepsilon) + (a'\varepsilon)(\gamma a'\varepsilon + (\bar{b} - \gamma b)). \tag{5.36}
\]

From convexity of \( J^*(\varepsilon) \), \( J^-*(\varepsilon) \leq J^+*(\varepsilon) \) where \( J^-*(\varepsilon) \) and \( J^+*(\varepsilon) \) are the left and right derivatives of \( J^*(\varepsilon) \) with respect to \( \varepsilon \). This implies \( q_j'(\varepsilon) \leq q_j'(\varepsilon) \) where \( q_j'(\varepsilon) \) and \( q_j'(\varepsilon) \) are
the derivatives of \( q_j(\varepsilon) \) and \( q_i(\varepsilon) \), respectively. Condition \( q_j'(\varepsilon) \leq q_i'(\varepsilon) \) is true if and only if \(- (\bar{b} - \gamma b) \leq 2\gamma(a'a)\varepsilon \), that implies \(- (\bar{b} - \gamma b) < 0 \) since \( \gamma < 0 \) and \( \varepsilon > 0 \).

From (5.36) \( q_j(\varepsilon) < q_i(\varepsilon) \) for all \( \varepsilon \in (0, \frac{-(\bar{b} - \gamma b)}{\gamma a'a}) \).

Thus there exists \( x \in \mathcal{P}_i \) with \( q_j(x) < q_i(x) \). This is a contradiction since from Theorem 5.5, \( \mathcal{A}_i \subset \mathcal{A}_j \).

Note that in case of degeneracy the value function \( J^*(x) \) in (5.24) may not be \( C^{(1)} \); counterexamples are given in [BRT97].

In Theorem 5.8 we have proven that the value function is \( C^{(1)} \). Now we want to show that the gradient of \( J^*(x) \) is a vector valued PWA descriptor function.

**Theorem 5.9.** Consider the value function \( J^*(x) \) in (5.24) and assume that the CFTOC problem (5.3) leads to a non-degenerate QP (5.10). Then the gradient \( m(x) := \nabla J^*(x) \), is a vector valued PWA descriptor function.

**Proof** From Theorem 5.8 we see that \( m(x) \) is continuous vector valued PWA function, while from equation (5.24) we get

\[
  m(x) := \nabla J^*(x) = 2Q_i x + T_i \tag{5.37}
\]

Since from Theorem 5.6 we know that \( Q_i \neq Q_j \) for all neighboring polyhedra, it follows that \( m(x) \) satisfies all conditions for a vector valued PWA descriptor function.

Combining results of Theorem 5.9 and Theorem 5.4 it follows that by using Algorithm 5.5 we can construct a PWA descriptor function from the gradient of the value function \( J^*(x) \).

**Generating a PWA descriptor function from the optimizer**

Another way to construct a vector valued PWA descriptor function \( m(x) \) emerges naturally if we look at the properties of the optimizer \( U^*(x) \) corresponding to the state feedback solution.
of the CFTOC problem (5.3). From Theorem 5.2, the optimizer $U^*(x)$ is continuous in $x$ and piecewise affine:

$$U^*(x) = l_i(x) := \bar{F}_i x + \bar{G}_i, \quad \text{if } x \in \mathcal{P}_i, \ i = 1, \ldots, N_p,$$

(5.38)

where $\bar{F}_i \in \mathbb{R}^{s \times n_x}$ and $\bar{G}_i \in \mathbb{R}^s$.

All we need to show is the following lemma.

**Lemma 5.1.** Consider the state feedback solution (5.38) of the CFTOC problem (5.3) and assume that CFTOC problem (5.3) leads to a non-degenerate QP (5.10). Let $\mathcal{P}_i, \mathcal{P}_j$ be two neighboring polyhedra, then $\bar{F}_i \neq \bar{F}_j$.

**Proof** The proof is a simple consequence of Theorem 5.6. As in Theorem 5.6, without loss of generality we can assume that $\mathcal{A}_i = \emptyset$. Suppose that the optimizer is the same for both polyhedra, i.e., $[\bar{F}_i \ G_i] = [\bar{F}_j \ G_j]$. Then the cost functions $q_i(x)$ and $q_j(x)$ are also equal. From the proof of Theorem 5.6 this implies that $\mathcal{P}_i = \mathcal{P}_j$, which is a contradiction. Thus we have $[\bar{F}_i \ G_i] \neq [\bar{F}_j \ G_j]$. Note that $\bar{F}_i = \bar{F}_j$ cannot happen since, from the continuity of $U^*(x)$, this would imply $\bar{G}_i = \bar{G}_j$. Consequently we have $\bar{F}_i \neq \bar{F}_j$.

From Lemma 5.1 and Theorem 5.4 it follows that an appropriate PWA descriptor function $f(x)$ can be calculated from the optimizer $U^*(x)$ by using Algorithm 5.5.

**Remark 5.5.** Note that even if we are implementing the receding horizon control strategy, the construction of the PWA descriptor function is based on the full optimization vector $U^*(x)$ and the corresponding matrices $\bar{F}_i$ and $\bar{G}_i$.

**Remark 5.6.** In some cases the use of the optimal control profile $U^*(x)$ for the construction of a descriptor function $f(x)$ can be extremely simple. If there is a row $r$, $r \leq n_u$ ($n_u$ is the dimension of $u$) for which $(\bar{F}_i)_r \neq (\bar{F}_j)_r), \ \forall i = 1, \ldots, N_p, \ \forall j \in \mathcal{C}_i$, it is enough to set $A_i^r = (\bar{F}_i)_r$ and $B_i^r = (\bar{G}_i)_r$, where $(\bar{F}_i)_r$ and $(\bar{G}_i)_r$ denote $r$-th row of the matrices $\bar{F}_i$ and $\bar{G}_i$, respectively. In this way we avoid the storage of the descriptor function, since it is equal to one component of the control law, which is stored anyway.

### 5.4 Example

As an example, we compare the performance of Algorithms 5.2, 5.3 and 5.4 on CFTOC problem for the discrete-time system

$$\begin{align*}
    \begin{cases}
        x(t+1) &= \begin{bmatrix} 4 & -1.5 & 0.5 & -0.25 \\ 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) \\
        y(t) &= \begin{bmatrix} 0.083 & 0.22 & 0.11 & 0.02 \end{bmatrix} x(t)
    \end{cases}
\end{align*}
$$

(5.39)

resulting from the linear system

$$y = \frac{1}{s^4} u,$$

(5.40)
sampled at $T_s = 1$, subject to the input constraint

$$-1 \leq u(t) \leq 1$$

and the output constraint

$$-10 \leq y(t) \leq 10.$$ 

5.4.1 CFTOC, linear cost Case

To regulate (5.39), we design a receding horizon controller based on the optimization problem (5.3) where $p = \infty$, $N = 2$, $Q = \text{diag}\{5, 10, 10, 10\}$, $R = 0.8$, $P = 0$. The PWA solution of the mp-LP problem was computed in 240 s on a Pentium III 900 MHz machine running Matlab 6.0. The corresponding polyhedral partition of the state-space consists of 136 regions. In Table 5.3 we report the comparison between the complexity of Algorithm 5.2 and Algorithm 5.3 for this example.

The average on-line RHC computation for a set of 1000 random points in the state space is 2259 flops (Algorithm 5.2), and 1088 flops (Algorithm 5.3). We point out that the solution of the same problem using Matlab’s LP solver (function linprog.m with interior point algorithm and LargeScale set to ’off’) takes 25459 flops on average.

Table 5.3: Complexity comparison of Algorithm 5.2 and Algorithm 5.3 for the example in Section 5.4.1

<table>
<thead>
<tr>
<th></th>
<th>Algorithm 5.2</th>
<th>Algorithm 5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage demand (real numbers)</td>
<td>5690</td>
<td>680</td>
</tr>
<tr>
<td>Number of flops (worst case)</td>
<td>9104</td>
<td>1088</td>
</tr>
</tbody>
</table>

5.4.2 CFTOC, quadratic cost Case

To regulate (5.39), we design a receding horizon controller based on the optimization problem (5.3) where $p = 2$, $N = 7$, $Q = I$, $R = 0.01$, $P = 0$. The PWA solution of the mp-QP problem was computed in 560 s on Pentium III 900 MHz machine running Matlab 6.0. The corresponding polyhedral partition of the state-space consists of 213 regions. For this example the choice of $w = [1 1 1 1 1 1 1]'$ is satisfactory to obtain a descriptor function from the optimizer. In Table 5.4 we report the comparison between the complexity of Algorithm 5.2 and Algorithm 5.4 for this example.

The average on-line RHC computation for a set of 1000 random points in the state space is 2114 flops (Algorithm 5.2), and 175 flops (Algorithm 5.4). The solution of the corresponding quadratic program with Matlab’s QP solver (function quadprog.m and LargeScale set to ’off’) takes 25221 flops on average.
Table 5.4: Complexity comparison of Algorithm 5.2 and Algorithm 5.4 for the example in Section 5.4.2

<table>
<thead>
<tr>
<th></th>
<th>Algorithm 5.2</th>
<th>Algorithm 5.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage demand (real numbers)</td>
<td>9740</td>
<td>1065</td>
</tr>
<tr>
<td>Number of flops (worst case)</td>
<td>15584</td>
<td>3439</td>
</tr>
</tbody>
</table>

5.5 Conclusion

By exploiting properties of the value function and the optimal solution to the CFTOC problem, we presented two algorithms that significantly improve the efficiency of the on-line calculation of the control action (LP-based and QP-based) in terms of storage demand and computational complexity. The following improvements are achieved

1. There is no need to store the polyhedral partition of the state space.

2. In the worst case, the optimal control law is computed after the evaluation of one linear function per polyhedron.
Constrained Finite Time Optimal Control of Piecewise Affine Systems

In this chapter we study the solution to optimal control problems for constrained discrete-time piecewise affine systems based on quadratic or linear performance criteria. First we give basic theoretical results on the structure of the optimal state-feedback solution and of the value function. Second we describe how the state-feedback optimal control law can be constructed by combining multi-parametric programming and dynamic programming.

6.1 Introduction

Recent technological innovations have caused a considerable interest in the study of dynamical processes of a mixed continuous and discrete nature, denoted as hybrid systems. In their most general form hybrid systems are characterized by the interaction of continuous-time models (governed by differential or difference equations), and of logic rules and discrete event systems (described, for example, by temporal logic, finite state machines, if-then-else rules) and discrete components (on/off switches or valves, gears or speed selectors, etc.). Such systems can switch between many operating modes where each mode is governed by its own characteristic dynamical laws. Mode transitions are triggered by variables crossing specific thresholds (state events), by the lapse of certain time periods (time events), or by external inputs (input events) [Ant00]. A detailed discussion of different modeling frameworks for hybrid systems that appeared in the literature goes beyond the scope of this thesis, the main concepts can be found in [Ant00, BBM98, BM99, LTS99].

Different methods for the analysis and design of controllers for hybrid systems have emerged over the last few years [Son81, LTS99, BM99]. Among them, the class of optimal controllers is one of the most studied. The approaches differ greatly in the hybrid models adopted, in the formulation of the optimal control problem and in the method used to solve it.

In this thesis we focus on discrete-time linear hybrid models. In our hybrid modeling framework we allow (i) the system to be discontinuous, (ii) both states and inputs to assume continuous and discrete values, (iii) events to be both internal, i.e., caused by the state reaching a particular boundary, and exogenous, i.e., forced by a switch to some other operating mode, and (iv) states and inputs to fulfill linear constraints. We will focus on discrete-time piecewise affine (PWA) models. Discrete-time PWA models can describe a large number
of processes, such as: discrete-time linear systems with static piecewise-linearities; discrete-
time linear systems with discrete states and inputs; switching systems where the dynamic
behaviour is described by a finite number of discrete-time linear models together with a set
of logic rules for switching among these models; approximation of nonlinear discrete-time
dynamics, e.g., via multiple linearizations at different operating points.

In discrete-time hybrid systems an event can occur only at instants that are multiples of
the sampling time and many interesting mathematical phenomena occurring in continuous-
time hybrid systems such as Zeno behaviors do not exist. However, the solution to optimal
control problems is still complex: the solution to the HJB equation can be discontinuous and
the number of possible switches grows exponentially with the length of the horizon of the
optimal control problem. Nevertheless, we will show that for the class of linear discrete-time
hybrid systems we can characterize and compute the optimal control law exactly without
gridding the state space.

The solution to optimal control problems for discrete-time hybrid systems was first outlined
by Sontag in [Son81]. In his plenary presentation [May01] at the 2001 European Control
Conference, Mayne presented an intuitively appealing characterization of the state-feedback
solution to optimal control problems for linear hybrid systems with performance criteria
based on quadratic and linear norms. The detailed exposition presented in the initial part
of this chapter follows a similar line of argumentation and shows that the state-feedback solution
to the finite time optimal control problem is a time-varying piecewise affine feedback control
law, possibly defined over non-convex regions. Moreover, we give insight into the structure
of the optimal state-feedback solution and of the value function.

In the second part of the chapter we describe how the optimal control law can be effi-
ciently computed by means of multi-parametric programming. In particular, we propose a
novel algorithm that solves the Hamilton-Jacobi-Bellman equation by using a simple multi-
parametric solver. In collaboration with different companies and institutes, the results de-
scribed in this chapter have been applied to a wide range of problems ([FTGS02, Mig02,
BM99, TB04, BBM03a, BGKH02, BBFH01, BVMP03, MBM03]). Simple examples that high-
light the main features of the hybrid system approach presented in this chapter can be found
in [BBBM03a].

Before formulating optimal control problems for hybrid systems we will give a short
overview on discrete-time piecewise affine systems.

### 6.2 Piecewise Affine Systems

Several modeling frameworks have been introduced for discrete-time hybrid systems. Among
them, piecewise affine (PWA) systems [Son81] are defined by partitioning the state space into
polyhedral regions and associating with each region a different affine state-update equation

\[
\dot{x}(t+1) = A^i x(t) + B^i u(t) + f^i, \quad \text{if} \quad x(t) \in C^i, \quad i = \{1, \ldots, s\},
\]

where \( x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_e}, u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_e}, \{C^i\}_{i=1}^s \) is a polyhedral partition of the set of
the state+input space \( \mathcal{C} \subset \mathbb{R}^{n+m} \), \( n := n_c + n_e, m := m_c + m_e \). We denote by \( x_c \in \mathbb{R}^{n_c} \) and
$u_c \in \mathbb{R}^{m_c}$ the real components of the state and input vector, respectively. We will give the following definitions of continuous PWA system.

**Definition 6.1.** We say that the discrete-time PWA system (6.1) is *continuous* if the mapping $(x_c(t), u_c(t)) \mapsto x_c(t + 1)$ is continuous and $n_t = m_t = 0$. The PWA system (6.1) is *continuous in the real input space* if the mapping $(x_c(t), u_c(t)) \mapsto x_c(t + 1)$ is continuous w.r.t. $u_c$. Analogously, we define PWA systems continuous in the real state space.

Our main motivation for focusing on discrete-time models stems from the need to analyze these systems and to solve optimization problems, such as optimal control or scheduling problems, for which the continuous time counterpart would not be easily computable.

PWA systems are equivalent to interconnections of linear systems and finite automata. In [HSB01] the authors have proven the equivalence of linear discrete-time PWA systems and other classes of discrete-time hybrid systems. PWA models can be generated automatically through appropriate conversion procedures [Bem04] from discrete hybrid automata, a very general class of linear hybrid systems that can be modeled in the language HYSDEL [TB04].

### 6.3 Problem Formulation

Consider the PWA system (6.1) subject to hard input and state constraints

$$Ex(t) + Lu(t) \leq M_c$$

for $t \geq 0$, and denote by *constrained PWA system* (CPWA) the restriction of the PWA system (6.1) over the set of states and inputs defined by (6.2),

$$x(t + 1) = A^i x(t) + B^i u(t) + f^i \quad \text{if} \quad \left[\begin{array}{c} x(t) \\ u(t) \end{array}\right] \in \tilde{C}^i,$$

where $\{\tilde{C}^i\}_{i=1}^s$ is the new polyhedral partition of the sets of state+input space $\mathbb{R}^{n+m}$ obtained by intersecting the sets $C^i$ in (6.1) with the polyhedron described by (6.2). The union of the polyhedral partitions $\tilde{C} := \cup_{i=1}^s \tilde{C}^i$ will implicitly define the feasible region $R_{feas}$ in the input space as a function of $x$:

$$R_{feas}(x) = \{ u \in \mathbb{R}^{m_c} \times \{0, 1\}^{m_\ell} | (x, u) \in \tilde{C} \}.$$  

We assume that $R_{feas}(x)$ is a compact set for any $x$ and the following

**Assumption 6.1.** System (6.3) is continuous in the real input and real state space.

Assumption 6.1 requires that the PWA function that defines the update of the continuous states is continuous on the boundaries of contiguous polyhedral cells, and therefore allows one to work with the closure of the sets $\tilde{C}^i$ without the need of introducing multi-valued state update equations. With abuse of notation in the next sections $\tilde{C}^i$ will always denote the closure of $\tilde{C}^i$. Discontinuous PWA systems will be discussed in Section 6.6.
We define the following cost function
\[
J(U_N, x(0)) := \|P x_N\|_p + \sum_{k=0}^{N-1} \|Q x_k\|_p + \|R u_k\|_p, \tag{6.4}
\]
and consider the constrained finite-time optimal control (CFTOC) problem
\[
J^*_0(x(0)) := \min_{U_N} J(U_N, x(0)) \tag{6.5}
\]
subj. to
\[
\begin{cases}
x_{k+1} = A^i x_k + B^i u_k + f^i \\
x_N \in X_f \\
x_0 = x(0)
\end{cases} \tag{6.6}
\]
where the column vector \(U_N := [u'_0, \ldots, u'_N-1]' \in \mathbb{R}^{mc \times \{0,1\}^{mN}}\), is the optimization vector, \(N\) is the optimal control horizon and \(X_f\) is a polyhedral terminal region. In (6.4), \(\|Q x\|_p\) denotes the \(p\)-norm of the vector \(Q x\) if \(p = 1, \infty\) or \(x' Q x\) if \(p = 2\). In (6.6) we have omitted the constraints \(u_k \in \mathcal{U}_{feas}(x_k), k = 1, \ldots, N\) assuming that they are implicit in the first constraints, i.e., if there exists no \(\tilde{C}^i\) that contains \([z_k]\) then this is an infeasible point.

We will use this implicit notation throughout the chapter.

Note that we distinguish between the input \(u(t)\) and the state \(x(t)\) of plant (6.3) at time \(t\) and the variables \(u_k\) and \(x_k\) of the optimization problem (6.6).

In the following, we will assume that \(Q = Q' \succeq 0, R = R' \succ 0, P \succeq 0, \) for \(p = 2\), and that \(Q, R, P\) are full column rank matrices for \(p = 1, \infty\). We will also denote by \(X_k \subseteq \mathbb{R}^{nc \times \{0,1\}^{me}}\) the set of states \(x_k\) that are feasible for (6.4)-(6.6):
\[
X_k = \left\{ x \in \mathbb{R}^{nc \times \{0,1\}^{me}}, \begin{array}{l}
\exists u \in \mathbb{R}^{mc \times \{0,1\}^{me}}, \\
\exists i \in \{1, \ldots, s\} \\
[z] \in \tilde{C}^i \text{ and } \\
A^i x_k + B^i u_k + f^i \in X_{k+1}\end{array} \right\}, \tag{6.7}
\]
\(X_N = X_f\).

Note that the optimizer function \(U^*_N\) may not be uniquely defined if the optimal set of problem (6.4)-(6.6) is not a singleton for some \(x(0)\).

In the following we need to distinguish between optimal control based on the 2-norm and optimal control based on the 1-norm or \(\infty\)-norm.

As a last remark, we want to point out that it is almost immediate to extend the results of the following sections to different formulations of hybrid optimal control problems, such as reference tracking problems or problems where penalties for switching between two different regions of operation are weighted in the cost function.

### 6.4 Solution Properties

**Theorem 6.1.** Consider the optimal control problem (6.4)-(6.6) with \(p = 2\) and let Assumption 6.1 hold. Then, there exists a solution in the form of a PWA state-feedback control
law

\[ u_k^i(x(k)) = F_k^i x(k) + G_k^i \]

if \( x(k) \in R_k^i \),

where \( R_k^i \), \( i = 1, \ldots, N_k \) is a partition of the set \( X_k \) of feasible states \( x(k) \), and the closure \( \bar{R}_k^i \) of the sets \( R_k^i \) has the following form:

\[
\bar{R}_k^i := \{ x : x(k)'L_k^i(j)x(k) + M_k^i(j)x(k) \leq N_k^i(j) , \]
\[
j = 1, \ldots, n_k^i \}, \quad k = 0, \ldots, N - 1,
\]

and

\[
x(k + 1) = A^i x(k) + B^i u_k^i(x(k)) + f^i
\]

if \( x(k) \in \bar{C}_i \), \( i = 1, \ldots, s \).

\[ J_{\bar{v}}(x(0)) := \min_{\bar{U}_N} J(U_N, x(0)) \]  

(6.11)

subject to

\[
x_{k+1} = A^{v_k^i} x_k + B^{v_k^i} u_k + f^{v_k^i}
\]

\[
[x_k] \in \bar{C}^{v_k^i}
\]

\[
k = 0, \ldots, N - 1
\]

(6.12)

\[
x_N \in X_f
\]

\[
x_0 = x(0)
\]

\begin{equation}
\end{equation}

Problem (6.11)-(6.12) is equivalent to a finite-time optimal control problem for a linear time-varying system with time-varying constraints and can be solved by using the approach of [BMDP02]. The first move \( u_0 \) of its solution is the PPWA feedback control law

\[ u_0^i(x(0)) = \bar{F}^{i,j} x(0) + \bar{G}^{i,j}, \quad \forall x(0) \in T^{i,j}, \quad j = 1, \ldots, N^{r,i} \]  

(6.13)

where \( D^i = \bigcup_{j=1}^{N^{r,i}} T^{i,j} \) is a polyhedral partition of the convex set \( D^i \) of feasible states \( x(0) \) for problem (6.11)-(6.12). \( N^{r,i} \) is the number of regions of the polyhedral partition of the solution and it is a function of the number of constraints in problem (6.11)-(6.12). The upper index \( i \) in (6.13) denotes that the input \( u_0^i(x(0)) \) is optimal when the switching sequence \( v_i \) is fixed.

The set \( X_0 \) of all feasible states at time 0 is \( X_0 = \bigcup_{i=1}^{s} D_i \) and in general it is not convex. Indeed, as some initial states can be feasible for different switching sequences, the sets
\[ D^i, \ i = 1, \ldots, s^N, \] in general, can overlap. The solution \( u_0^*(x(0)) \) to the original problem (6.4)-(6.6) can be computed in the following way. For every polyhedron \( T^{i,j} \) in (6.13),

1. If \( T^{i,j} \cap T^{l,m} = \emptyset \) for all \( l \neq i, \ l = 1, \ldots, s^N \), and for all \( m = 1, \ldots, N^{r_l} \), then the switching sequence \( v_i \) is the only feasible one for all the states belonging to \( T^{i,j} \) and therefore the optimal solution is given by (6.13), i.e.

\[
u_0^*(x(0)) = \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, \quad \forall x \in T^{i,j}.
\] (6.14)

2. If \( T^{i,j} \) intersects one or more polyhedra \( T^{l_1,m_1}, T^{l_2,m_2}, \ldots \), the states belonging to the intersection are feasible for more than one switching sequence \( v_i, v_{l_1}, v_{l_2}, \ldots \) and therefore the corresponding value functions \( J^*_v, J^*_{v_1}, J^*_{v_2}, \ldots \) in (6.11) have to be compared in order to compute the optimal control law.

Consider the simple case when only two polyhedra overlap, i.e. \( T^{i,j} \cap T^{l,m} := T^{(i,j),(l,m)} \neq \emptyset \). We will refer to \( T^{(i,j),(l,m)} \) as a double feasibility polyhedron. For all states belonging to \( T^{(i,j),(l,m)} \) the optimal solution is:

\[
u_0^*(x(0)) = \begin{cases} 
\tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, & \forall x(0) \in T^{(i,j),(l,m)} ; \\
\tilde{F}^{l,m} x(0) + \tilde{G}^{l,m}, & J^*_v(x(0)) < J^*_v(x(0)) ; \\
\tilde{F}^{i,j} x(0) + \tilde{G}^{i,j} \quad \text{or} \quad \tilde{F}^{l,m} x(0) + \tilde{G}^{l,m}, & J^*_v(x(0)) > J^*_v(x(0)) ; \\
\end{cases}
\] (6.15)

Because \( J^*_v \) and \( J^*_w \) are quadratic functions of \( x(0) \) on \( T^{i,j} \) and \( T^{l,m} \) respectively, we find the expression (6.9) of the control law domain. The sets \( T^{i,j} \setminus T^{l,m} \) and \( T^{l,m} \setminus T^{i,j} \) are two single feasibility non-Euclidean polyhedra which can be partitioned into a set of single feasibility polyhedra, and thus be described through (6.9) with \( L_k = 0 \).

In order to conclude the proof, the general case of \( n \) intersecting polyhedra has to be discussed. We follow three main steps. Step 1: generate one polyhedron of \( n^{th} \)-ple feasibility and \( 2^n - 2 \) polyhedra, generally non-Euclidean and possibly empty and disconnected, of single, double, \ldots, \( (n - 1)^{th} \)-ple feasibility. Step 2: the \( i^{th} \)-ple feasibility non-Euclidean polyhedron is partitioned into several \( i^{th} \)-ple feasibility polyhedra. Step 3: any \( i^{th} \)-ple feasibility polyhedron with \( i > 1 \) is further partitioned into at most \( i \) subsets (6.9) where in each one of them a certain feasible value function is greater than all the others. The procedure is depicted in Figure 6.1 when \( n = 3 \).

**Case 2: binary inputs, \( m_t \neq 0 \)**
The proof can be repeated in the presence of binary inputs, \( m_t \neq 0 \). In this case the switching sequences \( v_i \) are given by all combinations of region indices and binary inputs, i.e., \( i = 1, \ldots, (s \cdot m_t)^N \). The continuous component of the optimal input is given by (6.14) or (6.15). Such an optimal continuous component of the input has an associated optimal sequence \( v_i \), whose component provide the remaining binary components of the optimal input.
Figure 6.1: Graphical illustration of the main steps for the proof of Theorem 6.1 when 3 polyhedra intersect. Step 1: the three intersecting polyhedra are partitioned into: one polyhedron of triple feasibility (1,2,3), 2 polyhedra of double feasibility (1,2) and (1,3), 3 polyhedra of single feasibility (1),(2),(3). The sets (1), (2) and (1,2) are non-Euclidean polyhedra. Step 2: the sets (1), (2) and (1,2) are partitioned into six polyhedra of single feasibility. Step 3: value functions are compared inside the polyhedra of multiple feasibility.

**Case 3: binary states, \( n_l \neq 0 \)**

The proof can be repeated in the presence of binary states by a simple enumeration of all the possible \( n_l^N \) discrete state evolutions.

From the result of the theorem above one immediately concludes that the value function \( J^*_0 \) is piecewise quadratic:

\[
J^*_0(x(0)) = x(0)'H^1_1x(0) + H^2_2x(0) + H^3_3 \quad \text{if} \quad x(0) \in \mathcal{R}^i_0, \quad (6.16)
\]

The proof of Theorem 6.1 gives useful insights into the properties of the sets \( \mathcal{R}^i_k \) in (6.9). We will summarize them next.

Each set \( \mathcal{R}^i_k \) has an associated multiplicity \( j \) which means that \( j \) switching sequences are feasible for problem (6.4)-(6.6) starting from a state \( x(k) \in \mathcal{R}^i_k \). If \( j = 1 \), then \( \mathcal{R}^i_k \) is a polyhedron. In general, if \( j > 1 \) the boundaries of \( \mathcal{R}^i_k \) can be described either by an affine function or by a quadratic function. In the sequel boundaries which are described by quadratic functions but degenerate to hyperplanes or sets of hyperplanes will be considered affine boundaries.

**Quadratic** boundaries arise from the comparison of value functions associated with feasible switching sequences, thus a maximum of \( j - 1 \) quadratic boundaries can be present in a \( j \)-ple feasible set. The **affine** boundaries can be of three types. **Type a:** they are inherited from the original \( j \)-ple feasible non-Euclidean polyhedron. In this case beyond such boundaries the multiplicity of the feasibility changes. **Type b:** they are artificial cuts needed to describe the
original $j$-ple feasible non-Euclidean polyhedron as a set of $j$-ple feasible polyhedra. Beyond type $b$ boundaries the multiplicity of the feasibility does not change. Type $c$: they arise from the comparison of quadratic value functions which degenerate in an affine boundary.

In conclusion, we can state the following proposition

**Proposition 6.1.** The value function $J^*_k$

1. is a quadratic function of the states inside each $R^1_k$
2. is continuous on quadratic and affine boundaries of type $b$ and $c$
3. might be discontinuous only on affine boundaries of type $a$,

and the optimizer $u^*_k$

1. is an affine function of the states inside each $R^1_k$
2. is continuous and unique on affine boundaries of type $b$
3. is non-unique on quadratic boundaries, except possibly at isolated points.
4. might be non-unique on affine boundaries of type $c$,
5. might be discontinuous on affine boundaries of type $a$.

Based on Proposition 6.1 above one can highlight the only source of discontinuity of the value function: affine boundaries of type $a$. The following corollary gives a useful insight on the class of possible value functions.

**Corollary 6.1.** $J^*_0$ is a lower-semicontinuous PWQ function on $X_0$.

**Proof** The proof follows from the result on the minimization of lower-semicontinuous point-to-set maps in [Ber97]. Below we give a simple proof without introducing the notion of point-to-set maps.

Only points where a discontinuity occurs are relevant for the proof, i.e., states belonging to boundaries of type $a$. From Assumption 6.1 it follows that the feasible switching sequences for a given state $x(0)$ are all the feasible switching sequences associated with any set $R^j_0$ whose closure $\bar{R}^j_0$ contains $x(0)$. Consider a state $x(0)$ belonging to boundaries of type $a$ and the proof of Theorem 6.1. The only case of discontinuity can occur when

(i) a $j$-ple feasible set $P_1$ intersects an $i$-ple feasible set $P_2$ with $i < j$, (ii) it exists a point $x(0) \in P_1$, $P_2$ and a neighbor $N(x(0))$ with $x, y \in N(x(0))$, $x \in P_1$, $x \notin P_2$ and $y \in P_2$, $y \notin P_1$. The proof follows from the previous statements and the fact that $J^*_0(x(0))$ is the minimum of all $J^*_v(x(0))$ for all feasible switching sequences $v_i$.

The result of Corollary 6.1 will be extensively used in the next sections. Even if value function and optimizer are discontinuous, one can work with the closure $\bar{R}^j_k$ of the original sets $R^j_k$ without explicitly considering their boundaries. In fact, if a given state $x(0)$ belongs to several regions $\bar{R}^1_0, \ldots, \bar{R}^p_0$, then the minimum value among the optimal values (6.16)
associated with each region $\mathcal{R}_0^1, \ldots, \mathcal{R}_0^p$ allow us to identify the region of the set $\mathcal{R}_0^1, \ldots, \mathcal{R}_0^p$ containing $x(0)$.

Next we show some interesting properties of the optimal control law when we restrict our attention to smaller classes of PWA systems.

**Corollary 6.2.** Assume that the PWA system (6.3) is continuous, and that $E = 0$ in (6.2) and $X_f = \mathbb{R}^n$ in (6.6) (which means that there are no state constraints, i.e., $P$ is unbounded in the $x$-space). Then, the value function $J_0^*$ in (6.6) is continuous.

**Proof** Problem (6.4)-(6.6) becomes a multi-parametric program with only input constraints when the state at time $k$ is expressed as a function of the state at time 0 and the input sequence $u_0, \ldots, u_{k-1}$, i.e., $x_k = f_{\text{PWA}}(\cdots (f_{\text{PWA}}(x_0, u_0), u_1), \ldots, u_{k-2}, u_{k-1})$. $J$ in (6.4) will be a continuous function of $x_0$ and $u_0, \ldots, u_{N-1}$ since it is the composition of continuous functions. The input constraints on $u_0, \ldots, u_{N-1}$ are convex by assumption. The proof follows from the continuity of $J$ and Theorem 4.1.

Note that $E = 0$ is a sufficient condition for ensuring that constraints (6.2) are convex in the optimization variables $u_0, \ldots, u_n$. In general, even for continuous PWA systems with state constraints it is difficult to find weak assumptions ensuring the continuity of the value function $J_0^*$. Ensuring the continuity of the optimal control law $u(k) = u_k^*(x(k))$ is even more difficult. A list of sufficient conditions for $U_N^*$ to be continuous can be found in [Fia76].

In general, they require the convexity (or a relaxed form of it) of the cost $J(U_N, x(0))$ in $U_N$ for each $x(0)$ and the convexity of the constraints in (6.6) in $U_N$ for each $x(0)$. Such conditions are clearly very restrictive since the cost and the constraints in problem (6.6) are a composition of quadratic and linear functions, respectively, with the piecewise affine dynamics of the system.

The next theorem provides a condition under which the solution $u_k^*(x(k))$ of the optimal control problem (6.4)-(6.6) is a PWA state-feedback control law.

**Theorem 6.2.** Assume that the optimizer $U_N^*(x(0))$ of (6.4)-(6.6) is unique for all $x(0)$. Then the solution to the optimal control problem (6.4)-(6.6) is a PWA state feedback control law of the form

$$u_k^*(x(k)) = F_k^ix(k) + G_k^i \quad \text{if } x(k) \in \mathcal{P}_k^i \quad k = 0, \ldots, N - 1, \quad (6.17)$$

where $\mathcal{P}_k^i$, $i = 1, \ldots, N_k^*$, is a polyhedral partition of the set $X_k$ of feasible states $x(k)$.

**Proof :** In Proposition 6.1 we concluded that the value function $J_0^*(x(0))$ is continuous on quadratic type boundaries. By hypothesis, the optimizer $u_0^*(x(0))$ is unique. Theorem 6.1 implies that $F_k^i x(0) + G_k^i = F_k^m x(0) + G_k^m$, $\forall x(0)$ belonging to the quadratic boundary. This can occur only if the quadratic boundary degenerates to a single feasible point or to affine boundaries. The same arguments can be repeated for $u_k^*(x(k))$, $k = 1, \ldots, N - 1$. □

**Remark 6.1.** Theorem 6.2 relies on a rather strong uniqueness assumption. Sometimes, problem (6.4)-(6.6) can be modified in order to obtain uniqueness of the solution and use the result of Theorem 6.2 which excludes the existence of non-convex ellipsoidal sets. It is reasonable to believe that there are other conditions under which the state-feedback solution is PPWA without claiming uniqueness.
The previous results can be extended to piecewise linear cost functions, i.e., cost functions based on the $1$-norm or the $\infty$-norm.

**Theorem 6.3.** Consider the optimal control problem (6.4)-(6.6) with $p = 1, \infty$ and let Assumption 6.1 hold. Then there exists a solution in the form of a PPWA state-feedback control law

\[ u^*_k(x(k)) = F^i_k x(k) + G^i_k \quad \text{if} \quad x(k) \in \mathcal{P}^i_k, \]  

where $\mathcal{P}^i_k, i = 1, \ldots, N^r_k$ is a polyhedral partition of the set $\mathcal{X}_k$ of feasible states $x(k)$.

**Proof** The proof is similar to the proof of Theorem 6.1. Fix a certain switching sequence $v_i$, consider the problem (6.4)-(6.6) and constrain the state to switch according to the sequence $v_i$ to obtain problem (6.11)-(6.12). Problem (6.11)-(6.12) can be viewed as a finite time optimal control problem with a performance index based on $1$-norm or $\infty$-norm for a linear time varying system with time varying constraints and can be solved by using the multi-parametric linear program as described in [Bor03]. Its solution is a PPWA feedback control law

\[ u^i_0(x(0)) = \tilde{F}^{i,j} x(0) + \tilde{G}^{i,j}, \quad \forall x \in T^{i,j}, \quad j = 1, \ldots, N^{ri}, \]  

and the value function $J^*_v$ is piecewise affine on polyhedra and convex. The rest of the proof follows the proof of Theorem 6.1. Note that in this case the value functions to be compared are piecewise affine and not piecewise quadratic. \hfill $\square$

By comparing Theorem 6.1 and Theorem 6.3 it is clear that, while for performance indices based on $1$ or $\infty$ norms the solution is piecewise affine on polyhedra, in the $2$-norm case one may need to deal with nonconvex ellipsoidal regions.

### 6.5 Computation of the Optimal Control Law via Dynamic Programming

In the previous section the properties enjoyed by the solution of hybrid optimal control problems were investigated. The proof of Theorem 6.1 is constructive, but it is based on the enumeration of all the possible switching sequences of the hybrid system, the number of which grows exponentially with the time horizon. Although the computation is performed off-line (the on-line complexity is the one associated with the evaluation of the PWA control law (6.17)), more efficient methods than enumeration are desirable.

In [BM99] the main idea is to translate problem (6.4)-(6.6) into a linear or quadratic mixed integer program that can be solved by using standard commercial software. This approach does not provide the state-feedback law (6.8) or (6.18) but only the optimal control sequence $U^*_N(x(0))$ for a given initial state $x(0)$. In [Bor03] the state-feedback law (6.8) or (6.18) is computed by means of multi-parametric mixed integer programming. However, the use of multi-parametric mixed integer programming has a major drawback: the solver does not exploit the structure of the optimal control problem. In fact, a large part of the information associated with problem (6.4)-(6.6) becomes hidden when it is reformulated as a mixed integer program. In this section we show how linear and quadratic parametric programming can be used to solve the Hamilton-Jacobi-Bellman equations associated with
6 Constrained Finite Time Optimal Control of PWA Systems

CFTOC problem (6.4)–(6.6). In [BCM03a] we have compared the dynamic programming and the mixed-integer multi-parametric programming approach.

The PWA solution (6.8) will be computed proceeding backwards in time using two tools: a linear or quadratic multi-parametric programming solver (depending on the cost function used) and a special technique to store the solution which will be illustrated in the next sections. The algorithm will be presented for optimal control based on a quadratic performance criterion. Its extension to optimal control based on linear performance criteria is straightforward.

6.5.1 Preliminaries and Basic Steps

Consider the PWA map \( \zeta \) defined as

\[
\zeta : x \in \mathcal{R}_i \mapsto F_i x + G_i \quad \text{for} \quad i = 1, \ldots, N_R,
\]

where \( \mathcal{R}_i, i = 1, \ldots, N_R \) are subsets of the \( x \)-space. Note that if there exist \( l, m \in \{1, \ldots, N_R\} \) such that for \( x \in \mathcal{R}_l \cap \mathcal{R}_m, F_l x + G_l \neq F_m x + G_m \) the map \( \zeta \) (6.20) is not single valued.

**Definition 6.2.** Given a PWA map (6.20) we define \( f_{PWA}(x) = \zeta_o(x) \) as the ordered region single-valued function associated with (6.20) when

\[
\zeta_o(x) = \begin{cases} 
F_j x + G_j & \text{if} \quad x \in \mathcal{P}_1 \\
\vdots & \\
F_{N_R} x + G_{N_R} & \text{if} \quad x \in \mathcal{P}_{N_R}.
\end{cases} \]

and write it in the following form

\[
\zeta_o(x) = \begin{cases} 
F_1 x + G_1 & \text{if} \quad x \in \mathcal{P}_1 \\
\vdots & \\
F_{N_R} x + G_{N_R} & \text{if} \quad x \in \mathcal{P}_{N_R}.
\end{cases}
\]

Note that given a PWA map (6.20) the corresponding ordered region single-valued function \( \zeta_o \) changes if the order used to store the regions \( \mathcal{R}_i \) and the corresponding affine gains change. For illustration purposes consider the example depicted in Figure 6.2, where \( x \in \mathbb{R}, N_R = 2, F_1 = 0, G_1 = 0, \mathcal{R}_1 = [-2, 1], F_2 = 1, G_2 = 0, \mathcal{R}_2 = [0, 2] \).

In the following we assume that the sets \( \mathcal{R}_i^k \) in the optimal solution (6.8) can overlap. When we refer to the PWA function \( u_i^*(x(k)) \) in (6.8) we will implicitly mean the ordered region single-valued function associated with the mapping (6.8).

**Example 6.1.** Let \( J_1^* : \mathcal{P}_1 \to \mathbb{R} \) and \( J_2^* : \mathcal{P}_2 \to \mathbb{R} \) be two quadratic functions, \( J_1^*(x) := x'L_1 x + M_1 x + N_1 \) and \( J_2^*(x) := x'L_2 x + M_2 x + N_2 \), where \( \mathcal{P}_1 \) and \( \mathcal{P}_2 \) are convex polyhedra and \( J_i^*(x) = +\infty \) if \( x \notin \mathcal{P}_i, i \in \{1, 2\} \). Let \( u_1^* : \mathcal{P}_1 \to \mathbb{R}^m, u_2^* : \mathcal{P}_2 \to \mathbb{R}^m \) be vector functions. Let \( \mathcal{P}_1 \cap \mathcal{P}_2 := \mathcal{P}_3 \neq \emptyset \) and define

\[
J^*(x) := \min \{ J_1^*(x), J_2^*(x) \}
\]

\[
u^*(x) := \begin{cases} 
u_1^*(x) & \text{if} \quad J_1^*(x) \leq J_2^*(x) \\
u_2^*(x) & \text{if} \quad J_1^*(x) \geq J_2^*(x).
\end{cases}
\]
where $u^*(x)$ can be a set valued function. Let $L_3 = L_2 - L_1$, $M_3 = M_2 - M_1$, $N_3 = N_2 - N_1$. Then, corresponding to the three following cases

(i) $J_1^*(x) \leq J_2^*(x) \forall x \in \mathcal{P}_3$

(ii) $J_1^*(x) \geq J_2^*(x) \forall x \in \mathcal{P}_3$

(iii) $\exists x_1, x_2 \in \mathcal{P}_3 | J_1^*(x_1) < J_2^*(x_1)$ and $J_1^*(x_2) > J_2^*(x_2)$

the expressions (6.21) and a real-valued function that can be extracted from (6.22) can be written equivalently as:

1.  
   
   \[
   J^*(x) = \begin{cases}  
   J_1^*(x) & \text{if } x \in \mathcal{P}_1 \\  
   J_2^*(x) & \text{if } x \in \mathcal{P}_2  
   \end{cases}  
   \]  
   \[
   u^*(x) = \begin{cases}  
   u_1^*(x) & \text{if } x \in \mathcal{P}_1 \\  
   u_2^*(x) & \text{if } x \in \mathcal{P}_2  
   \end{cases}  
   \]  

2. as in (6.23) and (6.24) by switching the indices 1 and 2
3.

\[
J^*(x) = \begin{cases} 
\min \{J_1^*(x), J_2^*(x)\} & \text{if } x \in \mathcal{P}_3 \\
J_1^*(x) & \text{if } x \in \mathcal{P}_1 \\
J_2^*(x) & \text{if } x \in \mathcal{P}_2 
\end{cases}
\]  

(6.25)

\[
u^*(x) = \begin{cases} 
u_1^*(x) & \text{if } x \in \mathcal{P}_3 \cap \{x \mid x' L_3 x + M_3 x + N_3 \geq 0\} \\
u_2^*(x) & \text{if } x \in \mathcal{P}_3 \cap \{x \mid x' L_3 x + M_3 x + N_3 \leq 0\} \\
u_3^*(x) & \text{if } x \in \mathcal{P}_3 \\
u_4^*(x) & \text{if } x \in \mathcal{P}_2 
\end{cases}
\]  

(6.26)

where (6.23), (6.24), (6.25), and (6.26) have to be considered as PWA and PPWQ functions in the ordered region sense.

Example 6.1 shows how to

- avoid the storage of the intersections of two polyhedra in case (i) and (ii)
- avoid the storage of possibly non convex regions \(\mathcal{P}_1 \setminus \mathcal{P}_3\) and \(\mathcal{P}_2 \setminus \mathcal{P}_3\)
- work with multiple quadratic functions instead of quadratic functions defined over non-convex and non-polyhedral regions.

The three point listed above will be the three basic ingredients for storing and simplifying the optimal control law (6.8). Next we will show how to compute it.

**Remark 6.2.** To distinguish between cases (i), (ii) and (iii) of Example 6.1, in general one needs to solve an indefinite quadratic program, namely

\[
\min_x x' L_3 x + M_3 x + N_3 \\
\text{subj. to } x \in \mathcal{P}_3.
\]

(6.27)

In our approach, to avoid such a test the form (6.26) corresponding to case (iii) can be used. The only drawback is that the form (6.26) is, in general, a non-minimal representation of the value function and therefore it increases the complexity of evaluating and storing the optimal control profile (6.8).

### 6.5.2 Multiparametric Programming with Multiple Quadratic Functions

Consider the multi-parametric program

\[
J^*(x) := \min_u l(x, u) + q(f(x, u)) \\
\text{s.t. } f(x, u) \in \mathcal{P},
\]

(6.28)

where \(\mathcal{P} \subseteq \mathbb{R}^n\) is a compact set, \(f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n\), \(q : \mathcal{P} \rightarrow \mathbb{R}\), and \(l : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}\) is a convex quadratic function of \(x\) and \(u\). We aim at determining the region \(\mathcal{X}\) of variables \(x\) such that the program (6.28) is feasible and the optimum \(J^*(x)\) is finite, and at finding the
expression $u^*(x)$ of (one of) the optimizer(s). We point out that the constraint $f(x, u) \in \mathcal{P}$
implies a constraint on $u$ as a function of $x$ since $u$ can assume only values where $f(x, u)$ is
defined.

Next we show how to solve several forms of problem (6.28).

**Lemma 6.1** (one to one problem). Problem (6.28) where $f$ is linear, $q$ is quadratic and
strictly convex, and $\mathcal{P}$ is a polyhedron can be solved by one mp-QP.

**Proof** See [BMDP02]

**Lemma 6.2** (one to one problem of multiplicity $d$). Problem (6.28) where $f$ is linear, $q$
is a multiple quadratic function of multiplicity $d$ and $\mathcal{P}$ is a polyhedron can be solved by $d$
mp-QP’s.

**Proof** The multi-parametric program to be solved is

$$J^*(x) = \min_u \left\{ l(x, u) + \min \{ q_1(f(x, u)), \ldots, q_d(f(x, u)) \} \right\}$$

(subj. to $f(x, u) \in \mathcal{P}$)

and it is equivalent to

$$J^*(x) = \min \left\{ \begin{array}{l}
\min_u l(x, u) + q_1(f(x, u)), \\
\quad \text{subj. to } f(x, u) \in \mathcal{P} \\
\quad \vdots \\
\min_u l(x, u) + q_d(f(x, u)) \\
\quad \text{subj. to } f(x, u) \in \mathcal{P}
\end{array} \right\}.$$  

(6.30)

The $i^{th}$ sub-problems in (6.30)

$$J^*_i(x) := \min_u \left\{ l(x, u) + q_i(f(x, u)) \right\}$$

(subj. to $f(x, u) \in \mathcal{P}$)

(6.31)

(6.32)

is a one to one problem and therefore it is solvable by an mp-QP. Let the solution of the $i$-th
mp-QPs be $u^i(x) = \tilde{F}^{i,j} x + \tilde{G}^{i,j}$, $\forall x \in \mathcal{T}^{i,j}$, $j = 1, \ldots, N^{r_i}$

(6.33)

where $\mathcal{T}^i = \bigcup_{j=1}^{N^{r_i}} \mathcal{T}^{i,j}$ is a polyhedral partition of the convex set $\mathcal{T}^i$ of feasible $x$ for the $i$th
sub-problem and $N^{r_i}$ is the corresponding number of polyhedral regions. The feasible set $\mathcal{X}$ satisfies $\mathcal{X} = \mathcal{T}^1 = \ldots = \mathcal{T}^d$ since the constraints of the $d$ sub-problems are identical.

The solution $u^*(x)$ to the original problem (6.29) is obtained by comparing and storing the
solution of $d$ mp-QP subproblems (6.31)-(6.32) as explained in Example 6.1. Consider the
case $d = 2$, and consider the intersection of the polyhedra $\mathcal{T}^{1,i}$ and $\mathcal{T}^{2,l}$ for $i = 1, \ldots, N^{r_1}$,
l = 1, \ldots, $N^{r_2}$. For all $\mathcal{T}^{1,i} \cap \mathcal{T}^{2,l} := \mathcal{T}^{(1,i),(2,l)} \neq \emptyset$ the optimal solution is stored in an ordered way as described in Example 6.1, while paying attention to the fact that a region could be already stored. Moreover, when storing a new polyhedron with the corresponding
value function and optimizer, the relative order of the regions already stored must not be changed. The result of this Intersect and Compare procedure is

\[ u^*(x) = F^i x + G^i \text{ if } x \in \mathcal{R}^i, \]
\[ \mathcal{R}^i := \{ x : x' L^i(j) x + M^i(j) x \leq N^i(j), \ j = 1, \ldots, n^i \}, \]

where \( \mathcal{R} = \bigcup_{j=1}^{N_R} \mathcal{R}^j \) is a polyhedron and the value function

\[ J^*(x) = \tilde{J}^*_j(x) \text{ if } x \in \mathcal{D}^j, j = 1, \ldots, N^D, \]

where \( \tilde{J}^*_j(x) \) are multiple quadratic functions defined over the convex polyhedra \( \mathcal{D}^j \). The polyhedra \( \mathcal{D}^j \) can contain several regions \( \mathcal{R}^i \) or can coincide with one of them. Note that (6.34) and (6.35) have to be considered as PWA and PPWQ functions in the ordered region sense.

If \( d > 2 \) then the value function in (6.35) is intersected with the solution of the third mp-QP sub-problem and the procedure is iterated by making sure not to change the relative order of the polyhedra and corresponding gain of the solution constructed in the previous steps. The solution will still have the same form (6.34)–(6.35).

Lemma 6.3 (one to r problem). Problem (6.28) where \( f \) is linear, \( q \) is a lower-semicontinuous PPWQ function defined over \( r \) polyhedral regions and strictly convex on each polyhedron, and \( \mathcal{P} \) is a polyhedron, can be solved by \( r \) mp-QP’s

Proof Let \( q(x) := q_i \) if \( x \in \mathcal{P}_i \) the PWQ function where the closures \( \bar{\mathcal{P}}_i \) of \( \mathcal{P}_i \) are polyhedra and \( q_i \) strictly convex quadratic functions. The multi-parametric program to solve is

\[ J^*(x) = \min \begin{cases} \min_u l(x, u) + q_1(f(x, u)), \\ \text{subj. to } f(x, u) \in \bar{\mathcal{P}}_1 \\ f(x, u) \in \mathcal{P} \\ \vdots \\ \min_u l(x, u) + q_r(f(x, u))) \\ \text{subj. to } f(x, u) \in \bar{\mathcal{P}}_r \\ f(x, u) \in \mathcal{P} \end{cases}. \]

The proof follows the lines to the proof of the previous theorem with the exception that the constraints of the \( i \)-th mp-QP subproblem differ from the one of the \( j \)-th mp-QP subproblem, \( i \neq j \).

The lower-semicontinuity assumption on \( q(x) \) allows one to use the closure of the sets \( \mathcal{P}_i \) in (6.36). The cost function in problem (6.28) is lower-continuous since it is a composition of a lower-semicontinuous function and a continuous function. Then, since the domain is compact, problem (6.36) admits a minimum. Therefore for a given \( x \), there exists one mp-QP in problem (6.36) which yields the optimal solution. There might exist other mp-QP solutions in (6.36) feasible at \( x \) being that are neither optimal nor feasible for the original problem (6.28). However, since \( q(x) \) is lower-semicontinuous, such solutions will be discarded.
when the corresponding value functions are compared. The procedure based on solving mp-QPs and storing the results as in Example 6.1 will be the same as in Lemma 6.2 but the domain \( \mathcal{R} = \bigcup_{j=1}^{N} \mathcal{R}^j \) of the solution can be a non-Euclidean polyhedron.

If \( f \) is PPWA defined over \( s \) regions then we have a \( s \) to \( X \) problem where \( X \) can belong to any of the problems listed above. In particular, we have an \( s \) to \( r \) problem of multiplicity \( d \) if \( f \) is PPWA and defined over \( s \) regions and \( q \) is a multiple PPWQ function of multiplicity \( d \), defined over \( r \) polyhedral regions. The following lemma can be proven along the lines of the proofs given before.

**Lemma 6.4.** Problem (6.28) where \( f \) is linear and \( q \) is a lower-semicontinuous PPWQ function of multiplicity \( d \), defined over \( r \) polyhedral regions and strictly convex on each polyhedron, is a one to \( r \) problem of multiplicity \( d \) and can be solved by \( r \cdot d \) mp-QP’s.

An \( s \) to \( r \) problem of multiplicity \( d \) can be decomposed into \( s \) one to \( r \) problems of multiplicity \( d \). An \( s \) to one problem can be decomposed into \( s \) one to one problems.

### 6.5.3 Algorithmic Solution of the HJB Equations

In the following we will substitute the CPWA system equations (6.3) with the shorter form

\[
x(k + 1) = \tilde{f}_{PWA}(x(k), u(k))
\]  

where \( \tilde{f}_{PWA} : \tilde{\mathcal{C}} \to \mathbb{R}^n \) and \( \tilde{f}_{PWA}(x, u) = A^i x + B^i u + f^i \) if \( [\check{z}] \in \mathcal{\tilde{C}}^i, i = 1, \ldots, s \), and \( \{\mathcal{\tilde{C}}^i\} \) is a polyhedral partition of \( \mathcal{\tilde{C}} \).

Consider the dynamic programming formulation of the CFTOC problem (6.4)-(6.6),

\[
J^*_j(x(j)) := \min_{u_j} \|Q x_j\|_2 + \|R u_j\|_2 + J^*_{j+1}(\tilde{f}_{PWA}(x(j), u_j))
\]  

subject to \( \tilde{f}_{PWA}(x(j), u_j) \in \mathcal{X}_{j+1} \)

for \( j = N - 1, \ldots, 0 \), with terminal conditions

\[
\mathcal{X}_N = \mathcal{X}_f
\]  

\[
J^*_N(x) = \|P x\|_2
\]

where \( \mathcal{X}_j \) is the set of all states \( x(j) \) for which problem (6.38)-(6.39) is feasible:

\[
\mathcal{X}_j = \{x \in \mathbb{R}^n | \exists u, \tilde{f}_{PWA}(x, u) \in \mathcal{X}_{j+1}\}.
\]

Equations (6.38)-(6.42) are the discrete-time version of the well known Hamilton-Jacobi-Bellman equations for continuous-time optimal control problems.

Assume for the moment that there are no binary inputs and binary states, \( m_\ell = n_\ell = 0 \). The HJB equations (6.38)-(6.41) can be solved backwards in time by using a multi-parametric quadratic programming solver and the results of the previous section.
Consider the first step of the dynamic program (6.38)–(6.41)

\[
J_{N-1}^*(x_{N-1}) := \min_{\{u_{N-1}\}} \|Qx_{N-1}\|_2 + \|Ru_{N-1}\|_2 + J_N^*(\tilde{f}_{\text{PWA}}(x_{N-1}, u_{N-1}))
\]

\[
\text{subj. to } \tilde{f}_{\text{PWA}}(x_{N-1}, u_{N-1}) \in \mathcal{X}_f.
\]

The cost to go function \(J_N^*(x)\) in (6.43) is quadratic, the terminal region \(\mathcal{X}_f\) is a polyhedron and the constraints are piecewise affine. Problem (6.43)–(6.44) is an \textit{s to one problem} that can be solved by solving \textit{s mp-QPs} (Lemma 6.4). From the second step \(j = N - 2\) to the last one \(j = 0\) the cost to go function \(J_{j+1}^*(x)\) is a lower-semicontinuous PPWQ with a certain multiplicity \(d_{j+1}\), the terminal region \(\mathcal{X}_{j+1}\) is a polyhedron (in general non-Euclidean) and the constraints are piecewise affine. Therefore, problem (6.38)–(6.41) is an \textit{s to \(N_{j+1}^r\) problem with multiplicity} \(d_{j+1}\) (where \(N_{j+1}^r\) is the number of polyhedra of the cost to go function \(J_{j+1}^*\)), that can be solved by solving \(sN_{j+1}^rd_{j+1}\) mp-QPs (Lemma 6.4). The resulting optimal solution will have the form (6.8) considered in the ordered region sense.

In the presence of binary inputs the procedure can be repeated, with the difference that all the possible combinations of binary inputs must be enumerated. Therefore a \textit{one to one problem} becomes a \(2^m\) \textit{to one problem} and so on. In the presence of binary states the procedure can be repeated either by enumerating them all or by solving a dynamic programming algorithm at time step \(k\) from a relaxed state space to the set of binary states feasible at time \(k + 1\).

Next we summarize the main steps of the dynamic programming algorithm discussed in this section. We use boldface characters to denote sets of polyhedra, i.e., \(\mathbf{R} := \{\mathcal{R}_i\}_{i=1,\ldots,|\mathbf{R}|}\), where \(\mathcal{R}_i\) is a polyhedron and \(|\mathbf{R}|\) is the cardinality of the set \(\mathbf{R}\). Furthermore, when we say \textbf{SOLVE} an mp-QP we mean to compute and store the triplet \(S_{k,i,j}\) of expressions for the value function, the optimizer, and the polyhedral partition of the feasible space.
Algorithm 6.1.

**INPUT** CFTOC problem (6.4)—(6.6)

**OUTPUT** Solution (6.8) in the ordered region sense.

**LET** \( R_N = \{ X_j \} \)
**LET** \( J_{N,1}^*(x) := x'Px \)

**FOR** \( k = N-1, \ldots, 1, \)
**FOR** \( i = 1, \ldots, |R_{k+1}|, \)
**FOR** \( j = 1, \ldots, s, \)

**LET** \( S_{k,i,j} = \{ \} \)

**SOLVE** the mp-QP

\[
S_{k,i,j} \leftarrow \min_{u_k} \quad x_k'Qx_k + u_k'Ru_k + J_{k+1,i}^* (A_j x_k + B_j u_k + f_j)
\]

\[
\text{subj. to} \quad \begin{cases} 
A_j x_k + B_j u_k + f_j \in R_{k+1,i} \\
[ x_k' ] \in C^j.
\end{cases}
\]

**END**

**END**

**LET** \( R_k = \{ R_{k,i,j,l} \}_{i,j,l} \). Denote by \( R_{k,h} \) its elements, and by \( J_{k,h}^* \) and \( u_{k,h}^*(x) \) the associated costs and optimizers, with \( h \in \{ 1, \ldots, |R_k| \} \)

**KEEP** only triplets \( (J_{k,h}^*(x), u_{k,h}^*(x), R_{k,h}) \) for which

\( \exists x \in R_{k,h} : x \notin R_{k,d}, \forall d \neq h \ \text{\quad OR} \)

\( \exists x \in R_{k,h} : J_{k,h}^*(x) < J_{k,d}^*(x), \forall d \neq h \)

**CREATE** multiplicity information and additional regions for an ordered region solution as explained in Example 6.1

**END.** □

In Algorithm 6.1, the structure \( S_{k,i,j} \) stores the matrices defining quadratic function \( J_{k,i,j,l}^*(\cdot) \), affine function \( u_{k,i,j,l}^*(\cdot) \), and polyhedra \( R_{k,i,j,l} \), for all \( l \):

\[
S_{k,i,j} = \bigcup_l \{ (J_{k,i,j,l}^*(x), u_{k,i,j,l}^*(x), R_{k,i,j,l}) \}.
\]  

(6.45)

where the indices in (6.45) have the following meaning: \( k \) is the time step, \( i \) indexes the piece of the “cost-to-go” function that the DP algorithm is considering, \( j \) indexes the piece of the PWA dynamics the DP algorithm is considering, and \( l \) indexes the polyhedron in the mp-QP solution of the \((k, i, j)\)th mp-QP problem.

Step 11 of Algorithm 6.1 aims at discarding regions \( R_{k,h} \) that are completely covered by some other regions that have lower cost. Obviously, if there are some parts of the region \( R_{k,h} \) that are not covered at all by other regions (first condition) we need to keep it. Note
that comparing the cost functions (second condition) is, in general, non-convex optimization problem. One might consider solving the problem exactly, but since algorithm works even if some removable regions are kept, we usually formulate LMI relaxation of the problem at hand. While executing Step 11 of Algorithm 6.1 we can simultaneously obtain the information of multiplicity of polyhedral subsets of the region $\mathcal{R}_{k,h}$.

The output of Algorithm 6.1 is the state-feedback control law (6.8) considered in the ordered region sense. The online implementation of the control law requires simply the evaluation of the PWA controller (6.8) in the ordered region sense (note that the order the solution is stored is important).

### 6.6 Discontinuous PWA systems

Without Assumption 6.1 the optimal control problem (6.4)-(6.6) may be feasible but may not admit an optimizer for some $x(0)$ (the problem in this case should be to find an infimum rather than the minimum).

Under the assumption that the optimizer exists for all states $x(k)$, the approach explained in the previous sections can be applied to discontinuous systems by considering three elements. First, the PWA system (6.3) has to be defined on each polyhedron of its domain and all its lower dimensional facets. Secondly, dynamic programming has to be performed “from” and “to” any lower dimensional facet of each polyhedron of the PWA domain. Finally, value functions are not lower-semicontinuous, which implies that Lemma 6.3 cannot by used. Therefore, when considering the closure of polyhedral domains in multi-parametric programming (6.36), a post-processing is necessary in order to remove multi-parametric optimal solutions which do not belong to the original set but only to its closure. The tedious details of the dynamic programming algorithm for discontinuous PWA systems are not included here but can be immediately extracted from the results of the previous sections.

In practice, the approach just described for discontinuous PWA systems can easily be numerically prohibitive. The simplest approach from a practical point of view resorts to introducing gaps between the boundaries of any two polyhedra belonging to the PWA domain (or, equivalently, to shrinking by a quantity $\varepsilon$ the size of every polyhedron of the original PWA system). In this way, one deals with PWA systems defined over a disconnected union of closed polyhedra. By doing so, one can use the approach discussed previously in this chapter for continuous PWA systems. However, the optimal controller will not be defined at the points in the gaps and at points when the only feasible solution is in the gaps. Also, the computed solution might be arbitrarily different from the original solution to problem (6.4)-(6.6) at any feasible point $x$. Despite this, if the dimension $\varepsilon$ of the gaps is close to the machine precision and comparable to sensor/estimation errors, such an approach is very appealing in practice. To the best of our knowledge in some cases this approach is the only computationally tractable for computing controllers for discontinuous hybrid systems fulfilling state and input constraints that are implementable in real-time.

Without Assumption 6.1, problem (6.4)-(6.6) is well defined only if an optimizer exists for all $x(0)$. In general, this is not easy to check. The dynamic programming algorithm described here could be used for such a test but the details are not included in this thesis.
6.7 Conclusion

For discrete-time linear hybrid systems, we have described an off-line procedure to synthesize optimal control laws based on the minimization of quadratic and linear performance indices subject to linear constraints on inputs and states. The procedure is based on a combination of dynamic programming and multi-parametric quadratic programming. In collaboration with different companies and institutes, the results described in this paper have been applied to a wide range of problems ([FTGS02, BM99, TB04, BGKH02, BBM03a, BBFH01, BVMP03, MBM03]). Simple examples that highlight the main features of the hybrid system approach presented in this paper can be found in [BBBM03a].
Constrained (In-)Finite Time Optimal Control of PWA Systems with a Linear Performance Index

We consider the constrained finite and infinite time optimal control problem for the class of discrete-time linear hybrid systems. When a linear performance index is used the finite and infinite time optimal solution is a piecewise affine state feedback control law. In this chapter we present algorithms that compute the optimal solution to both problems in a computationally efficient manner and with guaranteed convergence and error bounds. Both algorithms combine a dynamic programming exploration strategy with multi-parametric linear programming and basic polyhedral manipulation.

7.1 Introduction

In the last few years several different techniques have been developed for the analysis and controller synthesis for hybrid systems [Son81, LTS99, BZ00, BMM00c, Bor03, Joh03]. A significant amount of the research in this field has focused on solving constrained optimal control problems, both for continuous-time and discrete-time hybrid systems.

We consider the class of discrete-time linear hybrid systems. In particular the class of constrained piecewise affine (PWA) systems that are obtained by partitioning the extended state-input space into polyhedral regions and associating with each region a different affine state update equation, cf. [Son81, HDB01]. As shown in [HDB01], the class of piecewise affine systems is of rather general nature and equivalent to many other hybrid system formalisms, such as for example mixed logical dynamical systems or linear complementary systems.

For piecewise affine systems the constrained finite time optimal control (CFTOC) problem can be solved by means of multi-parametric programming [Bor03]. The solution is a piecewise affine state feedback control law and can be computed by using multi-parametric mixed-integer quadratic programming (mp-MIQP) for a quadratic performance index and multi-parametric mixed-integer linear programming (mp-MILP) for a linear performance index, cf. [Bor03, DP00].

As recently shown by Borrelli et al. [BBBM03a] for a quadratic performance index and by [BCM03a, KM02] for a linear performance index, it is possible to obtain the optimal solution
to the CFTOC problem without the use of integer programming. In [BBBM03a, BCM03a] the authors propose efficient algorithms based on a dynamic programming strategy combined with multi-parametric quadratic or linear program (mp-QP or mp-LP) solvers.

However, stability and feasibility (constraint satisfaction) of the closed-loop system are not guaranteed if the solution to the CFTOC problem is used in a receding horizon control strategy. To remedy this deficiency various schemes have been proposed in the literature. For constrained linear systems stability can be (artificially) enforced by introducing ‘proper’ terminal set constraints and/or a terminal cost to the formulation of the CFTOC problem [MRRS00]. For the class of constrained PWA systems very few and restrictive stability criteria are known, e.g. [BM99, MRRS00]. Only recently ideas used for enforcing closed-loop stability of the CFTOC problem for constrained linear systems have been extended to PWA systems [GKBM04]. Unfortunately the technique presented in [GKBM04] introduces a certain level of sub-optimality in the solution.

The main advantages of the infinite time solution, compared to the corresponding finite time solution of the optimal control problem, are the inherent guaranteed stability and feasibility as well as optimality of the closed-loop system [SD87, MRRS00, BGW90, GBTM04].

Here we present novel, computationally efficient algorithms to solve the constrained finite time optimal control problem and the constrained infinite time optimal control (CITOC) problem with a linear performance index for PWA systems. The algorithms combine a dynamic programming exploration strategy with a multi-parametric linear programming solver and basic polyhedral manipulation. In the case of the CITOC problem the developed algorithm guarantees convergence to the optimal solution of the Bellman equation (if a bounded solution exists) and thus avoids potential pitfalls of other conservative approaches. However, the algorithm cannot obtain optimal solutions that have an unbounded cost; but this is hardly a practical limitation since in most applications we want to steer the state to some equilibrium point by spending a finite amount of ‘energy’.

### 7.2 Linear Hybrid Systems

Piecewise affine system are equivalent to many other hybrid system classes [Son81, HDB01], such as for example mixed logical dynamical systems, linear complementary systems, or min-max-plus-scaling systems and thus form a rather general class of linear hybrid systems. Moreover, piecewise affine systems present themselves to be a powerful class for approximating or identifying generic nonlinear systems [Son81, FTMLM03, RBL04].

We consider the class of discrete-time, stabilizable, linear hybrid systems that can be described as constrained continuous\(^1\) piecewise affine (PWA) systems of the following form

\[
x(t+1) = f_{\text{PWA}}(x(t), u(t)) = A_i x(t) + B_i u(t) + a_i, \quad \text{if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in D_i
\]

\(^1\)Here a PWA system defined over a disjoint domain \(D\) is called continuous if \(f_{\text{PWA}}\) is continuous over connected subsets of the domain.
where \( t \geq 0 \), \( x \in \mathbb{R}^n \) is the state, \( u \in \mathbb{R}^m \) is the control input, the domain \( \mathcal{D} \) of \( f_{PW A} \) is a non-empty compact set in \( \mathbb{R}^{m+n} \)

\[
\mathcal{D} := \bigcup_{i=1}^{n_d} \mathcal{D}_i,
\]

with \( n_d < \infty \) the number of system dynamics, and \( \{\mathcal{D}_i\}_{i=1}^{n_d} \) denotes the polyhedral partition of the domain \( \mathcal{D} \), i.e.,

\[
\mathcal{D}_i := \left\{ [x^i u^i] \in \mathbb{R}^{n+m} \mid D^x_i x + D^u_i u \leq D^0_i \right\},
\]

\[
\text{int}(\mathcal{D}_i) \cap \text{int}(\mathcal{D}_j) = \emptyset, \quad \forall i \neq j.
\]

Note that linear state and input constraints of the general form \( C^x x + C^u u \leq C^0 \) can be incorporated in the description of \( \mathcal{D}_i \).

The standing assumption throughout this chapter is:

**Assumption 7.1** (Equilibrium at the origin). The origin in the extended state-input space is an equilibrium point of the PWA system (7.1), i.e.

\[
0_{n+m} \in \mathcal{D} \quad \text{and} \quad 0_n = f_{PW A}(0_n, 0_m)
\]  \hspace{1cm} (7.2)

where \( 0_n := \left[ 0 \ 0 \ldots \ 0 \right]' \in \mathbb{R}^n \).

The above assumption is not limiting the scope of the algorithms. For simplicity, we consider only the cost that penalizes the deviation of the state and control action from the origin (equilibrium point) in the extended state-input space. However, all presented results also hold for any non-zero equilibrium point since such problems are easily translated to the ‘steer-to-the-origin’ problem by a simple linear substitution of the variables.

In the rest of the chapter we will refer to the following definition of feasibility:

**Definition 7.1** (Feasibility). Consider the general optimization problem

\[
\min_{u} \quad \tilde{J}(x, u)
\]

\[
\text{subj. to} \quad [x'u']' \in \tilde{\mathcal{D}}
\]  \hspace{1cm} (7.3)

where \( \tilde{J} \) is a real-valued function defined over the domain \( \tilde{\mathcal{D}} \subseteq \mathbb{R}^{n+m} \), \( x \in \mathbb{R}^n \) denotes a parameter vector (or state) and \( u \in \mathbb{R}^m \) denotes the optimization variable (or input).

We say that the optimization problem (7.3) is feasible if \( \tilde{\mathcal{D}} \neq \emptyset \). The point \( x \in \mathbb{R}^n \) (resp. \( u \in \mathbb{R}^m \)) is called feasible if and only if \( \exists u \in \mathbb{R}^m \) (resp. \( \exists x \in \mathbb{R}^n \)) such that \( [x'u']' \in \tilde{\mathcal{D}} \).

Please note that the terminology of stability of a system, i.e. stability of the origin (as it is considered in this chapter) only makes sense for feasible trajectories. Therefore, trajectories leaving a feasible set can not be considered as ‘unstable’. The terminology of stability (in the classical sense) is not defined outside of a feasible set.
7.3 Constrained Finite Time Optimal Control

We consider the piecewise affine system (7.1) and define the constrained finite time optimal control (CFTOC) problem

$$J^*_T(x(0)) := \min_{U_T} J_T(x(0), U_T)$$

subj. to

$$x(t + 1) = f_{\text{PWA}}(x(t), u(t)),$$

$$x(T) \in \mathcal{X}^f,$$

where

$$J_T(x(0), U_T) := \|P x(T)\|_p + \sum_{t=0}^{T-1} \|Q x(t)\|_p + \|R u(t)\|_p,$$

is the cost function (also called performance index) $U_T$ is the optimization variable defined as input sequence

$$U_T := \{u(t)\}_{t=0}^{T-1},$$

$T < \infty$ is the prediction horizon, $\mathcal{X}^f$ is a compact terminal set in $\mathbb{R}^n$, and $\|\cdot\|_p$ with $p \in \{1, \infty\}$ in (7.6) denotes the corresponding standard vector 1- or $\infty$-norm. The optimal value of the cost function, denoted with $J^*_T$, is called the value function. The optimization variable that achieves $J^*_T$ is called the optimizer and we denote it with $U^*_T := \{u^*(t)\}_{t=0}^{T-1}$.

Remark 7.1 (Infimum problem). Strictly speaking we should formulate the CFTOC problem (7.4)–(7.5) as a search for the infimum rather than for the minimum. However, in our case the cost function $J_T$ comprises a finite number of linear norms. Furthermore, since the set $\mathcal{X}^f$ is compact, $f_{\text{PWA}}$ is continuous over connected subsets of $\mathcal{D}$, $n_d$ is finite, and the sets $\mathcal{D}_i, i = 1, \ldots, n_d$, are compact, we know that the feasible space is compact. Consequently, the Bolzano-Weierstrass existence theorem guarantees that the minimum and infimum problem are equivalent. □

Remark 7.2 (Choice of $P$, $Q$, $R$). Although the problem (7.4)–(7.5) can be posed and solved for any choice of the matrices $P$, $Q$, and $R$, from a practical point of view (if we want to steer the state to the origin while avoiding unnecessary controller action) only the choice of $Q$ and $R$ being of full column rank makes sense. □

Remark 7.3 (Time-varying system and/or cost). The CFTOC problem, (7.4)–(7.5) naturally extends to PWA system and/or cost functions with time-varying parameters, i.e. $A_i(t)$,
B_i(t), a_i(t), D_i(t), as well as Q(t) and R(t) for t = 0, ..., T − 1. For simplicity we focus on the time-invariant case but the CFTOC problem with time-varying parameters is of the same form and complexity as the CFTOC problem with time-invariant parameters and therefore it can be solved in an analog manner.

We summarize the main result concerning the solution to the CFTOC problem (7.4)–(7.5) which is proved in [May01, Bor03].

**Theorem 7.1** (Solution to CFTOC). The solution to the optimal control problem (7.4)–(7.5) with $p \in \{1, \infty\}$ is a piecewise affine value function

$$J_i^*(x(0)) = \Phi_{T,i}x(0) + \Gamma_{T,i}, \quad \text{if} \quad x(0) \in \mathcal{P}_{T,i}$$

and the optimal input $u_i^*(t)$ is a time-varying piecewise affine function of the initial state $x(0)$,

$$u_i^*(t) = K_{T-t,i}x(0) + L_{T-t,i}, \quad \text{if} \quad x(0) \in \mathcal{P}_{T,i}$$

where $t = 0, \ldots, T - 1$, the sets $\mathcal{P}_{T,i}, i = 1, \ldots, N_T$, are polytopic, $\{\mathcal{P}_{T,i}\}_{i=1}^{N_T}$ is a polyhedral partition of the set of feasible states $x(0)$

$$\mathcal{X}_T = \bigcup_{i=1}^{N_T} \mathcal{P}_{T,i},$$

with the closure of $\mathcal{P}_{T,i}$ given by

$$\bar{\mathcal{P}}_{T,i} = \{x \in \mathbb{R}^n \mid P_{T,i}^x x \leq P_{T,i}^0\}.$$

### 7.3.1 The CFTOC Solution via mp-MILP

One way of solving the constrained finite time optimal control problem (7.4)–(7.5) is by reformulating the PWA system (7.1) into a set of inequalities with integer variables $\delta \in \{0, 1\}^n$ as switches between the different ‘dynamics’ $D_i$ of the hybrid system.

By using an upper bound $\varepsilon_i^x$ for each of the components, e.g. $\|Qx(t)\|_p \leq \varepsilon_i^x$, of the cost function (7.6) the CFTOC problem can be rewritten as a mixed-integer linear program (MILP) of the form

$$\min_{\varepsilon} \quad c^T \varepsilon$$

subject to

$$M^\varepsilon \varepsilon \leq M^0 + M^x x(0)$$

where $M^\varepsilon$, $M^0$, and $M^x$ are matrices of suitable dimension, $c = [0 \ldots 0 1 \ldots 1]'$, and the optimization variable is of the form $\varepsilon := [u(0)' \ldots u(T-1)' \delta(0)' \ldots \delta(T-1)' z(0)' \ldots z(T-1)' \varepsilon_0^x \ldots \varepsilon_T^x \varepsilon_0^u \ldots \varepsilon_T^u]'$ where $z(t)$ denotes an auxiliary continuous variable. Note that $x(0)$ can be considered as a parameter of the mp-MILP. The matrices $M^\varepsilon$, $M^0$, and $M^x$ contain the whole information on the state and input constraints, the weighting matrices $P$, $Q$, and $R$, as well as the state update equation for the whole time horizon $T$.

The reader is referred to [BCM03a] for further details on the computation of the CFTOC via an mp-MILP.
Due to Bellman’s optimality principle [Bel57, BD62, Ber00], the constrained finite time optimal control problem (7.4)–(7.5) can be solved in a computationally efficient way by solving an equivalent dynamic program (DP) backwards in time [BCM03a, Ber01, BS96]. The corresponding DP has the following form

$$J^*_k(x(t)) := \min_{u(t)} \|Qx(t)\|_p + \|Ru(t)\|_p$$

$$\begin{align*}
&+ J^*_{k-1}(f_{\text{PWA}}(x(t), u(t))), \\
&\text{subj. to} \quad f_{\text{PWA}}(x(t), u(t)) \in \mathcal{X}_{k-1}
\end{align*}$$

(7.12)

for $k = 1, \ldots, T$, with $t = T - k$, cf. Figure 7.1, and with boundary conditions

$$\mathcal{X}_0 = \mathcal{X}^f, \quad \text{and} \quad J^*_0(x(T)) = \|P x(T)\|_p$$

(7.14)

where

$$\mathcal{X}_k := \{x \in \mathbb{R}^n \mid \exists u \in \mathbb{R}^m, f_{\text{PWA}}(x, u) \in \mathcal{X}_{k-1}\}$$

(7.15)

is the set of all states at time $t = T - k$ for which the problem (7.12)–(7.14) is feasible.

Since $p \in \{1, \infty\}$ the dynamic programming problem (7.12)–(7.14) can be solved by multi-parametric linear programs, cf. [Bor03, BCM03a], where the state $x(t)$ is treated as a parameter and the control input $u(t)$ as an optimization variable. By solving such programs at each iteration step $k$, going backwards in time starting from the target set $\mathcal{X}^f$, we obtain the set $\mathcal{X}_k \subset \mathbb{R}^n$, the optimal control law $u^*(t) : \mathcal{X}_k \to \mathcal{U}_k$, with $t = T - k$, and the value function $J^*_k : \mathcal{X}_k \to \mathbb{R}$ that represents the so called ‘cost-to-go’. Properties of the solution are given in the following theorem, cf. [May01, Bor03].

**Theorem 7.2 (Solution to CFTOC via DP).** The solution to the optimal control problem (7.12)–(7.14) with $p \in \{1, \infty\}$ is a piecewise affine value function

$$J^*_k(x(t)) = \Phi_{k,i}x(t) + \Gamma_{k,i}, \quad \text{if} \quad x(t) \in \mathcal{P}_{k,i}$$

(7.16)
and the optimal input $u^*(t)$ is a time-varying piecewise affine function of the state $x(t)$, i.e. it is given as a state feedback control law

$$u^*(t) = \mu^*_k(x(t)) := F_{k,i}x(t) + G_{k,i}, \quad \text{if} \quad x(t) \in \mathcal{P}_{k,i}$$

(7.17)

where $k = 1, \ldots, T$, $t = T - k$, the sets $\mathcal{P}_{k,i}$, $i = 1, \ldots, N_k$, are polytopic, $\{\mathcal{P}_{k,i}\}_{i=1}^{N_k}$ is a polyhedral partition of the set of feasible states $x(t)$ at time $t$

$$\mathcal{X}_k = \cup_{i=1}^{N_k} \mathcal{P}_{k,i},$$

(7.18)

with the closure of $\mathcal{P}_{k,i}$ given by

$$\bar{\mathcal{P}}_{k,i} = \{x \in \mathbb{R}^n \mid P^*_{k,i}x \leq P^0_{k,i}\}.$$

□

Theorem 7.1 states that the solution to the CFTOC problem (7.4)–(7.5), i.e. the optimal input sequence $U^*_T$ given by (7.9), is a function of the initial state $x(0)$ only. On the other hand, Theorem 7.2 describes the solution to the dynamic program (7.12)–(7.14) as the optimal state feedback control law $\mu^*_k(x(t))$. Since we know that both solutions must be identical (assuming that the optimizer is unique), this implies that there is a connection between the matrices $K_{k,i}$ and $L_{k,i}$ in (7.9) and the matrices $F_{k,i}$ and $G_{k,i}$ in (7.17). It is easy to see that $K_{T,i} = F_{T,i}$ and $L_{T,i} = G_{T,i}$. To establish the connection for the other coefficients one would have to carry out the tedious sequence of substitutions $x(t) = f_{\text{PWA}}(x(t-1), \mu^*_{T-t+1}(x(t-1)))$, which would eventually express $x(t)$ in (7.17) as a function of $x(0)$ only. However, in this chapter we focus on the DP approach in solving the CFTOC problem and since both approaches give the same solution, we will not go beyond this note in establishing an explicit connection between those coefficients. Having this in mind, from this point onwards, when we speak of the solution to the CFTOC problem we consider the solution in the form given in Theorem 7.2.

In the rest of the chapter with $\mu$ we denote a generic state feedback control law that maps a set of states $\mathcal{X}$ to a set of control actions $\mathcal{U}$. Thus $\mu$ specifies the control action (or input action) $u(t) = \mu(x(t))$ that will be chosen at time $t$ when the state is $x(t)$. Furthermore, with $\pi$ we denote a control policy that maps a set of states to a sequence of control actions. For instance, in the case of the CFTOC problem (7.4)–(7.5) with prediction horizon $T$ the optimal control policy is defined by $\pi^*_T := \{\mu^*_T, \ldots, \mu^*_1\}$.

### 7.3.3 An Efficient Algorithm for the CFTOC Solution

In order to present Algorithm 7.1 to solve the CFTOC problem via the dynamic program (7.12)–(7.14), some explanation of the notation and employed functions needs to be given.

When we say SOLVE iteration $k$ of a DP, we mean formulate several multi-parametric linear programs (mp-LPs) for it and obtain a triplet of expressions for the value function, the optimizer, and the polyhedral partition of the feasible state space

$$\mathcal{S}^*_k := \left( J_k^*, \mu^*_k, \{\mathcal{P}_{k,i}\}_{i=1}^{N_k} \right).$$

(7.19)
By inspection of the DP problem (7.12)–(7.14) we see that at each iteration step we are solving \( n_d N_{k-1} \) mp-LPs, where \( n_d \) is the number of system dynamics. After that, by using polyhedral manipulation we have to compare all generated regions, check if they intersect and remove the redundant ones, before storing a new partition that has \( N_k \) regions.

Algorithm 7.1 (Generating the CFTOC solution).

**INPUT** \( f_{\text{PWA}}(x, u), \{D_i\}_{i=1}^{n_d}, p, P, Q, R, T, \mathcal{X}^f \)

**OUTPUT** The CFTOC solution \( S^*_1, \ldots, S^*_T \)

**LET** \( S^*_0 \leftarrow (J^*_0(x) := \|Px\|_p, \mu^*_0(x) := 0_m, \mathcal{X}_0 := \mathcal{X}^f) \)

**FOR** \( k = 1 \) TO \( T \)

**FOR** \( i = 1 \) TO \( n_d \)

**FOR EACH** \( P_{k-1,j} \in \mathcal{X}_{k-1} \)

\[ s_{i,j} \leftarrow \text{SOLVE } \min_u \|Qx\|_p + \|Ru\|_p + J^*_k-1(f_{\text{PWA}}(x, u)), \]

subj. to \( \left\{ [x', u']' \in D_i, \right. \]

\left. f_{\text{PWA}}(x, u) \in P_{k-1,j} \right\} \]

**END**

**END**

**LET** \( S^*_k \leftarrow \text{INTERSECT & COMPARE } \{s_{i,j}\} \)

**END**

In the step where \( \text{INTERSECT & COMPARE} \) is performed, we are removing redundant polyhedra, i.e. we remove such polyhedra that are completely covered with other polyhedra [BT03] which have a ‘better’ (meaning smaller) corresponding value function expression. If some polyhedron \( P_{k,j} \) is only partially covered with ‘better’ regions, the part of \( P_{k,j} \) with the smaller cost can be partitioned into a set of convex polyhedra. Thus we preserve the polyhedral nature of the feasible state space partition in each iteration.

### 7.3.4 Comments on the Dynamic Programming Approach for the CFTOC Problem

In this section some general remarks on what has been observed as being important issues of the new technique will be given.

An important advantage of the dynamic programming approach, compared to the approach based on multi-parametric mixed-integer programming, shortly described in Section 7.3.1, is that after every iteration step, starting from \( k = 1 \) to \( k = T \), the data of all the intermediate optimal control laws, the polyhedral partitions of the state space, and the piecewise affine value functions are available. Thus \( T \) different CFTOC problems with time horizons varying from 1 to \( T \) are simultaneously solved and can be used for analysis and control purposes.

This, in addition, makes it possible to detect if the solution for a specific time horizon is identical to the \( \text{infinite time solution} \) \( (T \to \infty) \), i.e. if for \( T = T_\infty \) the cost as a function
of the initial state $x(0)$ for all feasible states $x(0)$ is identical to the cost for $T \geq T_\infty$. For further detail on the infinite time solution and explanation see Section 7.4.

In some parts of the state space, especially in the regions around the origin, it is likely that in two successive steps of the dynamic programming algorithm identical ‘regions’ – in terms of the regions’ dimensions and the associated value function – are generated. Such a case is depicted in Figure 7.6 ($T = 4$) and Figure 7.7 ($T = 5$) for Example (7.22) where the white encircled regions are identical.

However, only when the piecewise affine value function converges on the whole feasible state space can we claim that the infinite time solution is obtained in any part of the state space. As a consequence it would be wrong to deduce that the infinite time solution was obtained in parts of the state space for some $T < T_\infty$. Such a claim can only be made a posteriori, i.e. after computing the solution to the CFTOC problem with $T \geq T_\infty$.

A modification of the algorithm that aims for the construction of the infinite time solution in a computationally efficient manner by limiting the exploration of the state space in intermediate iteration steps of the dynamic programming algorithm is presented in Section 7.4.

### 7.3.5 Receding Horizon Control

In the case that the receding horizon (RH) control policy [MRRS00] is used in closed-loop the control is given as a time-invariant state feedback control law of the form

$$\mu_{RH}(x(t)) := F_{T,i}x(t) + G_{T,i}, \quad \text{if } x(t) \in \mathcal{P}_{T,i}$$

with $u(t) = \mu_{RH}(x(t))$ and the time-invariant value function is

$$J_{RH}(x(t)) := \Phi_{T,i}x(t) + \Gamma_{T,i}, \quad \text{if } x(t) \in \mathcal{P}_{T,i}$$

for $t \geq 0$. Thus only $N_{RH} := N_T$ (in the worst case different) control laws have to be stored.

Note that in general $J_{RH}$ in (7.21) does not represent the value function of the closed-loop system when the receding horizon control law $\mu_{RH}$ is applied because $J_{RH}(x(0))$ denotes the cost-to-go from $x(0)$ to $x(T)$ when the open-loop input sequence is applied. In the special case when the finite time solution is equivalent to the infinite time solution, i.e. $J_T^* \equiv J_\infty^*$ for some $T < \infty$, $J_{RH}$ in fact does represent the value function of the closed-loop system when applying $\mu_{RH}$, see Remark 7.4.

### 7.3.6 Example: Constrained PWA System

Consider the piecewise affine system [BM99]

$$\begin{cases}
    x(t + 1) = 0.8 \begin{bmatrix}
        \cos \alpha(x(t)) - \sin \alpha(x(t)) \\
        \sin \alpha(x(t)) \\
        \cos \alpha(x(t))
    \end{bmatrix} x(t) + \begin{bmatrix} 0 \end{bmatrix} u(t), \\
    \alpha(x(t)) = \begin{cases}
        2\pi & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \geq 0, \\
        -\frac{2\pi}{3} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) < 0,
    \end{cases} \\
    x(t) \in [-10, 10] \times [-10, 10], \\
    u(t) \in [-1, 1].
\end{cases}$$

(7.22)
The CFTOC problem (7.4)–(7.5) is solved with $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 1$, $P = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, and $\mathcal{X}' = [-10, 10] \times [-10, 10]$ for $p = \infty$.

Figure 7.2 shows the state space partition of the finite time horizon solution for $T = 8$ computed with the dynamic programming algorithm proposed in Section 7.3.3. The same color corresponds to the same affine control law $\mu_8^*(x(0))$. There exist 19 different affine control laws in 262 polyhedral regions. Each polyhedral region corresponds to a different affine value function.

Figure 7.3 depicts the state space partition for the infinite time horizon solution computed with the dynamic programming algorithm of Section 7.3.3. A posteriori it can be shown with the dynamic programming procedure that the finite time solution for a horizon $T \geq 11 = T_\infty$ is in fact identical to the infinite time solution of the constrained optimal control problem, cf. Section 7.3.4. The infinite time solution for this example was solved in 1515 seconds on a Pentium 4, 2.2 GHz machine running MATLAB® 6.1.

The same coloring scheme corresponds to the same affine control law. There exist 23 different affine control laws $\mu^*_\infty(x(0))$ in 252 polyhedral regions. Figure 7.4 reveals the corresponding value function for the state space partition. The same color corresponds to the same cost. The minimum cost is naturally achieved at the origin. Figure 7.5 shows the state and control action evolution with an initial state of $x(0) = [-10\ 10]'$ for the infinite time solution obtained with the dynamic programming procedure.
Figure 7.3: State space partition of the infinite time solution \((T = T_\infty = 11)\) for Example (7.22) derived with the dynamic programming algorithm. Same color corresponds to the same affine control law \(\mu^*_{T_\infty}(x(0))\).

Figure 7.4: State space partition of the infinite time solution \(J^*_{T_\infty}(x)\) with \(T = T_\infty = 11\) for Example (7.22) derived with the dynamic programming algorithm. Same color corresponds to the same cost value.
7 CFTOC and CITOC of PWA Systems with a Linear Performance Index

Figure 7.5: State and control action evolution of the infinite time solution derived with the dynamic programming algorithm for Example (7.22). Initial state $x(0) = [-10\ 10]'$.

Figure 7.6: State space partition of the finite time solution $J^*_T(x)$ for $T = 4$ for Example (7.22) derived with the dynamic programming algorithm. Same color corresponds to the same cost value. The white marked region is identical to the infinite time solution.

7.4 Constrained Infinite Time Optimal Control

As in the previous section we consider the piecewise affine system (7.1) subject to state and input constraints and by letting $T \to \infty$ the cost function (7.6) takes the following form
Figure 7.7: State space partition of the finite time solution $J^*_T(x)$ for $T = 5$ for Example (7.22) derived with the dynamic programming algorithm. Same color corresponds to the same cost value. The white marked region is identical to the infinite time solution.

(assuming that the limit exists)

$$J_\infty(x(0), U_\infty) := \lim_{T \to \infty} \sum_{t=0}^{T} g(x(t), u(t))$$

(7.23)

where the function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ also called the stage-cost is defined by

$$g(x, u) := \|Qx\|_p + \|Ru\|_p,$$

(7.24)

with $p \in \{1, \infty\}$. Moreover, (7.4)–(7.5) becomes the constrained infinite time optimal control (CITOC) problem

$$J^*_\infty(x(0)) := \min_{U_\infty} J_\infty(x(0), U_\infty),$$

(7.25)

subj. to $x(t + 1) = f_{\text{PWA}}(x(t), u(t))$

(7.26)

where by $U_\infty := \{u(t)\}_{t=0}^\infty$ we denote the optimization input sequence and by $U^*_\infty := \{u^*(t)\}_{t=0}^\infty$ the optimizer of (7.25)–(7.26).

We denote with

$$\mathcal{X}_\infty := \{x \in \mathbb{R}^n \mid \exists U^*_\infty, J^*_\infty(x) < \infty\}.$$  

(7.27)

the set of states $x(0)$ for which the CITOC problem (7.25)–(7.26) is well defined, i.e. the minimum is achieved for some feasible input sequence $U^*_\infty$, and $J^*_\infty(x(0)) < \infty$.

In order to guarantee closed-loop stability we assume that $Q$ is of full column rank as it will be shown in the following, cf. Lemma 7.1. Additionally, also in the infinite time case it
can be assumed that $R$ is of full column rank even though these assumptions are not strictly needed, cf. Remark 7.2.

**Assumption 7.2** (Boundedness of $J^*_{\infty}$). The CITOC problem (7.25)–(7.26) is well defined, i.e. the minimum is achieved for some feasible input sequence $U^*_\infty$, and $J^*_{\infty}(x(0)) < \infty$ for any feasible state $x(0)$ on a closed set $X^\infty$.

This assumption is hardly a limitation to the applicability of the presented method, since in most practical applications we want to steer the state from some given state $x(0)$ or set to some equilibrium point (here the origin, cf. Assumption 7.1 and the paragraph below) by spending a finite amount of ‘energy’. This is illustrated in the following example.

**Example 7.1** (Constrained LTI system). Consider the simple CITOC problem

$$J^*_{\infty}(x(0)) = \min_{U^*_{\infty}} \lim_{T \to \infty} \sum_{t=0}^{T} |x(t)|, \tag{7.28}$$

subj. to

$$\begin{cases}
  x(t+1) = 2x(t) + u(t), \\
  |x(t)| \leq 1, \text{ and } |u(t)| \leq 1
\end{cases} \tag{7.29}$$

for the constrained one-dimensional LTI system (7.29).

Problem (7.28)–(7.29) is feasible for all initial states in $\bar{X}^\infty = [-1, 1]$ and one can observe that the closed-loop system for the optimal state feedback control law

$$\mu^*_\infty(x) = \begin{cases}
  1 & \text{if } x \in [-1, -0.5], \\
  -2x & \text{if } x \in [-0.5, 0.5], \\
  -1 & \text{if } x \in [0.5, 1]
\end{cases} \tag{7.30}$$

inherits three equilibria at $-1$, $0$, and $1$. However, the closed-loop system is only asymptotically stable for the open set $X^\infty = (-1, 1)$. Figure 7.8 illustrates that the optimal value function $J^*_{\infty}(x(0)) \to \infty$ as $x(0) \to \pm 1$ and therefore the problem is not well defined in the sense of Assumption 7.2. In practice, one can compute a $\delta$-close approximation of $J^*_{\infty}$ on a closed subset $X^\delta \subset X^\infty$ as it was performed for obtaining Figure 7.8. Note that choosing any $R \neq 0$ does only influence the ‘shape’ of $J^*_{\infty}$ and $\mu^*_\infty$ but does not influence the above mentioned characteristic behavior of the solution.

Please note that most of the following results hold also (or are straight forwardly extended) for general continuous nonlinear systems and are not restricted to the considered class of PWA systems.

**Lemma 7.1** (Stability of the CITOC solution). Consider the CITOC problem (7.25)–(7.26). Then the following hold:

(a) The origin $[x' u']' = 0_{n+m}$ is a part of the infinite time solution, i.e. $0_n \in X^\infty$ with $J^*_{\infty}(0_n) = 0$ and $u^*(t) = 0_m$.

(b) Let Assumption 7.2 hold. Then by applying the optimizer $U^*_\infty$ to the system, any system state $x \in X^\infty$ is driven to the origin, i.e. if $x(0) \in X^\infty$ then $x(t) \in X^\infty$ for all $t \geq 0$ and $\lim_{t \to -\infty} x(t) = 0_n$.  \qed
7 CFTOC and CITOC of PWA Systems with a Linear Performance Index

Figure 7.8: $\delta$-close approximation of the optimal value function $J^*_\infty(x)$ for Example 7.1. The colored $x$-axis denotes the different regions over which the piecewise affine value function $J^*_\infty(x)$ is defined.

Proof  (a) Because $[x'u']' = 0_{n+m}$ is an equilibrium point of the system (7.1) and $g(x,u) \geq 0$ for all $[x'u']' \in \mathcal{D}$, the minimum of $J_\infty(x(0),U_\infty) \geq 0$ for $x(0) = 0_n$ is achieved with e.g. $u^*(t) = 0_m$ for all $t \geq 0$. That means $J^*_\infty(0_n) = 0$ which is the smallest possible value of $J^*_\infty(\cdot,\cdot)$.

(b) By Assumption 7.2 and $g(\cdot,\cdot) \geq 0$ we have $0 \leq J^*_\infty(\cdot,\cdot) < \infty$, i.e. $J^*_\infty(\cdot,\cdot)$ is bounded from above and below. Additionally, the sequence $J_T := \sum_{t=0}^{T} g(x(t),u(t))$ for any sequence $\{(x(t),u(t))\}_{t=0}^{T}$, as $T$ increases, is non-decreasing. Since we are using $U^*_\infty$, $J_T$ converges to $J^*_\infty$. Consequently, for every $\varepsilon > 0$ there exists a $T_0 < \infty$ with $|J_T - J^*_\infty| < \varepsilon$ for all $T \geq T_0$. Therefore necessarily $\lim_{T \to \infty} g(x(T),u(T)) = 0$ and because $Q$ is of full column rank it follows $\lim_{T \to \infty} x(T) = 0_n$.  

7.4.1 The CITOC Solution via Dynamic Programming

Similar to the recasting of the CFTOC problem into a recursive dynamic program as presented in Section 7.3.2, it is possible to formulate for the CITOC problem (7.25)–(7.26) the corresponding dynamic program (DP) as follows

$$J_k(x(t)) := \min_{u(t)} g(x(t),u(t)) + J_{k-1}(f_{\text{PWA}}(x(t),u(t))), \quad (7.31)$$

$$\text{subj. to } f_{\text{PWA}}(x(t),u(t)) \in \mathcal{X}_{k-1} \quad (7.32)$$
for $k = 1, \ldots, \infty$, with initial conditions
\begin{align}
X_0 &= \{ x \in \mathbb{R}^n \mid \exists u, \ [x'u'] \in D \}, \quad \text{and} \\
J_0(x) &= 0 \quad \forall x \in X_0. \quad (7.33) \\
J_0(x) &= 0 \quad \forall x \in X_0. \quad (7.34)
\end{align}

The set of all initial states for which the problem (7.31)–(7.34) is feasible at iteration step $k$ is given by
\begin{align}
X_k := \{ x \in \mathbb{R}^n \mid \exists u, \ f_{\text{PWA}}(x, u) \in X_{k-1} \} \\
&= \bigcup_{i=1}^{N_k} P_{k,i}.
\end{align}

Furthermore, we define the feasible set of states as $k \to \infty$ by
\begin{equation}
X_\infty := \lim_{k \to \infty} X_k,
\end{equation}
and the limit value function of the dynamic program (7.31)–(7.34) by
\begin{equation}
J_{\infty,\text{DP}}^* := \lim_{k \to \infty} J_k.
\end{equation}

From $0_n \in X_0$, $J_0(0_n) = 0$, and Assumption 7.1 it is easy to see that the following properties
\begin{equation}
0_n \in X_k \quad \text{and} \quad J_k(0_n) = 0 \quad \text{for all } k \geq 0
\end{equation}
of the dynamic program (7.31)–(7.34) hold.

**Lemma 7.2** (Stability of the DP solution). Consider the DP problem (7.31)–(7.34). Then, when applying the corresponding optimal control law $u^*(t) = \mu_{\infty,\text{DP}}^*(x(t))$ for all $x(t) \in X_\infty$ to the system, the following holds:

(a) the limit value function $J_{\infty,\text{DP}}^*(x) < \infty$ for all $x \in X_\infty$, with $J_{\infty,\text{DP}}^*(0_n) = 0$, is a global Lyapunov function for the closed-loop system, and

(b) any system state $x \in X_\infty$ is driven to the origin, i.e. if $x(0) \in X_\infty$ then $x(t) \in X_\infty$ for all $t \geq 0$ and $\lim_{t \to \infty} x(t) = 0_n$. □

**Proof** (a) Because $J_{\infty,\text{DP}}^*$ is a limit function of the DP (7.31)–(7.34) we have $J_{\infty,\text{DP}}^*(x) = g(x, \mu_{\infty,\text{DP}}^*(x)) + J_{\infty,\text{DP}}^*(f_{\text{PWA}}(x, \mu_{\infty,\text{DP}}^*(x)))$ for all $x \in X_\infty$. Thus, with $J_{\infty,\text{DP}}^*(\cdot) \geq 0$, it follows that $-\Delta J_{\infty,\text{DP}}^*(x) := J_{\infty,\text{DP}}^*(x) - J_{\infty,\text{DP}}^*(f_{\text{PWA}}(x, \mu_{\infty,\text{DP}}^*(x))) = g(x, \mu_{\infty,\text{DP}}^*(x)) \geq \|Qx\|_p$ for all $x \in X_\infty$. Because $Q$ is of full column rank, there always exists a finite $\alpha_1 > 0$ with $\|Qx\|_p \geq \alpha_1 \|x\|_2$ and thus $-\Delta J_{\infty,\text{DP}}^*(x) \geq \alpha_1 \|x\|_2$ for all $x \in X_\infty$. This means that $-\Delta J_{\infty,\text{DP}}^*(\cdot)$ is always bounded below by some $K$-class function. Similarly, we have that $J_{\infty,\text{DP}}^*(x) \geq g(x, \mu_{\infty,\text{DP}}^*(x)) \geq \|Qx\|_p \geq \alpha_2 \|x\|_2$ for all $x \in X_\infty$ and some finite $\alpha_2 > 0$. Moreover, $J_{\infty,\text{DP}}^*(\cdot)$ is bounded over a bounded set and from property (7.38) we have that $J_{\infty,\text{DP}}^*(0) = 0$. From these statements it follows directly that $J_{\infty,\text{DP}}^*$ is a global Lyapunov function [Vid93] for the closed-loop system.

(b) Due to the existence of a global Lyapunov function (Lemma 7.2(a)) for the closed-loop system on the set $X_\infty$ it follows immediately that the closed-loop system is globally asymptotic stable [Vid93], i.e. if $x(0) \in X_\infty$ then $x(t) \in X_\infty$ for all $t \geq 0$ and $\lim_{t \to \infty} x(t) = 0_n$. ■
Note that in the infinite time case, in contrast to the finite time case discussed in Section 7.3, the equivalence of the solution of the dynamic program $J^*_{\infty,DP}$ and the optimal solution $J^*_\infty$ of the CITOC problem is not immediate. Before we prove this equivalence it is useful to introduce the following operators.

**Definition 7.2 (Operator $T$ and $T_\mu$).** For any function $J : \mathcal{X} \to \mathbb{R}_{\geq 0}$ we define the following mapping
\[
(TJ)(x) := \min_{u \in \mathcal{U}} g(x, u) + J(f_{\text{PWA}}(x, u)) \quad (7.39)
\]
where the set of feasible control actions $\mathcal{U} \subset \mathbb{R}^m$ is defined implicitly through the domains of $J$ and $f_{\text{PWA}}$
\[
\mathcal{U} := \{ u \in \mathbb{R}^m \mid \exists x, [x'u']' \in \mathcal{D}, f_{\text{PWA}}(x, u) \in \mathcal{X} \}.
\]
$T$ transforms the function $J$ on $\mathcal{X}$ into the function $TJ : \tilde{\mathcal{X}} \to \mathbb{R}_{\geq 0}$.

$T_k$ denotes the $k$-times operator of $T$ with itself, i.e. $(T^kJ)(\cdot) = (T(T^{k-1}J))(\cdot)$ and $(T^0J)(\cdot) = J(\cdot)$ with $k \in \mathbb{N}_{\geq 0}$. Accordingly, we use
\[
(T_\mu J)(x) := g(x, \mu(x)) + J(f_{\text{PWA}}(x, \mu(x))) \quad (7.40)
\]
for any function $J : \mathcal{X} \to \mathbb{R}_{\geq 0}$ and any control function $\mu : \mathcal{X} \to \mathcal{U}$ defined on the state space $\mathcal{X}$.

The DP procedure (7.31)–(7.34) can now be simply stated as $J_k := T J_{k-1}$, with $J_0(\cdot) = 0$, $J^*_{\infty,DP} = \lim_{k \to \infty} T^k J_0$. The solution to the DP procedure, $J^*_{\infty,DP}$, satisfies the Bellman equation
\[
J = TJ \quad (7.41)
\]
which is effectively being used as a stopping criteria to decide when the DP procedure has terminated.

To prove that the solution to the CITOC problem (7.25)–(7.26), $J^*_\infty$, is identical to the solution of the dynamic program (7.31)–(7.34), $J^*_{\infty,DP}$, we actually have to answer two questions: first, under which conditions does the DP procedure (7.31)–(7.34) converge, and second, when is $J^*_{\infty,DP}$ a unique solution to the dynamic program (7.31)–(7.34).

**Theorem 7.3 (Equivalence: CITOC solution–DP solution).** Let Assumption 7.2 hold, then the solution to the CITOC problem (7.25)–(7.26), $J^*_\infty$, is identical to the solution of the dynamic program (7.31)–(7.34), $J^*_{\infty,DP}$, over the same feasible set $\mathcal{X}_\infty$. In other words $J^*_\infty(x) = J^*_\infty,DP(x)$ for all $x \in \mathcal{X}_\infty$. Moreover, the solution $J^*_\infty,DP$ is a unique solution to the dynamic program (7.31)–(7.34).

**Proof** According to [Ber01, Sec. 3], the solution $J^*_\infty$ to the CITOC problem (7.25)–(7.26) satisfies the dynamic program (7.31)–(7.34), that is $J^*_\infty = TJ^*_\infty$. On the other hand, in general, the Bellman equation (7.41) may have no solution or it may have multiple solutions, but at most one solution has the property
\[
\lim_{t \to \infty} J^*_{\infty,DP}(x(t)) = 0, \quad (7.42)
\]
cf. [SL89, Sec. 4.]. Now, if the DP (7.31)–(7.34) has a solution $J^*_{\infty, \text{DP}}(\cdot) < \infty$ fulfilling
this property then it is unique and according to [SL89, Thm. 4.3] this solution satisfies the
CITOC problem. From Lemma 7.2 it follows immediately that the DP solution does in
fact satisfy property (7.42). This completes the proof of equivalence and uniqueness of the
respective solutions. ■

Having established this result, in the following we will denote the solution to both prob-
lems, the CITOC problem (7.25)–(7.26) and the DP problem (7.31)–(7.34), with $J^*_{\infty}$.

Lemma 7.3 (Optimal control law $\mu^*_{\infty}$, [Ber01, Prop. 3.1.3]). A stationary control law $\mu^*_{\infty}$
is optimal if and only if $(TJ^*_\infty)(x) = (T\mu^*_{\infty}J^*_\infty)(x)$ for all $x \in \mathcal{X}_\infty$. In other words, $\mu^*_{\infty}(x)$
is optimal if and only if the minimum of (7.31) is obtained with $\mu^*_{\infty}(x)$ for all $x \in \mathcal{X}_\infty$. □

Now we are ready to state the theorem that characterizes the optimal solution $J^*_{\infty}$ and
the optimal state feedback control law $\mu^*_{\infty}$.

Theorem 7.4 (Solution to the CITOC and DP problem). Under Assumption 7.2 the solution
to the optimal control problem (7.25)–(7.26) with $p \in \{1, \infty\}$ is a piecewise affine value
function

\[
J^*_{\infty}(x(t)) = \Phi_{\infty,i}x(t) + \Gamma_{\infty,i}, \quad \text{if} \quad x(t) \in \mathcal{P}_{\infty,i}
\]  

(7.43)

and the optimal state feedback control law is of the time-invariant piecewise affine form

\[
\mu^*_{\infty}(x(t)) = F_{\infty,i}x(t) + G_{\infty,i}, \quad \text{if} \quad x(t) \in \mathcal{P}_{\infty,i}
\]  

(7.44)

where $\{\mathcal{P}_{\infty,i}\}_{i=1}^N$ is a polyhedral partition of the set $\mathcal{X}_\infty$ of feasible states $x(t)$ at time $t$ with $t \geq 0$. □

Proof Theorem 7.4 follows directly from the construction of the DP iterations (7.31)–(7.34),
Theorem 7.2, and Assumption 7.2. ■

Remark 7.4 (Receding Horizon Control). Note that the infinite time optimal solution (7.44)
has the same form as the corresponding receding horizon control policy (7.20). This means
that with the optimal control law $\mu^*_{\infty}(x(t))$ the closed-loop and open-loop response of the
system (7.1) are identical. Moreover, the value $J^*_{\infty}(x(t))$ is the total ‘cost-to-go’ from $x(t)$
to the origin, applying the optimal control policy. □

In the dynamic program (7.31)–(7.34) we start the value function iteration procedure
with the zero-function (7.34) as initial condition. Due to the monotonicity property of the
operator $T$, cf. [Ber01, Lem. 1.1.1], this guarantees the convergence to the optimal solution
$J^*_{\infty}$ of ‘arbitrary’ dynamic programming problems if $J_\infty(x) := \lim_{k \to \infty}(T^k J_0)(x) < \infty$ is
a stationary solution, i.e. $J_\infty(x) = (TJ_\infty)(x)$ for all feasible $x$, cf. [Ber01, Prop. 3.1.5].
In addition, the following results show that we can guarantee convergence to the optimal
solution $J^*_{\infty}$ from (almost) arbitrary $J_0$. 

Having established this result, in the following we will denote the solution to both prob-
lems, the CITOC problem (7.25)–(7.26) and the DP problem (7.31)–(7.34), with $J^*_{\infty}$. 

Lemma 7.3 (Optimal control law $\mu^*_{\infty}$, [Ber01, Prop. 3.1.3]). A stationary control law $\mu^*_{\infty}$
is optimal if and only if $(TJ^*_\infty)(x) = (T\mu^*_{\infty}J^*_\infty)(x)$ for all $x \in \mathcal{X}_\infty$. In other words, $\mu^*_{\infty}(x)$
is optimal if and only if the minimum of (7.31) is obtained with $\mu^*_{\infty}(x)$ for all $x \in \mathcal{X}_\infty$. □

Now we are ready to state the theorem that characterizes the optimal solution $J^*_{\infty}$ and
the optimal state feedback control law $\mu^*_{\infty}$.

Theorem 7.4 (Solution to the CITOC and DP problem). Under Assumption 7.2 the solution
to the optimal control problem (7.25)–(7.26) with $p \in \{1, \infty\}$ is a piecewise affine value
function

\[
J^*_{\infty}(x(t)) = \Phi_{\infty,i}x(t) + \Gamma_{\infty,i}, \quad \text{if} \quad x(t) \in \mathcal{P}_{\infty,i}
\]  

(7.43)

and the optimal state feedback control law is of the time-invariant piecewise affine form

\[
\mu^*_{\infty}(x(t)) = F_{\infty,i}x(t) + G_{\infty,i}, \quad \text{if} \quad x(t) \in \mathcal{P}_{\infty,i}
\]  

(7.44)

where $\{\mathcal{P}_{\infty,i}\}_{i=1}^N$ is a polyhedral partition of the set $\mathcal{X}_\infty$ of feasible states $x(t)$ at time $t$ with $t \geq 0$. □

Proof Theorem 7.4 follows directly from the construction of the DP iterations (7.31)–(7.34),
Theorem 7.2, and Assumption 7.2. ■

Remark 7.4 (Receding Horizon Control). Note that the infinite time optimal solution (7.44)
has the same form as the corresponding receding horizon control policy (7.20). This means
that with the optimal control law $\mu^*_{\infty}(x(t))$ the closed-loop and open-loop response of the
system (7.1) are identical. Moreover, the value $J^*_{\infty}(x(t))$ is the total ‘cost-to-go’ from $x(t)$
to the origin, applying the optimal control policy. □
Lemma 7.4 (Lower bound \( \mathcal{J} \)). Let the function \( \mathcal{J} : \mathcal{X}_\infty \to \mathbb{R}_{\geq 0} \) with
\[
\mathcal{J}(0_n) = 0 \quad \text{and} \quad \mathcal{J}(x) \leq g(x, u) + \mathcal{J}(f_{\text{PWA}}(x, u))
\]
for all \((x, u) \in \mathcal{X}_\infty \times \mathcal{U} \) then \( \mathcal{J}(x) \leq J_\infty^*(x) \) \( \forall \ x \in \mathcal{X}_\infty. \)

Proof By Lemma 7.3 the control input sequence \( \{\mu_\infty^*(x(t))\}_{t=0}^\infty \) obtains the optimal value function \( J_\infty^*(x(0)) \) for all \( x(0) \in \mathcal{X}_\infty. \) Using a \( \mathcal{J} \) with the properties given in Equation (7.45) it follows for the optimal sequence \( \{(x(t), \mu_\infty^*(x(t)))\}_{t=0}^\infty \) that \( g(x(t), \mu_\infty^*(x(t))) \geq \mathcal{J}(x(t)) - \mathcal{J}(x(t + 1)) \) for all \( t \geq 0. \) Moreover, under Assumption 7.2, from Lemma 7.1 we have \( J_\infty^*(0_n) = 0 \) and \( \lim_{t \to \infty} x(t) = 0 \), i.e. \( J_\infty^*(x(0)) < \infty \) for all \( x(0) \in \mathcal{X}_\infty. \) Therefore it follows that \( \infty > J_\infty^*(x(0)) = \lim_{T \to \infty} \sum_{t=0}^T g(x(t), \mu_\infty^*(x(t))) \geq \lim_{T \to \infty} \sum_{t=0}^T \mathcal{J}(x(t)) - \mathcal{J}(x(t + 1)) = \lim_{T \to \infty} \mathcal{J}(x(0)) - \mathcal{J}(x(T)) = \mathcal{J}(x(0)) - \mathcal{J}(0_n) = \mathcal{J}(x(0)) \) for all \( x(0) \in \mathcal{X}_\infty. \)

Theorem 7.5 (Initial value function \( J_0 \)). Let Assumption 7.2 hold and let \( J_0 : \mathcal{X}_\infty \to \mathbb{R}_{\geq 0} \), with \( J_0(0_n) = 0 \), \( J_0(\bullet) < \infty \), and \( 0_n \in \mathcal{X}_\infty \) be the initial value function of the dynamic program iteration (7.31)-(7.32). Moreover, let
\[
J_\mu(x) := \lim_{k \to \infty} (T^k J_0)(x)
\]
be a finitely bounded limit function, i.e. \( J_\mu(\bullet) < \infty \), with \( J_\mu(0_n) = 0 \), and
\[
\mu(x) := \arg\min_{u \in \mathcal{U}} g(x, u) + J_\mu(f_{\text{PWA}}(x, u))
\]
its corresponding stationary control law for all \( x \in \mathcal{X}_\infty. \)

(a) If \( J_0 \), in addition, is some (arbitrary) finitely bounded function on \( \mathcal{X}_\infty \) with \( 0 \leq J_0(x) \leq J_\infty^*(x) \) for all \( x \in \mathcal{X}_\infty \) then \( J_\mu(x) = J_\infty^*(x) \) and \( \mu(x) \) is an optimal control law for all \( x \in \mathcal{X}_\infty. \)

(b) If \( J_0 \), in addition, is some (arbitrary) finitely bounded function on \( \mathcal{X}_\infty \), and if after some (possibly infinite) DP iterations \( J_\mu \) is a stationary solution, i.e. \( J_\mu(x) = (T J_\mu)(x) \) for all \( x \in \mathcal{X}_\infty \), then \( J_\mu(x) = J_\infty^*(x) \) and \( \mu(x) \) is an optimal control law for all \( x \in \mathcal{X}_\infty. \)

(c) If \( J_0 \), in addition, is some (arbitrary) finitely bounded function on \( \mathcal{X}_\infty \) and if \( J_0(x) \geq (T J_0)(x) \) for all \( x \in \mathcal{X}_\infty \), then the existence of \( J_\mu \) is guaranteed, \( J_\mu(x) = J_\infty^*(x) \), and \( \mu(x) \) is an optimal control law for all \( x \in \mathcal{X}_\infty. \)

(d) If \( J_0 \), in addition, is a finitely bounded realization of some feasible input sequence \( \{\overline{g}(x(t))\}_{t=0}^\infty \) with \( \overline{g}(x(t)) \in \mathcal{U} \), i.e. \( J_0(x(0)) := \lim_{T \to \infty} \sum_{t=0}^T g(x(t), \overline{g}(x(t))) \) for all \( x(0) \in \mathcal{X}_\infty \), then the existence of \( J_\mu \) is guaranteed, \( J_\mu(x) = J_\infty^*(x) \), and \( \mu(x) \) is an optimal control law for all \( x \in \mathcal{X}_\infty. \)

Proof
(a) See [Ber01, Prop. 1.1.5].
(b) We have that \( J_\mu \) is a stationary solution of the dynamic program (7.31)–(7.32), i.e. \( J_\mu(x) = (TJ_\mu)(x) \) for all \( x \in X_\infty \) with \( J_\mu(0_n) = 0 \). However, due to optimality of \( J_\mu^* \) we have that \( J_\mu(x) \geq J_\mu^*(x) \) for all \( x \in X_\infty \); at the same time it follows from \( J_\mu(x) = (TJ_\mu)(x) \) that \( J_\mu(x) \leq g(x, u) + J_\mu(f_{PWA}(x, u)) \) for all \( (x, u) \in X_\infty \times U \). Thus from Lemma 7.4 follows \( J_\mu(x) \leq J_\mu^*(x) \) for all \( x \in X_\infty \). Therefore, \( J_\mu = J_\mu^* \).

(c) Due to monotonicity of the operator \( T \) [Ber01, Lem. 1.1.1] we have that \( \langle T^k J_0 \rangle(x) \geq \langle T^{k+1} J_0 \rangle(x) \) for all \( x \in X_\infty \) and \( k \in \mathbb{N}_{\geq 0} \). Additionally, the sequence \( \langle T^k J_0 \rangle(x(0)) \) with increasing \( k \) is bounded from below by \( J_\mu^*(x(0)) \) for all \( x(0) \in X_\infty \), thus the sequence converges to some stationary, bounded solution \( J_\mu \), i.e. \( J_\mu(x) = (TJ_\mu)(x) \) for all \( x \in X_\infty \). In addition, from Part (b) of this theorem we then have \( J_\mu = J_\mu^* \).

(d) For an arbitrary feasible input sequence \( \{\bar{\mu}(x(t))\}_{t=0}^\infty \), due to sub-optimality, we clearly have \( \infty > J_0(x(0)) \geq J_\mu^*(x(0)) \geq 0 \) for all \( x(0) \in X_\infty \). Because \( J_0(x(0)) \) for all \( x(0) \in X_\infty \) is a realizable cost it follows that

\[
J_0(x(0)) = \lim_{T \to \infty} \sum_{t=0}^{\infty} g(x(t), \bar{\mu}(x(t)))
\]

\[
= g(x(0), \bar{\mu}(x(0))) + \lim_{T \to \infty} \sum_{t=1}^{T} g(x(t), \bar{\mu}(x(t)))
\]

\[
= g(x(0), \bar{\mu}(x(0))) + J_0(f_{PWA}(x(0), \bar{\mu}(x(0))))
\]

\[
\geq \min_{u \in U} g(x(0), u) + J_0(f_{PWA}(x(0), u))
\]

\[
= (TJ_0)(x(0))
\]

for all \( x(0) \in X_\infty \). The rest follows directly from Part (c) of this theorem.

(a)–(d) From \( J_\mu = J_\mu^* \) and the stationarity of \( J_\mu^* \) follows directly that \( \langle T^k J_\mu^* \rangle = \langle T^k J_\mu^* \rangle \) and thus with Lemma 7.3 we have that \( \mu(x) \) is an optimal control law for all \( x \in X_\infty \).

\[\blacksquare\]

**Corollary 7.1** (Control Lyapunov function). *If in addition to the assumptions of Theorem 7.5, \( J_0 \) is chosen to be a (global) control Lyapunov function on \( X_\infty \) then the existence of \( J_\mu \) is guaranteed, \( J_\mu(x) = J_\mu^*(x) \), and \( \mu(x) = \mu^*(x) \) for all \( x \in X_\infty \).*

**Proof** Corollary 7.1 follows directly from Theorem 7.5(c) and the definition of a control Lyapunov function.

This is a rather strong and computationally important result guaranteeing that we will always find the unique optimal finitely bounded solution if one exists.

**Theorem 7.6** (Convergence rate). *Let \( J_k, X_k \) be solutions to the DP procedure (7.31)–(7.34) such that \( X_k = X, \forall k \geq \bar{k} \), for some \( X \subseteq \mathbb{R}^n \) and \( \bar{k} \in \mathbb{N} \). Then the convergence rate

\[
\gamma(k) := \max_{x \in X} |J_{k+1}(x) - J_k(x)|, \quad k \geq \bar{k}.
\]

(7.48)

is a monotonically non-increasing function, i.e. \( \gamma(k+1) \leq \gamma(k) \), for all \( k \geq \bar{k} \).
\textbf{Proof} Let } k \geq \bar{k} + 1. \text{ From the definition of } J_k(x) \text{ we see that for all } x \in \mathcal{X} \text{ the following holds}

\begin{align*}
J_k(x) &:= J_{k-1}(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
&\leq J_{k-1}(f_{\text{PWA}}(x, u)) + g(x, u), \\
&\forall (x, u) : f_{\text{PWA}}(x, u) \in \mathcal{X}
\end{align*}

(7.49)

(7.50)

where } \mu^*_k(x) \text{ is the optimal control for a given state } x \in \mathcal{X}, \text{ i.e., it is a function of } x. \text{ Hence it follows}

\begin{align*}
\max_{x \in \mathcal{X}} &\quad J_k(x) - J_{k+1}(x) \\
= &\max_{x \in \mathcal{X}} J_{k-1}(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
&\quad - J_k(f_{\text{PWA}}(x, \mu^*_{k+1}(x))) + g(x, \mu^*_{k+1}(x)) \\
\leq &\max_{x \in \mathcal{X}} J_{k-1}(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
&\quad - J_k(f_{\text{PWA}}(x, \mu^*_{k+1}(x))) + g(x, \mu^*_{k+1}(x)) \\
= &\max_{x \in \mathcal{X}} J_{k-1}(y) - J_k(y) \\
\text{subj.to } &\quad y = f_{\text{PWA}}(x, \mu^*_{k+1}(x)) \\
\leq &\max_{y \in \mathcal{X}} J_{k-1}(y) - J_k(y)
\end{align*}

where the first inequality is due to (7.50), and the second one due to the removal of some of the constraints from the problem \((x \in \mathcal{X} \text{ and } y = f_{\text{PWA}}(x, \mu^*_{k+1}(x)))\). In a similar way we get

\begin{align*}
\max_{x \in \mathcal{X}} &\quad J_{k+1}(x) - J_k(x) \\
= &\max_{x \in \mathcal{X}} J_k(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
&\quad - J_{k-1}(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
\leq &\max_{x \in \mathcal{X}} J_k(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
&\quad - J_{k-1}(f_{\text{PWA}}(x, \mu^*_k(x))) + g(x, \mu^*_k(x)) \\
= &\max_{x, y \in \mathcal{X}} J_k(y) - J_{k-1}(y) \\
\text{subj.to } &\quad y = f_{\text{PWA}}(x, \mu^*_k(x)) \\
\leq &\max_{y \in \mathcal{X}} J_k(y) - J_{k-1}(y)
\end{align*}

Finally, by taking into account that for any function } h(x) \text{ we have

\[
\max_x |h(x)| = \max \{ \max_x h(x), \max_x -h(x) \},
\]
then the rest of the proof follows easily:

\[ \gamma(k) := \max_{x \in X} |J_{k+1}(x) - J_k(x)| \]

\[ = \max \{ \max_{x \in X} J_{k+1}(x) - J_k(x), \ \max_{x \in X} J_k(x) - J_{k+1}(x) \} \]

\[ \leq \max \{ \max_{x \in X} J_k(x) - J_{k-1}(x), \ \max_{x \in X} J_{k-1}(x) - J_k(x) \} \]

\[ = \max_{x \in X} |J_k(x) - J_{k-1}(x)| =: \gamma(k - 1) \]

\[ \square \]

**Remark 7.5** (Local convergence). Effectively, Theorem 7.6 states that the DP procedure (7.31)-(7.34) does not exhibit quasi-stationary behavior (confer Figure 7.9), i.e. it cannot happen that a succession of functions \( J_k \) differ only slightly and then suddenly a drastic change in \( J_{k+1} \) in one additional iteration step of the DP. However, note that we compare functions \( J_k \) and \( J_{k+1} \) over the whole feasible state space and not for a single point \( \bar{x} \). For a fixed \( \bar{x} \) the local convergence rate \( \gamma(k, \bar{x}) = |J_k(\bar{x}) - J_{k+1}(\bar{x})| \) is not necessarily a monotonic function.

\[ \square \]

**Theorem 7.7** (Stabilizing sub-optimal control). Let Assumption 7.2 hold and let \( J_k : \mathcal{X}_\infty \rightarrow \mathbb{R}_{\geq 0} \) be some arbitrary finitely bounded function with \( J_k(0_n) = 0 \).

If \( J_k(x) \geq (TJ_k)(x) \) for all \( x \in \mathcal{X}_\infty \) then \( \rho(x) := \arg\min_{u \in \mathcal{U}} g(x, u) + J_k(f_{\text{PWA}}(x, u)) \) is a globally asymptotic stabilizing (sub-optimal) control law for all \( x \in \mathcal{X}_\infty \). Moreover, \( J_k \) is a global Lyapunov function for the controlled system when the control law \( \rho \) is applied.

\[ \square \]

**Proof** First, from \( J_k(x) \geq (TJ_k)(x) = g(x, \rho(x)) + J_k(f_{\text{PWA}}(x, \rho(x))) \) follows directly that \( -\Delta J_k(x) := J_k(x) - J_k(f_{\text{PWA}}(x, \rho(x))) \geq g(x, \rho(x)) \geq \|Qx\|_p \) for all \( x \in \mathcal{X}_\infty \). Because \( Q \) is of full column rank, there always exists a finite \( \alpha_1 > 0 \) with \( \|Qx\|_p \geq \alpha_1 \|x\|_2 \) and thus \( -\Delta J_k(x) \geq \alpha_1 \|x\|_2 \) for all \( x \in \mathcal{X}_\infty \). This means that \( -\Delta J_k(\cdot) \) is always bounded below by some \( K \)-class function. Second, from \( J_k(\cdot) \geq 0 \) it follows \( J_k(x) \geq (TJ_k)(x) \geq \min_{u \in \mathcal{U}} g(x, u) \geq \|Qx\|_p \geq \alpha_2 \|x\|_2 \) for some finite \( \alpha_2 > 0 \). Third, \( J_k \) is bounded over a bounded set and \( J_k(0_n) = 0 \). From these three statements it follows that \( J_k \) is a global Lyapunov function [Vid93] on \( \mathcal{X}_\infty \) and thus the stability argument of \( \rho \) follows directly.
This result is somewhat intuitive but at the same time very interesting: it allows one to compute in an ‘easy’ way a stabilizing controller $\tilde{\mu}$ as well as a bound $T J_k$ on the optimality of the controller. Simultaneously a Lyapunov function, $J_k$, for the controlled system is given. Moreover, this result shows that if the value function iteration is started with some $J_0$ (confer e.g. Theorem 7.5(b)) and at some iteration $k$ of the dynamic program it is detected that $J_k(\cdot) \geq (T J_k)(\cdot)$, then at all the following iteration steps of the dynamic program a stabilizing controller and a continuously improving optimality bound is computed. For initial value functions $J_0$ which fulfill the conditions of Theorem 7.5(c)–(d) this is already true from the beginning, i.e. for all $k \geq 0$. Please note that for initial value functions $J_0$ which fulfill the classical convergence conditions of Theorem 7.5(a) this stabilization and optimality bound property can only be given after computing the limit function $J^*_\infty$ but not already for the intermediate iteration steps.

As observed by the authors in [BCM03a], it may happen that the dynamic program (7.31)–(7.34) converges after a finite number of steps $k = k_\infty + 1 < \infty$. Here we mean by convergence that in two successive iterations of the dynamic program the value functions as well as their domains do not change, i.e. $J_k(\cdot) \equiv J_{k-1}(\cdot)$. Thus we stop the dynamic program when the following condition is met

$$\forall j_1 \in \{1, \ldots, N^k\} \quad \exists j_2 \in \{1, \ldots, N^{k-1}\} \text{ such that}$$

$$P_{j_1} \equiv P_{j_2}, \quad \|\Phi_{j_1}^k - \Phi_{j_2}^{k-1}\| < \varepsilon, \quad \|\Gamma_{j_1}^k - \Gamma_{j_2}^{k-1}\| < \varepsilon \tag{7.51}$$

where $\varepsilon \geq 0$ is some small tolerance.

This would seem to imply that the optimal control law steers any feasible state $x(0)$ after at most $k_\infty$ time steps to the origin. However, several observations should be made. First we should point out that the $1-/\infty$-norm CITOC problem may lead to two types of solutions: (a) an optimal control sequence that in a finite number of time steps steers the state to the origin, and (b) an optimal control sequence that takes an infinite number of time steps to steer the state to the origin. This type of behavior may be observed even for constrained linear systems as shown in the following Example 7.2.

**Example 7.2** (CITOC of a constrained LTI system). Consider the one-dimensional constrained linear system

$$\begin{cases} x(t + 1) &= 0.3 x(t) + u(t) \quad \text{if} \quad 0.2 x(t) - u(t) \leq 0, \\ x(t) &\in [-5, 5], \quad \text{and} \\ u(t) &\in [-1,1]. \end{cases}$$

The CITOC problem (7.25)–(7.26) is solved with $Q = 1, R = 1, \text{and } \Lambda_0 = [-5, 5]$ for $p = \infty$.

The feasible state space and the optimal infinite time control law are depicted in the extended state-input space in Figure 7.10. If one considers $x(t) \in (0, 5]$ it is obvious that the infinite time optimal control law is in fact $\mu^*_\infty(x(t)) = 0.2 x(t)$ and therefore the optimal
Figure 7.10: Feasible state-input space and optimal infinite time control law for Example 7.2.

Figure 7.11: Closed-loop simulation for Example 7.2 for 3 different initial values.

The value function or ‘cost-to-go’ can be computed with \( x(t) = 0.5^t x(0) \) as

\[
J^*_\infty(x(0)) = \min_{U_\infty} \lim_{T \to \infty} \sum_{t=0}^{T} |x(t)| + |u(t)|
\]

\[
= \lim_{T \to \infty} \sum_{t=0}^{T} 1.2 \cdot 0.5^t |x(0)| = 2.4x(0).
\]
From this we see that for any state starting in \((0, 5]\) it will take an infinite number of steps until the state reaches the origin; whereas if the state \(x(t) \in \left[-\frac{10}{3}, 0\right]\) the infinite time optimal control law is given by \(\mu_\infty^*(x(t)) = -0.3x(t)\) and thus drives the state in one single step to the origin.

With similar considerations for the rest of the feasible state space one finds for the infinite time optimal control law

\[
\mu_\infty^*(x(t)) = \begin{cases} 
0.2x(t) & \text{if } x(t) \in [0, 5], \\
-0.3x(t) & \text{if } x(t) \in \left[-\frac{10}{3}, 0\right], \\
1 & \text{if } x(t) \in [-5, -\frac{10}{3}] 
\end{cases}
\]

and for the cost-to-go

\[
J_\infty^*(x(0)) = \begin{cases} 
2.4x(0) & \text{if } x(0) \in [0, 5], \\
-1.3x(0) & \text{if } x(0) \in \left[-\frac{10}{3}, 0\right], \\
-1.39x(0) - 0.3 & \text{if } x(0) \in [-5, -\frac{10}{3}] 
\end{cases}
\]

The optimal infinite time closed-loop trajectory for three different initial values is depicted in Figure 7.11. 

\[\square\]

### 7.4.2 An Efficient Algorithm for the CITOC Solution

In this section we first describe how the properties of the solution to the rather general DP problem (7.31)–(7.34) can be exploited and used for an efficient computation of the infinite time solution. Subsequently, the specific implementation of the algorithm is presented.

In the rest of the chapter for some set or function \(\tau\) we denote as \(\tilde{\tau}\) its restriction to the neighborhood of the origin \((x, u) = (0, 0)\). For instance, \(\tilde{\mathcal{D}}\) describes the domain of those PWA dynamics that are valid for the origin \(\{\tilde{\mathcal{D}}_i\}_{i=1}^{n_d} := \{\mathcal{D}_j \mid [0] \in \mathcal{D}_j\}_{j=1}^{n_d}\), while \(\tilde{f}_{\text{PWA}}\) represents the restriction of the function \(f_{\text{PWA}}\) to that domain. Furthermore, equivalently to Section 7.3.3, when we say SOLVE iteration \(k\) of a DP, we mean formulate a multi-parametric linear program for it and obtain a triplet of expressions for the value function, the control function (optimizer), and the polyhedral partition of the feasible state space

\[
(J_k(x), \mu_k(x), \{P_i^{k}\}_{i=1}^{N_k}) \quad \text{(7.52)}
\]

cf. Algorithm 7.2 and 7.3. By inspection of the DP problem (7.31)–(7.34) we see that at each iteration step we are solving \(n_d N^{k-1}\) mp-LPs. After that, by using polyhedral manipulation we have to compare all generated regions, check if they intersect and remove the redundant ones, before storing a new partition that has \(N^k\) regions.

For a system with only one equilibrium point, i.e. the origin \((0, 0)\) itself, all trajectories that converge to that point in an infinite number of time steps have to go through some of the PWA dynamics associated with the region \(\tilde{\mathcal{D}}_i, i = 1, \ldots, n_d\), and regions \(\tilde{\mathcal{P}}_j^\infty, j = 1, \ldots, N^\infty\), that are touching the origin, cf. Example 7.2. Thus at the beginning, instead of focusing on the
whole feasible state space and all dynamics, we can limit our algorithm to the neighborhood of the origin and only after attaining convergence do we proceed with the exploration of the rest of the state space. In this way at each iteration of the DP we would – on average – have to solve a much smaller number of mp-LPs. We will call the solution to such a restricted problem the ‘core’ \( C_0 \). Note that in general the core \( C_0 \) is a non-convex polyhedral partition.

Any positive invariant set is a valid candidate for the core \( C_0 \), as long as an associated control strategy is feasible and steers the state to the origin. The only prerequisite is that for any given initial feasible state, i.e. the state for which the original problem has a solution, we can reach at least one element of the core in a finite number of time steps. However, as its name says, the core is used as a ‘seed’ for the future construction and exploration of the feasible state space. Thus, obtaining a good sub-optimal solution is desirable.

The task of solving the CITOC problem (7.25)–(7.26) is split into two sub-problems and respective algorithms. In the first algorithm (Algorithm 7.2) we explore the portion of the state space around the origin and construct the core of the infinite time solution. In the second algorithm (Algorithm 7.3), starting from the core \( C_0 \), we build a sequence of additions – named ‘rings’ \( R_k \) – to the core \( C_{k-1} \) until the algorithm converges for the whole feasible state space \( X_{\infty} \). At the end we have the infinite time solution \( S_{\infty}^* \). Here with \( C \), \( R \), and \( S \) we denote the triplets of the form given in Equation (7.52).

In an ideal scenario the core \( C_0 \) would already be a part of an infinite time optimal solution, and every ring \( R_k \) would also be a part of an infinite time solution. Then in all intermediate steps we would have to explore only the one step ahead optimal transitions from all PWA dynamics to the latest ring (instead of going from all dynamics to the initial core and all previous rings). In practice we are likely to observe sub-optimal scenarios: the newly generated ring, \( R_k \), may contain polyhedra with associated cost functions that are ‘worse’ (meaning bigger) than the infinite time solution and thus such polyhedra will be altered in the future steps of the algorithm.
Algorithm 7.2 (Generating the CORE of the infinite time solution).

**INPUT** \( k_{\text{max}}, \varepsilon, \hat{f}_{\text{PWA}}(x, u), \{\hat{D}_i\}_{i=1}^{\hat{n}_d}, p, Q, R \)

**OUTPUT** The core \( C_0 \)

**LET** \( \hat{S}_0 \leftarrow (\hat{J}_0(x) = 0, \hat{\mu}_0(x) = 0, \{\hat{P}_j^0 = \mathbb{R}^n\}) \)

**LET** \( k = 0, \text{finished} = \text{FALSE} \)

**WHILE** \( k < k_{\text{max}} \) AND NOT finished

**LET** \( k \leftarrow k + 1 \)

**FOR** \( i = 1 \) **TO** \( \hat{n}_d \)

**FOR EACH** \( \hat{P}_j^{k-1} \in \hat{S}_{k-1} \)

\[ s_{i,j} \leftarrow \text{SOLVE} \min_u \|Qx\|_p + \|Ru\|_p + \hat{J}_{k-1}(\hat{f}_{\text{PWA}}(x, u)) \]

subj. to \[ \begin{cases} [x', u'] \in \hat{D}_i, \\ \hat{f}_{\text{PWA}}(x, u) \in \hat{P}_j^{k-1} \end{cases} \]

**END**

**LET** \( \hat{S}_k \leftarrow \text{INTERSECT & COMPARE} \{s_{i,j}\} \)

**LET** \( \hat{S}_k \leftarrow \text{RESTRICTION of} \ S_k \) to the origin

**IF** \( \hat{S}_k \equiv \hat{S}_{k-1} \) (in the sense of Condition (7.51))

**THEN** \( \text{finished} = \text{TRUE}, C_0 \leftarrow \hat{S}_k \)

**END**

As stated in Section 7.3.3, the function INTERSECT & COMPARE removes such polyhedra that are completely covered with other polyhedra [BT03] which have a corresponding ‘better’ (meaning smaller) value function expression. If some polyhedron \( P_j^k \) is only partially covered with better regions the part of \( P_j^k \) with the smaller cost can be partitioned into a set of convex polyhedra. Thus we preserve the polyhedral nature of the feasible state space partition in each iteration of the Algorithm 7.2. Note that in the SOLVE step we are solving a smaller number of problems than in the general DP. Since we are restricting ourselves to the neighborhood of the origin, the number of regions at each step is likely to remain rather small and should stay constant after a certain number of iterations. However, the choice of the initial \( \hat{J}_0 \equiv 0 \) may lead to a big number of iterations depending on the desired precision. If a better or other initial guess for \( \hat{J}_0 \) is known, confer Section 7.4.3, it can be used to speed up Algorithm 7.2. Note that \( C_0 \) is a positive invariant set by construction.

After we have constructed the initial core, \( C_0 \), we can proceed with the exploration of the rest of the state space as described in the following Algorithm 7.3.
Algorithm 7.3 (Generating the infinite time solution).

**INPUT** \( k_{\text{max}}, \varepsilon, f_{\text{PW A}}(x, u), \{D_i\}_{i=1}^{n_d}, \) the core \( C_0, p, Q, R \)

**OUTPUT** The infinite time solution \( S^*_{\infty} \)

**LET** Solution \( S_0 \leftarrow C_0 \)

**LET** Ring \( R_0 \leftarrow C_0 \)

**LET** \( k = 0, \) finished = FALSE

**WHILE** \( k < k_{\text{max}} \) AND NOT finished

**LET** \( k \leftarrow k + 1 \)

**FOR** \( i = 1 \) TO \( n_d \)

**FOR EACH** \( P_{k-1} \in R_{k-1} \)

\[ s_{i,j} \leftarrow \text{SOLVE} \quad \min_u \|Qx\|_p + \|Ru\|_p + J_{k-1}(f_{\text{PW A}}(x, u)), \]

**subj. to** \[
\begin{cases}
|x', u'|' \in D_i, \\
f_{\text{PW A}}(x, u) \in P_{j-1}
\end{cases}
\]

**END**

**END**

**LET** \( S_k \leftarrow \text{INTERSECT} \& \text{COMPARE} \) \( S_{k-1}, \{s_{i,j}\} \)

**LET** \( C_k \leftarrow S_k \cap S_{k-1} \) (in the sense of Condition (7.51))

**LET** \( R_k \leftarrow S_k \setminus C_k \)

**IF** \( R_k = \emptyset \)

**THEN** finished = TRUE, \( J^*_{\infty}(x) \leftarrow J_k(x), S^*_{\infty} \leftarrow S_k \)

**END** \[ \square \]

Note that if Algorithm 7.2 ends successfully and if any optimal closed-loop trajectory starting in \( C_0 \) happens to stay in \( C_0 \) for all time then the value function computed in Algorithm 7.2 associated to \( C_0 \) is in fact the optimal value function \( J^*_\infty(\cdot) \) associated to the feasible set of \( C_0 \). However, if any optimal closed-loop trajectory starting in \( C_0 \) leaves this set in finite time then the value function computed in Algorithm 7.2 associated to \( C_0 \) is the best current upper bound of the optimal value function \( J^*_\infty(x(0)) \) for all \( x(0) \in C_0 \). In the case of the later scenario, by optimality, Algorithm 7.3 will account for this fact and improve the cost in the corresponding part of \( C_0 \) by going through additional DP iterations until optimality is reached. The same argument holds for all the computed intermediate ‘cores’, \( C_k \), of Algorithm 7.3 as well.

### 7.4.3 Alternative Choices of the Initial Value Function \( J_0 \)

Theorem 7.5 guarantees the convergence of the dynamic programming based algorithm to the optimal solution \( J^*_\infty(\cdot) \) starting from an almost arbitrary initial value function \( J_0(\cdot) \). This can be used in order to decrease or limit the number of iterations needed in the proposed algorithm. Moreover, at the same time an upper bound to the optimal solution can be given.
From the above discussion and from Theorem 7.4 we know that \( J^*_\infty(0) = 0 \) as well as \( \mu^*_\infty(0) = 0 \); and for some set \( \mathcal{X} \subseteq \mathcal{X}_\infty \) around the origin the optimal value function \( J^*_\infty(\cdot) \) and the optimal input function \( \mu^*_\infty(\cdot) \) is a piecewise linear function of the state \( x \).

In Algorithm 7.2 we limit the exploration and computation to such regions around the origin and consider only the domain of the system dynamics that touch the origin or have the origin in the interior.

**J_0 as an Upper Bound to \( J^*_\infty \) by Approximation**

Consider the linear, stabilizable system

\[
x(t + 1) = Ax(t) + Bu(t).
\]

with polyhedral state and input constraints, i.e. \( (x, u) \in \mathcal{X} \times \mathcal{U} \). In addition, consider for this system the stabilizing state feedback control law for states \( x(t) \in \mathcal{X} \) around the origin

\[
u(x(t)) = Kx(t) \quad \text{with} \quad \|A_{\text{CL}}\|_p < 1
\]

where \( A_{\text{CL}} := A + BK \) and \( \|A_{\text{CL}}\|_p \) with \( p \in \{1, \infty\} \) denotes the standard Hölder matrix 1- or \( \infty \)-norm [Lüt96, HJ85] of \( A_{\text{CL}} \).

For control law (7.54) we have for all \( x(0) \in \mathcal{X} \) that

\[
J^*_\infty(x(0)) \\
\leq J_\infty(x(0), u = Kx) \\
= \lim_{T \to \infty} \sum_{t=0}^{T} \|QA^t_{\text{CL}}x(0)\|_p + \|RK(A^t_{\text{CL}}x(0))\|_p \\
\leq \lim_{T \to \infty} \sum_{t=0}^{T} (\|Q\|_p + \|RK\|_p) \cdot \|A^t_{\text{CL}}\|_p \cdot \|x(0)\|_p \\
\leq (\|Q\|_p + \|RK\|_p) \cdot \|x(0)\|_p \cdot \lim_{T \to \infty} \sum_{t=0}^{T} \|A_{\text{CL}}^t\|_p \\
\leq \left\| \frac{\|Q\|_p + \|RK\|_p}{1 - \|A + BK\|_p} \cdot I \cdot x(0) \right\|_p \\
=: \bar{J}_\infty(x(0), u = Kx).
\]

Clearly, for any bounded \( x(0) \) we have that \( \infty > \bar{J}_\infty(x(0), u = Kx) \geq J^*_\infty(x(0)) \). However, finding a tight bound \( \bar{J}_\infty(x(0), u = Kx) \) on \( J^*_\infty(x(0)) \) for all \( x(0) \in \mathcal{X} \), i.e.

\[
\min_k \quad \left\| \frac{\|Q\|_p + \|RK\|_p}{1 - \|A + BK\|_p} \right\|_p \\
\text{subj. to} \quad \|A + BK\|_p < 1,
\]

might be a difficult non-linear or even infeasible problem due to Condition (7.59), depending on the system data.
$J_0$ as an Upper Bound to $J^*_\infty$ by Lyapunov Function Construction

In the case of $p = \infty$, for a given stabilizing state feedback control law $u(x) = Kx$ for the system (7.53), we can always efficiently compute a Lyapunov function for the closed-loop system of the type

$$V(x) = \|Wx\|_\infty$$

with $W \in \mathbb{R}^{l \times n}$, $\text{rank}(W) = n$, $\infty > l \geq n$ as proposed in [KAS92, Pol95, Bit88]. In order to guarantee that such a Lyapunov function is always an upper bound to $J^*_\infty(\cdot)$ the scaling

$$\bar{J}_0(x, u = Kx) := \alpha \|Wx\|_\infty$$

with $\alpha \in \mathbb{R}_{>0}$ needs to be performed.

Due to stabilizability of the system (7.53) one can always design a controller gain $K$ such that the closed-loop system matrix $A_{CL}$ has at least one real eigenvector $e$ with its corresponding real eigenvalue $\lambda$. Then it follows for an initial condition $\bar{x}(0) = re$ with $r \in \mathbb{R}_{\neq 0}$

$$J_\infty(\bar{x}(0), u = Kx)$$

$$= \lim_{T \to \infty} \sum_{t=0}^{T} \|QA_{CL}^t re\|_\infty + \|RK^t e\|_\infty$$

$$= |r| \cdot (\|Qe\|_\infty + \|RKe\|_\infty) \lim_{T \to \infty} \sum_{t=0}^{T} |\lambda|^t$$

$$= \frac{|r|}{1 - |\lambda|} (\|Qe\|_\infty + \|RKe\|_\infty).$$

Consequently, for the choice of

$$\alpha := \frac{J_\infty(\bar{x}(0), u = Kx)}{\|W \bar{x}(0)\|_\infty} = \frac{\|Qe\|_\infty + \|RKe\|_\infty}{(1 - |\lambda|)\|We\|_\infty}$$

we have $J^*_\infty(x) \leq \bar{J}_0(x, u = Kx) < \infty$ for all $x \in \mathcal{X}$.

**Theorem 7.8.** Let $J_0(\cdot)$ be in the class of functions described by Equation (7.61) and (7.64). If $J_0(\cdot)$ is used as initial value function of the dynamic programming iteration (7.31)–(7.32) for the system (7.53) then $\lim_{k \to \infty} (T^k J_0) = J^*_\infty$. □

**Proof** $J_0$ is a Lyapunov function by construction, thus we have $J_0(x(t)) \geq g(x(t), u(t) = Kx(t)) + J_0((A + BK)x(t)) \geq \min_{u(t)} g(x(t), u(t)) + J_0(Ax(t) + Bu(t)) =: (T^k J_0)(x(t))$. Using Theorem 7.5(c), then $\lim_{k \to \infty} (T^k J_0) = J^*_\infty$. □
Figure 7.12: State space partition of the value function $J^*_{\infty}(\cdot)$ of the infinite time solution $S^*_{\infty}$. Same coloring implies the same cost value.

7.4.4 Examples

**Example 7.3** (Constrained PWA system). Consider again the constrained piecewise affine system [BM99]

$$x(t + 1) = 0.8 \begin{bmatrix} \cos \alpha(x(t)) - \sin \alpha(x(t)) \\ \sin \alpha(x(t)) \cos \alpha(x(t)) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u(t),$$

$$\alpha(x(t)) = \begin{cases} \frac{\pi}{3} & \text{if } [1 0] x(t) \geq 0, \\ -\frac{\pi}{3} & \text{if } [1 0] x(t) < 0, \end{cases}$$

$$x(t) \in [-10, 10] \times [-10, 10],$$

$$u(t) \in [-1, 1].$$

(7.65)

The constrained infinite time optimal control problem (7.25)–(7.26) is solved with $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 1$, and $X^0 = [-10, 10] \times [-10, 10]$ for $p = \infty$.

As described in Section 7.4.2 the algorithm is divided into two parts: first the so called ‘inner core’ $C_0$ is constructed via a dynamic programming approach (Algorithm 7.2). After this inner core has converged it serves as a ‘seed’ (or optimal current upper bound of the value function restricted to the part of the state space in this particular step) for the second part of the algorithm (Algorithm 7.3) where from the seed the rest of the feasible state space is explored until the piecewise affine value function for the whole feasible state space $X_\infty$ does not change for two successive steps in the exploration procedure.

For Example 7.3 the inner core, $C_0$, is computed in 11.6 seconds in 5 iteration steps. Figure 7.13(a) shows the state space partitioning comprising 10 polyhedral regions of the value function of the inner core. Figure 7.13(b) shows the state space partition of the value
Figure 7.13: state space partition of the value function and of the piecewise affine control law of Example 7.3. Same coloring implies the same cost value or to the same affine control law, respectively.

function with 188 polyhedral regions at the intermediate step \( k = 4 \) of the second part of the algorithm. Figure 7.13(e) shows the state space partition \( \mathcal{C}_4 \) of the current optimal upper bound of the infinite horizon value function which does not change from the intermediate step \( k = 3 \) to \( k = 4 \) and it consists of 104 regions. This can be viewed as the new ‘core’ in step \( k = 4 \) from which the ‘ring’ \( \mathcal{R}_4 \) in Figure 7.13(d) was computed in step \( k = 4 \). After \( k = 7 \) steps of the second part of the algorithm the whole feasible state space is explored and the value function \( J_k(x) \) does not change from step \( k = 7 \) to \( k = 8 \) (Figure 7.13(e) and Figure 7.12) and thus the infinite horizon solution \( J^*_\infty(x) \) is obtained. The constructed state space partition consists of 252 polyhedral regions. The PWA control law of the infinite horizon solution consists of 23 different piecewise affine control laws and is depicted in Figure 7.13(f). The same coloring implies the same affine control law.

The infinite time solution to (7.25)–(7.26) for this example was solved in 184 seconds on a Pentium 4, 2.2 GHz machine running MATLAB® 6.1. This shows the efficiency of the proposed algorithm compared to the approach given in Section 7.3 or [BCM03a] where the computation of the infinite horizon solution took 1515 seconds on the same machine. \( \square \)
7.5 Software Implementation

The presented algorithms to solve the CFTOC problem (7.4)–(7.5) and the CITOC problem (7.25)–(7.26) are implemented in the Multi-Parametric Toolbox (MPT) [KGB04] for MATLAB®. The toolbox can be downloaded for free at:

http://control.ee.ethz.ch/~mpt/
Piecewise affine (PWA) systems are powerful models for describing both non-linear and hybrid systems. One of the key problems in controlling these systems is the inherent computational complexity of controller synthesis and analysis, especially if constraints on states and inputs are present. In addition, few results are available which address the issue of computing stabilizing controllers for PWA systems without placing constraints on the location of the origin.

This chapter first introduces a method to obtain stability guarantees for receding horizon control of discrete-time PWA systems. Based on this result, three algorithms which provide low complexity state feedback controllers are introduced. Specifically, we demonstrate how multi-parametric programming can be used to obtain minimum-time controllers and ‘low-switching’ controllers, i.e. controllers which drive the state into a pre-specified target set in minimum time, and stabilizing controllers which attempt to remain within one (affine) system dynamic for as long as possible. In a third segment, we show how controllers of even lower complexity can be obtained by separately dealing with constraint satisfaction and stability properties. To this end, we introduce a method to compute PWA Lyapunov functions for discrete-time PWA systems via linear programming. Finally, we report results of an extensive case study which justify our claims of complexity reduction.

8.1 Introduction

Optimal control of piecewise affine (PWA) systems has garnered increasing interest in the research community since this system type represents a powerful tool for approximating non-linear systems and because of their equivalence to many classes of hybrid systems [HSB01, Son81]. The optimal control inputs for discrete-time PWA systems may be obtained by solving mixed-integer optimization problems on-line [BM99, MR03], or as was shown in [BCM03a, BBBM03a, KM02, Bor03], by solving off-line a number of multi-parametric programs. By multi-parametric programming, a linear (mp-LP) or quadratic (mp-QP) optimization problem is solved off-line for a range of parameters.

In their pioneering work [BMDP02] the authors show how to formulate an optimal control problem for constrained linear discrete-time systems as a multi-parametric program (by
treating the state vector as a parameter) and how to solve such a program. Basic ideas from [BMDP02] for linear systems were extended to PWA systems in [BCM03a, BBBM03a, KM02, Bor03]. The associated solution (optimal control inputs) takes the form of a PWA state feedback law. In particular, the state-space is partitioned into polyhedral sets and for each of these sets the optimal control law is given as an affine function of the state. In the on-line implementation of such controllers, input computation reduces to a simple set-membership test. Even though the approaches in [BCM03a, BBBM03a, KM02, Bor03] rely on off-line computation of a feedback law, the computation quickly becomes prohibitive for larger problems. This is not only due to the high complexity of the multi-parametric programs involved [GM03], but mainly because of the large number of multi-parametric programs which need to be solved when a controller is computed in a dynamic programming fashion [BBBM03a, KM02].

In addition, there are few results in the literature which explicitly address the issue of computing feedback controllers for PWA systems which provide stability guarantees. The few publications that address this issue (e.g., [MR03]) assume that the origin is contained in the interior of one system dynamic. The only exception is the infinite horizon solution proposed in [BCM03b], which is computationally intractable for larger problems.

This chapter addresses the clear need for low complexity controllers for piecewise affine systems that provide stability guarantees even if the origin is located on the boundary of several different system dynamics. Three algorithms are presented in this chapter which achieve this goal.

First, a general scheme for obtaining stability guarantees for generic PWA systems subject to receding horizon control will be presented. This scheme can be used in connection with other controller computation methods (e.g., [MR03, BCM03a, BBBM03a, KM02]) to obtain stability guarantees.

Subsequently, the computation of a minimum-time feedback controller is presented as well as a control scheme that aims at obtaining a low (but not necessarily minimal) number of switches in the system dynamics. As the final section will show, the resulting controllers are of such low complexity compared to what one can obtain with traditional methods [BCM03a, BBBM03a] that a whole new class of problems becomes tractable.

In a third segment, we show how controllers of even lower complexity can be obtained by separately dealing with the issue of constraint satisfaction and asymptotic stability. To this end, we introduce a method to compute a PWA Lyapunov function for discrete-time PWA systems via linear programming. The computation is guaranteed to find a PWA Lyapunov function for a given partition, if it exists. The numerical results in the final section illustrate several cases where the PWA Lyapunov analysis succeeds and the search for a PWQ Lyapunov function via LMIs [GLPM03, Joh03, FTCMM02] fails.

8.2 Problem Description and Properties

This section covers some of the fundamentals of multi-parametric programming for linear systems before restating recent results for PWA systems. Consider a discrete-time linear
time-invariant system
\[ x(k + 1) = Ax(k) + Bu(k) \]  
(8.1)

with \( A \in \mathbb{R}^{n \times n} \) and \( B \in \mathbb{R}^{n \times m} \). Let \( x(k) \) denote the measured state at time \( k \) and \( x_k (u_k) \) the predicted state (input) at time \( k \), given \( x(0) \). Assume now that the states and the inputs of the system in (8.1) are subject to the following constraints
\[ x(k) \in \mathbb{X} \subseteq \mathbb{R}^n, \quad u(k) \in \mathbb{U} \subseteq \mathbb{R}^m, \quad \forall k \geq 0, \]  
(8.2)

where \( \mathbb{X} \) and \( \mathbb{U} \) are polytopic sets containing the origin in their interior.

**Remark 8.1.** For ease of notation, we restrict ourselves to separate constraints on state and input in (8.2). It is straightforward to modify all algorithms in this chapter to deal with systems subject to mixed constraints, i.e. \( C^x x(k) + C^u u(k) \leq C^c, \forall k \geq 0. \)

Consider the constrained finite-time optimal control problem with a linear objective
\[
J^*_N(x(0)) = \min_{u_0, \ldots, u_{N-1}} \sum_{k=0}^{N-1} \left( \|Ru_k\|_{1,\infty} + \|Qx_k\|_{1,\infty} \right) + \|Qf x_N\|_{1,\infty},
\]  
(8.3a)

subj. to, \( x_k \in \mathbb{X}, \quad u_{k-1} \in \mathbb{U}, \quad \forall k \in \{1, \ldots, N\}, \) \( x_N \in \mathbb{T}_{\text{set}}, \) \( x_{k+1} = Ax_k + Bu_k, \quad x_0 = x(0), \)  
(8.3b, 8.3c, 8.3d)

where (8.3c) is a user defined set-constraint on the final state and \( \| \cdot \|_{1,\infty} \) denotes the 1- or \( \infty \)-norm of a vector, respectively.

**Definition 8.1.** We define the \( N \)-step feasible set \( \mathcal{F}_N \subseteq \mathbb{R}^n \) as the set of initial states \( x(0) \) for which the optimal control problem (8.3) is feasible, i.e.
\[
\mathcal{F}_N = \{ x(0) \in \mathbb{R}^n \mid \exists U_N \in \mathbb{R}^{N m},
\]
\[
x_k \in \mathbb{X}, \quad x_N \in \mathbb{T}_{\text{set}}, \quad u_{k-1} \in \mathbb{U}, \quad \forall k \in \{1, \ldots, N\}\}
\]

where \( U_N = [u'_0, \ldots, u'_{N-1}]' \) is the optimization vector. By considering \( x(0) \) as a parameter, problem (8.3) can be stated as an mp-LP [BBM00b] that can be solved to obtain a feedback solution with the following properties (derived from [Bor03, Gal95]):

**Theorem 8.1.** Consider the finite time constrained regulation problem (8.3), with a linear objective in (8.3a). Then, the set of feasible parameters \( \mathcal{F}_N \) is convex, there exists an optimizer \( U^*_N : \mathcal{F}_N \to \mathbb{R}^{N m} \) which is continuous and piecewise affine (PWA) over polyhedra, i.e.
\[
U^*_N(x(0)) = F_r x(0) + G_r, \quad \text{if} \quad x(0) \in \mathcal{P}_r
\]
\[
\mathcal{P}_r = \{ x \in \mathbb{R}^n \mid H_r x \leq K_r \}, \quad r = 1, \ldots, R
\]

and the value function \( J^*_N : \mathcal{F}_N \to \mathbb{R} \) is continuous, convex and piecewise affine.
According to Theorem 8.1, the feasible state space $\mathcal{F}_N$ is partitioned into $R$ polytopic regions, i.e. $\mathcal{F}_N = \bigcup_{r=1,...,R} \mathcal{P}_r$. The results in [BBM00b] were extended in [BBBM03a] to compute the optimal explicit feedback controller for PWA systems of the form
\[
\begin{align*}
x(k+1) &= A_i x(k) + B_i u(k) + f_i, \quad \text{if } x(k) \in D_i, \quad i \in \mathcal{I}
\end{align*}
\]
subject to the constraints (8.2). The set $\mathcal{I}$ is defined as $\mathcal{I} := \{1, 2, \ldots, D\}$ where $D$ denotes the number of different dynamics. We will henceforth assume that the different dynamic regions $D_i$ are non-overlapping. Henceforth, we will abbreviate (8.4a) and (8.4b) with $x(k+1) = f_{\text{PWA}}(x(k), u(k))$. The optimization problem considered here is thus given by
\[
\begin{align*}
J_N^* (x(0)) &= \min_{u_0, \ldots, u_{N-1}} \sum_{k=0}^{N-1} \left( \|Ru_k\|_{1,\infty} + \|Qx_k\|_{1,\infty} \right) + \|Q_f x_N\|_{1,\infty}, \\
\text{subj. to } & \quad x_N \in \mathcal{T}_{\text{set}}, \\
& \quad x_k \in \mathcal{X}, \quad u_{k-1} \in \mathcal{U}, \quad k \in \{1, \ldots, N\}, \\
& \quad x_{k+1} = f_{\text{PWA}}(x_k, u_k), \quad x_0 = x(0),
\end{align*}
\]
In [BCM03a], multi-parametric Linear Programs (mp-LP) were solved in a dynamic programming fashion to obtain the feedback solution to (8.5). In [BBBM03a], the feedback solution to (8.5) with a quadratic objective in (8.5a) was computed by solving a sequence of multi-parametric Quadratic Programs (mp-QP) in a dynamic programming fashion. Methods to obtain feedback solutions to linear or quadratic optimization problems for PWA systems are also given in [Bor03, MR03, BCM03b, KM02]. Note that we do not require $f_{\text{PWA}} : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$ to be continuous. However if $f_{\text{PWA}}$ is discontinuous, computing the solution to (8.5), if one exists, becomes rather cumbersome since special care has to be taken of the open and closed boundaries of $D_i$.

### 8.3 Computation of Stabilizing Controllers for Piecewise Affine Systems

A large part of the literature has focussed on end-point constraints to guarantee asymptotic stability of the closed-loop PWA system (e.g., [BM99, Bor03]). This type of constraint generally requires the use of large prediction horizons for the controller to cover the entire controllable state space, such that the computational complexity quickly becomes prohibitive. Other methods (e.g., [MR03]) only provide stability guarantees if the origin is contained in the interior of one of the dynamics $D_i$.

In this section a method is presented for obtaining stabilizing controllers for generic PWA systems. For any dynamical system, stability is guaranteed if an invariant set is imposed as a terminal state constraint (see (8.5b)) and the terminal cost in (8.5) corresponds to a Lyapunov function for that set [MRRS00]. In addition, the decay rate of the 'terminal Lyapunov function' must be greater than the stage cost. We here show how to obtain the
invariant maximum admissible set \( \mathcal{O}_\infty^{\text{PWA}} \) with the associated feedback law and Lyapunov function. In a first step, we select all dynamics \( i \in \mathcal{I}_0 \), which contain the origin, i.e.

\[
\mathcal{I}_0 := \{ i \in \mathcal{I} \mid 0 \in D_i \}.
\]

We are assuming that the origin is an equilibrium state of the PWA system and hence the closed loop dynamics \( f_i = 0, \forall i \in \mathcal{I}_0 \) (see (8.4)). If this assumption is not satisfied, the approach proposed here will fail.

The search for stabilizing piecewise linear feedback controllers \( F_i \) and an associated Lyapunov function \( V(x) = x'Px \) can now be posed as

\[
x'Px \geq 0, \quad \forall x \in \mathcal{X},
\]

\[
x'(A_i + B_i F_i)'P(A_i + B_i F_i)x - x'Px \leq -x'Qx - x'F_i'RF_i x, \quad \forall x \in D_i, \forall i \in \mathcal{I}_0.
\]

If we relax this condition by setting \( D_i = \mathbb{R}^n, \forall i \in \mathcal{I}_0 \), the problem can be rewritten as an SDP by using Schur complements and introducing the new variables \( Y_i = F_i Z \) and \( Z = \frac{1}{\gamma} P^{-1} \) (see [KBM96, MFTM00] for details),

\[
\min_{Y_i, Z, \gamma} \gamma, \quad \text{subj. to,} \quad Z > 0, \quad (8.6a)
\]

\[
\begin{bmatrix}
Z & (A_i Z + B_i Y_i)' & (Q^{0.5} Z)' & (R^{0.5} Y_i)'
(A_i Z + B_i Y_i)' & Z & 0 & 0
(Q^{0.5} Z) & 0 & \gamma I & 0
(R^{0.5} Y_i) & 0 & 0 & \gamma I
\end{bmatrix} \succeq 0, \quad \forall i \in \mathcal{I}_0. \quad (8.6c)
\]

where the scalar \( \gamma \) is introduced to optimize for the worst case performance, whereby the ‘worst case’ corresponds to an arbitrary switching sequence. Note that it may not be possible for the worst case switching sequence considered in (8.6) to occur in practice, since not all dynamics \( i \) are defined over the entire state space.

**Remark 8.2.** If (8.6) is posed for an LTI system (i.e. \( \mathcal{I}_0 = \{ 1 \} \)), the optimal LQR state feedback solution \( K \) and the solution to the Algebraic Riccati Equation \( P \) are recovered.

Alternatively, one can solve a max-det problem to obtain the largest invariant ellipsoidal target set [BGFB94]. Large target sets generally make the subsequent controller computations simpler. Note however, that the feedback laws associated to the maximal volume invariant ellipsoidal set may not yield the maximal volume invariant polytopic set.

In a second step, the maximal admissible set \( \mathcal{O}_\infty^{\text{PWA}} \) of the PWA system subject to the feedback controllers \( F_i = Y_i Z^{-1} \) can be computed with the algorithm in [RGK+04], which is guaranteed to terminate in finite time for the problem at hand, since the closed loop system is asymptotically stable. The proposed computation scheme is summarized in the following algorithm:

**Algorithm 8.1.** Computation of Maximal Admissible Set \( \mathcal{O}_\infty^{\text{PWA}} \)
1. Identify all dynamics \( i \) which contain the origin, i.e. \( \mathcal{I}_0 := \{ i \in \mathcal{I} \mid 0 \in \mathcal{D}_i \} \).

2. Solve (8.6) for all \( i \in \mathcal{I}_0 \), to obtain \( F_i \) and \( P \). If (8.6) is infeasible, abort the algorithm.

3. Compute the maximal output admissible set \( \mathcal{O}^\text{PWA}_\infty \) corresponding to the closed loop system \( x_{k+1} = (A_i + B_i F_i)x_k \), if \( x_k \in \mathcal{D}_i \) and constraints (8.2) with the method in [RGK+04].

4. Return the target set \( \mathcal{O}^\text{PWA}_\infty \), the feedback laws \( F_i \) and the associated matrix \( P \).

**Theorem 8.2.** Assume the optimization problem (8.5) is given with a quadratic objective (8.5a),
\[
J^*_N(x(0)) = \min_{u_0, \ldots, u_{N-1}} \sum_{k=0}^{N-1} \left( u'_k R u_k + x'_k Q x_k \right) + x'_N Q_f x_N,
\]
a terminal set \( \mathcal{T}_\text{set} = \mathcal{O}^\text{PWA}_\infty \) and a terminal cost \( Q_f = P \) (obtained with Algorithm 8.1). If this problem is solved at each time step for the PWA system (8.4) and only the first input is applied (receding horizon control), then the closed loop system is asymptotically stable.

**Proof** The result of Algorithm 8.1 trivially satisfies the conditions for asymptotic stability in [MRRS00, Section 3.3].

Note that we only need to consider a single convex terminal set for linear systems [GT91] whereas for PWA systems, the terminal set \( \mathcal{O}^\text{PWA}_\infty \) is given as a union of several convex sets \( \mathcal{O}^\text{PWA}_\infty = \bigcup \mathcal{O}_i \). If the union \( \bigcup \mathcal{O}_i \) is convex, the regions can be merged with the method in [BFT01]. This is a desirable property since simpler target sets \( \mathcal{T}_\text{set} \) generally lead to reduced algorithm run-time and solution complexity for the type of optimization problem given in (8.5).

**Remark 8.3.** The procedure described in this section is merely sufficient for asymptotic stability. We cannot guarantee that the Lyapunov function and the associated state feedback laws will be found in the suggested manner. However, we have observed in an extensive case study that the approach works very well in practice. Short of the computationally very demanding construction of the infinite horizon solution proposed in [BCM03b], there is currently no alternative method for guaranteeing closed-loop stability for control of generic PWA systems. Furthermore, the method we propose here can easily be combined with most other controller computation schemes (e.g., [BBBM03a, MR03, KM02, BCM03a]).

### 8.4 Computation of Low Complexity Controllers for Piecewise Affine Systems

The goal in this section is the design of explicit state feedback controllers, which ensure that the system constraints (8.2) are satisfied for all time and provide asymptotic stability guarantees. Without loss of generality, we restrict ourselves to the regulation problem, i.e.
how the state $x(k)$ can be steered to the origin without violating any of the system constraints along the closed loop trajectory. General tracking problems can easily be formulated as regulation problems by augmenting the state space appropriately [PK03].

One of the key problems in the control of PWA systems is the lack of convexity of the controlled sets, which produces a significant computational overhead. Furthermore, the complexity of the cost-to-go function in the dynamic programming approach in [BBBM03a, KM02] makes it necessary to explore an exponentially growing number of possible target sets during the iterations. The algorithms presented here avoid these issues to some extent by considering ‘simpler’ control objectives (e.g. minimum time control). Note that all controllers presented here guarantee constraint satisfaction for all time as well as asymptotic stability.

8.4.1 Computation of a Minimum Time Controller

The minimum time controller considered here aims at driving the system state $x(k)$ into a pre-specified target set $O_{\infty}^{\text{PWA}}$ in minimum time. Unlike the approaches in [BBBM03a,KM02], the cost-to-go for the minimum-time controller assumes only integer values. Because of the ‘simple’ cost-to-go, the target sets which need to be considered at each iteration step are larger and fewer in number than those which would be obtained if an optimal controller with a different cost objective were to be computed [BBBM03a, KM02, BCM03b]. Thus, both the complexity of the feedback law as well as the computation time are greatly reduced, in general.

When the proposed algorithm terminates, the associated feedback controller will cover the $N$-step stabilizable set $K_N^{\text{PWA}}(O_{\infty}^{\text{PWA}})$.

Definition 8.2. The set $K_N^{\text{PWA}}(O_{\infty}^{\text{PWA}})$ denotes the $N$-step stabilizable set for a PWA system (8.4), i.e., it contains all states which can be steered into $O_{\infty}^{\text{PWA}}$ in $N$ steps. Specifically,

$$K_N^{\text{PWA}}(O_{\infty}^{\text{PWA}}) = \{ x(0) \in \mathbb{R}^n \mid \exists u(k) \in \mathbb{R}^m, \text{ s.t. } x(N) \in O_{\infty}^{\text{PWA}}, \ x(k) \in X, u(k) \in U, \ x(k+1) = f_{\text{PWA}}(x(k), u(k)), \ \forall k \in \{0, \ldots, N\} \}. $$

Accordingly, the set $K_{\infty}^{\text{PWA}}(O_{\infty}^{\text{PWA}})$ denotes the maximal stabilizable set for $N \to \infty$.

Note that the $N$-step stabilizable set $K_N^{\text{PWA}}(O_{\infty}^{\text{PWA}})$ is a control invariant set. The mathematical formalism of set invariance theory is viability theory, for which a comprehensive theoretical exposition can be found in [Aub91].

Minimum Time Controller: Off-Line Computation

Before presenting the algorithm, some preliminaries will be introduced. Assume a possibly non-convex union $\mathcal{X}^0$ of $L^0$ polytopes $\mathcal{X}^0_l$, i.e. $\mathcal{X}^0 = \bigcup_{l \in L^0} \mathcal{X}^0_l$, where $L^0 := \{1, 2, \ldots, L^0\}$. In the following, the set $\mathcal{X}$ without subscript will be used to denote unions of polytopes while
the subscript is used to denote polytopes. All states which can be driven into the set \( \mathcal{X}^0 \) for the PWA system (8.4) are defined by:

\[
\text{Pre}(\mathcal{X}^0) = \{ x \in \mathbb{X} \mid \exists u \in \mathbb{U}, \ f_{\text{PWA}}(x, u) \in \mathcal{X}^0 \} = \bigcup_{i \in I} \bigcup_{l \in L^0} \left\{ x \in \mathbb{X} \mid \exists u \in \mathbb{U}, \ x \in D_i, \ A_i x + B_i u + f_i \in \mathcal{X}^0 \right\} = \bigcup_{j \in J^0} \mathcal{F}_{1,j}.
\]

For a fixed \( i \) and \( l \), the target set \( \mathcal{X}^0_l \) is convex and the dynamics affine, such that it is possible to apply standard multi-parametric programming techniques to compute the set of states which can be driven into \( \mathcal{X}^0_l \) \cite{BMDP02}. Therefore the set \( \text{Pre}(\mathcal{X}^0) \) is a union of polytopes and can be computed by solving \( J^0 = D \cdot L^0 \) multi-parametric programs, where \( D \) denotes the number of dynamics and \( L^0 \) is the number of polytopes which define \( \mathcal{X}^0 \). Each of these multi-parametric programs will yield a controller partition \( \{ \mathcal{P}^0_{j,r} \}_{r=1}^R \) consisting of \( R \) controller regions whose union covers the feasible set \( \mathcal{F}_{1,j} \) (see Definition 8.1). Since the set \( \text{Pre}(\mathcal{X}^0) \) is computed via multi-parametric programming, we also obtain an associated feedback law \( u(x) \) which provides feasible inputs as a function of the state (see Theorem 8.1). Note that the various controller partitions may overlap, but that each controller will drive the state into \( \mathcal{X}^0 \) in one time step, i.e. \( f_{\text{PWA}}(x, u(x)) \in \mathcal{X}^0 \). Henceforth, we will use the notation \( \mathcal{X}^{\text{iter}+1} = \text{Pre}(\mathcal{X}^{\text{iter}}) = \bigcup_{j \in J^{\text{iter}+1}} \mathcal{X}^{\text{iter}+1}_j \).

In the following, the Algorithm for computing the minimum time controller for PWA systems will be introduced.

**Algorithm 8.2.** Minimum Time Controller Computation

1. Compute the invariant set \( \mathcal{O}_{\infty}^{\text{PWA}} \) around the origin (see Figure 8.1(a)) as well as the associated Lyapunov function \( V(x) = x'Px \) and feedback laws \( F_i \) as described by Algorithm 8.1. Note that the various controller partitions may overlap, but that each controller will drive the state into \( \mathcal{X}^0 \) in one time step, i.e. \( f_{\text{PWA}}(x, u(x)) \in \mathcal{X}^0 \). Henceforth, we will use the notation \( \mathcal{X}^{\text{iter}+1} = \text{Pre}(\mathcal{X}^{\text{iter}}) = \bigcup_{j \in J^{\text{iter}+1}} \mathcal{X}^{\text{iter}+1}_j \).

2. Initialize the set list \( \mathcal{X}^0 = \mathcal{O}_{\infty}^{\text{PWA}} \) and initialize the iteration counter \( \text{iter} = 0 \).

3. Compute \( \mathcal{X}^{\text{iter}+1} = \text{Pre}(\mathcal{X}^{\text{iter}}) = \bigcup_{j \in J^{\text{iter}+1}} \mathcal{X}^{\text{iter}+1}_j \), by solving a sequence of multi-parametric programs (see Figure 8.1(b)). Thus, a feedback controller partition \( \{ \mathcal{P}^{\text{iter}+1}_{j,r} \}_{r=1}^R \) is associated with each obtained set \( \mathcal{X}^{\text{iter}+1}_j \). Obviously, the number of regions \( R \) of each partition is a function of \( \text{iter} \) and \( j \).

4. For all \( j^* \in J^{\text{iter}+1} \). If \( \mathcal{X}^{\text{iter}+1}_{j^*} \subseteq \left( \bigcup_{j \in J^{\text{iter}+1} \setminus \{j^*\}} \mathcal{X}^{\text{iter}+1}_j \right) \cup \left\{ \bigcup_{i \in \{1, \ldots, \text{iter} \}} \mathcal{X}^i \right\} \), then discard \( \mathcal{X}^{\text{iter}+1}_{j^*} \) from \( \mathcal{X}^{\text{iter}+1} \) and set \( J^{\text{iter}+1} = J^{\text{iter}+1} \setminus \{j^*\} \) (see Figures 8.1(c) and 8.1(d)).

5. If \( \mathcal{X}^{\text{iter}+1} \neq \emptyset \), set \( \text{iter} = \text{iter} + 1 \) and goto step 3.
6. For all \( k \in \{1, \ldots, \text{iter} - 1\} \) and \( r \in \mathbb{N}^+ \) discard all controller regions \( P^{k+1}_{j,r} \) for which \( P^{k+1}_{j,r} \subseteq \bigcup_{i \in \{1, \ldots, k\}} X^i \) since the associated control laws are not time-optimal and will never be applied.

![Figure 8.1: Description of Algorithm 8.2.](image)

The index \( \text{iter} \) corresponds to the number of steps in which a state trajectory will enter the terminal set \( O^{\text{PWA}}_\infty \) if a receding horizon control policy is applied. If the algorithm terminates in finite time, then the union of all controlled sets \( X^{\text{iter}} \) is the maximum controllable set \( K^{\text{PWA}}_\infty (O^{\text{PWA}}_\infty) \) as given in Definition 8.2.

**Remark 8.4.** Note that Algorithm 8.2 may not terminate in finite time (e.g., if states are unbounded). This is a problem inherent property and not a result of the computation scheme. It is therefore advisable to specify a maximum step distance \( N \) which can be used as a termination criterion in step 5 of Algorithm 8.2. The resulting controller computation will then terminate in finite time and the feedback controller will cover \( K^{\text{PWA}}_N (O^{\text{PWA}}_\infty) \).

**Remark 8.5.** The implementation of Algorithm 8.2 requires a function that can detect if a convex polyhedron \( P_0 \) is covered by a finite set of non-empty convex polyhedra \( \{P_r\}_{r=1}^R \), i.e., if \( P_0 \subseteq \bigcup_{r \in \{1, \ldots, R\}} P_r \). For instance, this operation is needed to check if two unions of polyhedra cover the same non-convex set [RGK+04] (e.g., Step 5 of Algorithm 8.2). Due to space constraints, we refer the reader to [BT03], where an efficient algorithm is given to perform this task.

**Minimum Time Controller: On-Line Application**

In the minimum time algorithm presented in this chapter, we can take advantage of some of the algorithm features to speed up the on-line region identification procedure. We propose a three-tiered search tree structure which serves to significantly speed up the region identification. Unlike the search tree proposed in [TJB03], the tree structure proposed here is computed automatically by Algorithm 8.2, i.e., no post-processing is necessary. The three levels of the search tree are as follows:

**Algorithm 8.3.** On-Line Application of Minimum Time Controller
1. Identify the active dynamics $i$, such that $x \in D_i$, $i \in I$ (see Figure 8.2(a)).

2. Identify controller set $X_j^{\text{iter}}$ associated with dynamic $i$ which is ‘closest’ to the target set $X^0$, i.e. $\min_{\text{iter}, j} \text{iter}$, s.t. $x \in X_j^{\text{iter}}$, $j \in J^{\text{iter}}$ (see Figure 8.2(b)).

3. Extract the controller partition $\{P_{j,r}^{\text{iter}}\}_{r=1}^R$ with the corresponding feedback laws $F_r, G_r$ and identify the region $r$ which contains the state $x \in P_{j,r}^{\text{iter}}$ (see Figure 8.2(c)).

4. Apply the control input $u = F_r x + G_r$. Goto 1.

Note that the association of controller partitions $X_j^{\text{iter}}$ to active dynamics in step 2 is trivially implemented by building an appropriate lookup-table during the off-line computation in Algorithm 8.2.

Theorem 8.3. The controller obtained with Algorithm 8.2 and applied to a PWA system (8.4) in a receding horizon control fashion according to Algorithm 8.3, guarantees asymptotic stability and feasibility of the closed loop system, provided $x(0) \in K_{\text{PWA}}^P(O_{\text{PWA}}^\infty)$.

Proof Assume the initial state $x(0)$ is contained in the set $X^{\text{iter}}$ with a step distance to $O_{\text{PWA}}^\infty$ of $\text{iter}$. The control law at step 4 of Algorithm 8.3 will drive the state into a set $X^{\text{iter}-1}$ in one time step (see step 3 of Algorithm 8.2). Therefore, the state will enter $O_{\text{PWA}}^\infty$ in $\text{iter}$ steps. Once the state enters $O_{\text{PWA}}^\infty$ the feedback controllers associated with the common quadratic Lyapunov ensure stability.

The proof stretches the classic definition of stability, since the Lyapunov function is discontinuous and assumes only discrete values for $x \notin O_{\text{PWA}}^\infty$. However, this is not a problem since Lyapunov functions do not need to be continuous for discrete-time systems.

8.4.2 Controller with Reduced Number of Switches

In general, it is possible to obtain even simpler controllers and faster computation times by modifying Algorithm 8.2. Instead of computing a minimum time controller, an alternative
scheme which aims at reducing the number of switches can be applied. A change in the active system dynamic $D_i \rightarrow D_j$, ($i \neq j$) is referred to as a switch. The proposed procedure does not guarantee the minimum number of switches, though straightforward modifications to the algorithm would yield such a solution. The “minimum number of switches” solution was not pursued in this chapter since computation time was the primary objective.

The proposed reduced switch controller will avoid switching the active dynamics for as long as possible. We will introduce the following operator

$$Pre_i(X^\text{iter}_j) = \{x \in X \mid \exists u \in U, x \in D_i, A_i x + B_i u + f_i \in X^\text{iter}_j\}, \quad i \in I.$$ 

Once the $j$ - $th$ controller set $X^\text{iter}_j$ obtained at iteration $\text{iter}$ is computed (see Algorithm 8.2, step 3) for dynamics $i$, the set is subsequently used as a target set for as long as the controllable set of states can be enlarged without switching the active dynamics $i$. With this scheme, the total number of convex sets needed to describe the controlled set $X^\text{iter}$ remains constant while the size of $X^\text{iter}$ increases. Therefore, this scheme generally results in significantly fewer sets during the dynamic programming iterations compared to Algorithm 8.2.

Specifically, the algorithm works as follows:

**Algorithm 8.4.** Computation: Controller with Reduced Number of Switches

1. Compute the invariant set $O^\text{PWA}_\infty$ around the origin (see Figure 8.1(a)) as well as the associated Lyapunov function $V(x) = x'Px$ and linear feedback laws $F_r$ as described by Algorithm 8.1.

2. Initialize the set list $X^0 = O^\text{PWA}_\infty = \bigcup_{j \in J^0} X^0_j$ and initialize the iteration counter $\text{iter} = 0$.

3. Initialize $X^\text{iter}_j = \emptyset$ and execute the following for all $i \in I$ and $j \in J^\text{iter}$:
   
   a) Initialize counter $c = 0$ and set $C^0 = X^\text{iter}_j$.
   
   b) Compute $C^{c+1} = Pre_i(C^c)$ by using multi-parametric programming and store the associated controller partition. Thus, a feedback controller partition $\{P^c_{j,r,\text{iter}}\}_{r=1}^R$ is obtained.
   
   c) If $C^c \subset C^{c+1}$, set $c = c + 1$ and goto step 3b.
   
   d) Set $X^{\text{iter}+1}_j = X^{\text{iter}+1}_j \cup C^c$ (see Figure 8.3).

4. If $X^{\text{iter}+1} \neq X^{\text{iter}}$, set $\text{iter} = \text{iter} + 1$ and goto 3.

5. For all $k \in \{1, \ldots, \text{iter} - 1\}$, $c \in \mathbb{N}$ and $r \in \mathbb{N}^+$ discard all controller regions $P^{c,k+1}_{j,r}$ for which $P^{c,k+1}_{j,r} \subset \bigcup_{i \in \{1, \ldots, k\}} X^i$ since the associated control law has a non-minimum number of switches and will never be applied.

The on-line computation is identical to the scheme described in Section 8.4.1 and the same finite time termination conditions as in Remark 8.4 apply.
Remark 8.6. In Algorithm 8.4 the counter \( \text{iter} \) associated to the control sets \( \mathcal{X}^{\text{iter}} \) corresponds to the number of dynamic switches which will occur before the target set \( \mathcal{O}_{\infty}^{\text{PWA}} \) is reached.

Remark 8.7. If we always have \( \mathcal{C}^c \not\subset \mathcal{C}^{c+1} \) in step 3c of Algorithm 8.4, then Algorithm 8.4 is identical to Algorithm 8.2. However if \( \mathcal{C}^c \subset \mathcal{C}^{c+1} \), it is possible to perform a large part of the computations on convex sets, which makes Algorithm 8.4 more efficient than Algorithm 8.2, in general.

Theorem 8.4. A controller computed according to Algorithm 8.4 and applied to a PWA system (8.4) according to Algorithm 8.3, guarantees stability and feasibility of the closed loop system, provided \( x(0) \in \mathcal{K}_{\infty}^{\text{PWA}}(\mathcal{O}_{\infty}^{\text{PWA}}) \).

Proof Follows from Theorem 8.3. \( \square \)

8.4.3 One-Step Controller

In the previous sections, stability was guaranteed by imposing an appropriate terminal set constraint. In order to cover large parts of the state space, this type of constraint generally entails the use of large prediction horizons which results in controllers with a large number of regions.

In this section, instead of enforcing asymptotic stability with an appropriate terminal set (and the associated cost), we propose to enforce constraint satisfaction only. This can be easily achieved by imposing a set-constraint on the first predicted state in the MPC formulation. Hence, the terminal-set constraint (8.5b) becomes superfluous and we do not need to rely on large prediction horizons. Asymptotic stability is analyzed in a second step. This scheme is inspired by promising complexity reduction results for LTI systems in [GPM03, GM03].

Constraint Satisfaction

If (8.5) is solved via multi-parametric programming for any prediction horizon \( N' \) with set constraints \( x_1 \in \mathcal{K}_{N'}^{\text{PWA}}(\mathcal{O}_{\infty}^{\text{PWA}}) \) and \( x_{N'} \in \mathcal{T}_{\text{set}} = \mathbb{R}^n \), the resulting MPC controller will...
guarantee that the state will remain within $\mathcal{K}^{\text{PWA}}_N(\mathcal{O}^{\text{PWA}}_\infty)$ for all time. The set $\mathcal{O}^{\text{PWA}}_\infty$ is obtained by applying Algorithm 8.1. The set constraint on the first step guarantees that the resulting controller partition will be positive invariant, which directly implies feasibility for all time [Bla99, Ker00].

Note that this allows us to control large volume sets $\mathcal{K}^{\text{PWA}}_N(\mathcal{O}^{\text{PWA}}_\infty)$ with short prediction horizons $N'$, i.e. $N' \ll N$. We will henceforth assume $N' = 1$, $N \to \infty$ and refer to the proposed controller as one-step controller. Note that in the examples provided in Section 8.5.1, the set $\mathcal{K}^{\text{PWA}}_\infty(\mathcal{O}^{\text{PWA}}_\infty)$ was always finitely determined. This is not always the case such that in practice it is advisable to limit $N$ to be a large but finite value.

Since the target set $\mathcal{K}^{\text{PWA}}_\infty(\mathcal{O}^{\text{PWA}}_\infty) = \bigcup_{c \in \{1, \ldots, C\}} \mathcal{K}^c_\infty$ is non-convex in general (i.e. a union of $C$ polytopes $\mathcal{K}^c_\infty$) a controller partition can be obtained by solving a sequence of $C \cdot D$ multi-parametric programs (8.3), where $D$ corresponds to the total number of different dynamics. Specifically, the one-step controller can be obtained by solving $C \cdot D$ problems (8.3) for $N' = 1$ with $x_1 \in T_{\text{set}} = \mathcal{K}^c_\infty$ in (8.3c) ($C$ different sets) and for $D$ different dynamics in (8.3d).

### Stability Analysis

The controller partition obtained in Section 8.4.3 will generally contain overlaps such that the closed-loop dynamics associated with a given state $x(0)$ may not be unique. It is therefore not possible to perform a non-conservative stability analysis of the closed-loop system. However, by using the PWA value function $J_N^*(x)$ in (8.5a) as a selection criterion it is possible to obtain a non-overlapping partition ( [GKBM03] or [Bor03], pg. 158-160) by solving a number of LPs, i.e. only the cost optimal controller is stored.

The resulting controller partition is invariant and a unique controller region $r$ ($x \in P_r$, $u = F_rx + G_r$) and unique dynamics $l$ ($x \in D_l$) is associated with each state $x$, i.e. the closed loop system corresponds to an autonomous PWA system

$$x_{k+1} = (A_l + B_lF_r)x_k + B_lG_r + f_l, \quad x_k \in P_r \cap D_l \quad (8.7a)$$
$$= \tilde{A}_r x_k + \tilde{f}_r, \quad x_k \in P_r. \quad (8.7b)$$

Since every controller region $P_r$ is only contained in one unique dynamic $D_l$, the update matrix $\tilde{A}_r$ and vector $\tilde{f}_r$ are uniquely defined. It will be shown in the following how to formulate the search for a PWA Lyapunov function for autonomous PWA systems as a linear program (LP).

It was shown how to use Semi-Definite Programming (SDP) to compute piecewise quadratic (PWQ) Lyapunov functions for continuous-time PWA systems in [JR98] and for discrete-time PWA systems in [FTCMM02, GLPM03]. The search for a PWQ Lyapunov function is conservative, since the SDP formulation is based on the $S$-procedure, which is not lossless for the cases considered [BGFB94]. Therefore, instead of searching for a PWQ Lyapunov function via SDP, we here show how to compute a PWA Lyapunov function via LP. The proposed scheme is based on results for continuous time systems which were published in [Joh03].

The computation scheme for the PWA Lyapunov function is non-conservative (i.e. if a PWA Lyapunov function exists for the given partition, it will be found) such that it may
succeed when no PWQ Lyapunov function can be found with the schemes in [FTCMM02, GLPM03]. However, the converse is also true. Furthermore, the value function associated with a mp-LP controller partition is PWA, such that this function type is a natural candidate in the search for a Lyapunov function. The following Theorem is based on [Vid93, p. 267]:

**Theorem 8.5 (Asymptotic Stability).** The origin $x = 0$ is asymptotically stable if there exists a function $V(x)$ and scalar coefficients $\alpha > 0$, $\beta > 0$, $\rho > 0$ such that: $\beta \| x_k \| \geq V(x_k)$ and $V(x_{k+1}) - V(x_k) \leq -\rho \| x_k \|$, $\forall x_k \in X$ and $V(x) = \infty$, $\forall x \notin X$. The successor state $x_{k+1}$ is defined in (8.7b), $\| \cdot \|$ denotes a vector norm and $X$ denotes the state space of interest.

In order to pose the problem of finding a PWA Lyapunov function without introducing conservative relaxations, a region transition map is created. Specifically, a transition map $\mathcal{S}$ is created according to

$$\mathcal{S}(i, j) = \begin{cases} 1, & \text{if } \exists x_k \in \text{int}(P_i), \text{ s.t. } x_{k+1} \in P_j, \\ 0, & \text{otherwise.} \end{cases}$$

where $x_{k+1}$ is defined by (8.4) and Theorem 8.1 and int(·) denotes the strict interior of a set.

**Remark 8.8.** In principle, one LP needs to be solved for each element of the transition map $\mathcal{S}$, i.e. a total of $R^2$ LPs, where $R$ denotes the total number of system dynamics. However, instead of solving LPs directly, it is advisable to first compute bounding boxes (hyper-rectangles) for each region $P_r$ ($r \in R$). In addition, a bounding box of the affine map of the region $P^+_r = \{ \tilde{A}_r x + f_r \in \mathbb{R}^n \mid x \in P_r \}$ needs to be computed. The number of LPs which need to be solved in order to compute the bounding boxes is linear in the number of regions $R$ and state space dimension $n$. This computation is tractable even for very complex partitions. The bounding boxes can be efficiently checked for intersections, such that certain transitions $i \rightarrow j$ can be ruled out. In our experience, the bounding box implementation is the most effective way to compute $T$ for complex region partitions.

In a second step, the polytopic transition sets $P_{ij}$ for system (8.7b) are explicitly computed as

$$P_{ij} = \{ x_k \in \mathbb{R}^n \mid x_k \in P_i, \ x_{k+1} \in P_j \}, \ \forall i, j \in \{1, \ldots, R\}, \text{ s.t. } \mathcal{S}(i, j) = 1.$$ 

If $\mathcal{S}(i, j) = 0$, we can directly set $P_{ij} = \emptyset$. Subsequently, the vertices of the transition sets (vert$(P_{ij})$) and the controller sets (vert$(P_i)$) are computed. The problem of finding a PWA Lyapunov function,

$$\text{PWA}_i(x) = L_i x + C_i, \quad \text{if } x \in P_i, \ i = 1, \ldots, R$$

for the autonomous PWA system (8.7b) such that the conditions in Theorem 8.5 are satisfied can now be stated as

$$\beta \| x \|_1 \geq \text{PWA}_i(x) \geq \alpha \| x \|_1, \ \alpha, \beta > 0, \ \forall x \in \text{vert}(P_i), \ \forall i \in \{1, \ldots, R\}, \ (8.8a)$$

$$\text{PWA}_j(x_{k+1}) - \text{PWA}_i(x_k) \leq \rho \| x_k \|_1, \ \rho < 0, \ \forall x_k \in \text{vert}(P_{ij}), \ \forall i, j \in \{1, \ldots, R\}. \ (8.8b)$$

Since the vertices of all sets $P_i$ and $P_{ij}$ are known, the resulting problem is linear in $L_i, C_i, \alpha, \beta, \rho$ and can therefore be solved as an LP.
Theorem 8.6. If the LP (8.8) associated with the autonomous PWA system (8.7b) is feasible, then this system is asymptotically stable.

Proof Since the function $PWA_i(x)$ is piecewise affine, it follows that satisfaction of (8.8a) for all vertices of $P_i$ implies that the inequalities in (8.8a) will also hold $\forall x \in P_i$. Furthermore, if (8.8b) holds for all vertices of $P_{ij}$, it follows from linearity of the system dynamics (8.7b) that the inequality will hold for all states $x \in P_{ij}$. Since the partition $\mathcal{F}_N$ is invariant, it follows that $\mathcal{F}_N = \bigcup_{i \in \{1, \ldots, R\}} P_i = \bigcup_{i,j \in \{1, \ldots, R\}} P_{ij}$. Therefore, the inequalities in (8.8a) and (8.8b) hold $\forall x \in \mathcal{F}_N$ such that the conditions in Theorem 8.5 are satisfied, i.e. feasibility of (8.8) implies asymptotic stability of the autonomous PWA system (8.7b).

It should be noted that the required computation time may become large because of the extensive reachability analysis, vertex enumeration and size of the final LP. Specifically, the LP (8.8) introduces one constraint for each vertex of each region $P_r$, $\forall r \in \{1, \ldots, R\}$ (see (8.8a)) and one constraint for each vertex of each $P_{ij}$, $\forall i, j \in \{1, \ldots, R\}$ (see (8.8b)). The number of variables is $(n + 1)R$, where $R$ denotes the number of regions and $n$ the state space dimension. However, in the authors' experience, stability analysis problems for a couple of hundred regions in a state space dimension of less than five are tractable and the necessary computation effort is comparable to the approaches in [FTCMM02, GLPM03].

One-Step Controller Computation

The one-step control scheme utilizes tools from invariant set computation and stability analysis in order to compute controllers with small prediction horizons which guarantee constraint satisfaction as well as asymptotic stability. The basic procedure consists of two main stages. In the first stage, a one-step optimal controller is computed which guarantees constraint satisfaction for all time (Section 8.4.3). Since constraint satisfaction does not imply asymptotic stability, it is necessary to analyze the stability properties of the closed-loop system in a second stage (Section 8.4.3). Specifically, the algorithm works as follows.

Algorithm 8.5. Computation: One-Step Controller

1. Compute the invariant set $\mathcal{O}^{PWA}_\infty$ around the origin and an associated Lyapunov function as described by Algorithm 8.1.

2. Compute the set $\mathcal{K}^{PWA}_N(\mathcal{O}^{PWA}_\infty) = \bigcup_{c \in \{1, \ldots, C\}} \mathcal{K}^{c}_{N}(N \to \infty)$ by applying Algorithm 8.2.

3. Solve a sequence of $C \cdot D$ mp-LPs (8.3) for prediction horizon $N' = 1$ with $\mathcal{T}_{set} = \mathcal{K}^{c}_{N'}$, $\forall c \in \{1, \ldots, C\}$ in (8.3c) and affine dynamics $d \in \{1, \ldots, D\}$ in (8.3d).

4. Remove the region overlaps by using the PWA value function as a selection criterion (see [BT03, Bor03] for details).
5. Attempt to find a PWA Lyapunov function by solving the LP (8.8) or attempt to find a PWQ Lyapunov function as described in [GLPM03].

There is no guarantee that step 2 of Algorithm 8.5 will terminate in finite time or that a Lyapunov function can be found in step 5. The finite time termination conditions are discussed in Remark 8.4. If no Lyapunov function is found, the resulting controller is still guaranteed to satisfy the system constraints for all time, but no proof of asymptotic stability can be given. Note that this does not imply that the closed-loop system is unstable, it merely shows that no PWA Lyapunov function exists for the given partition.

**Remark 8.9.** If the stability analysis in Step 5 of Algorithm 8.5 fails, it is advisable to recompute the controller in Step 3 using different weights $R, Q, Q_f$ and/or a different prediction horizon $N'$ in (8.3). Slight modifications in these parameters may make the subsequent stability analysis in Step 5 feasible. We have observed that large weights on the states (i.e. $Q, Q_f$ large) and a larger prediction horizon $N'$ have a positive effect on the likelihood of success.

### 8.5 Numerical Examples

#### 8.5.1 Controller Computation

As was shown in [GM03,GPM03] and will also be illustrated in this section, algorithms of type 8.2 – 8.5 generally yield controllers of significantly lower complexity than those which are obtained if a linear norm-objective is minimized as in (8.5) [BCM03b,BCM03a].

**Example 8.1.** Consider the 2-dimensional problem adopted from [MR03]:

\[
x(k + 1) = \begin{cases}
1 & 0.2 \\
0 & 1
\end{cases} + \begin{cases}
0 & 1 \\
0 & 0
\end{cases} u(k) + \begin{cases}
0 & 0 \\
0 & 0
\end{cases} \quad \text{if } x_1(k) \leq 1 \\
0.5 & 0.2 \\
0 & 1
\end{cases} + \begin{cases}
0 & 1 \\
0 & 0
\end{cases} u(k) + \begin{cases}
0.5 & 0 \\
0 & 0
\end{cases} \quad \text{if } x_1(k) \geq 1
\]

(8.9)

subject to constraints $-x_1(k) + x_2(k) \leq 15$, $-3x_1(k) - x_2(k) \leq 25$, $0.2x_1(k) + x_2(k) \leq 9$, $x_1(k) \geq -6$, $x_1(k) \leq 8$, and $-1 \leq u(k) \leq 1$. Weight matrices in the cost function were chosen as $Q = I$ and $R = 0.1$ in (8.5).

**Example 8.2.** Consider the 3-dimensional PWA system introduced in [MR03]:

\[
x(k + 1) = \begin{cases}
1 & 0.5 & 0.3 \\
0 & 1 & 1 \\
0 & 0 & 1
\end{cases} + \begin{cases}
0 & 0 \\
0 & 0 \\
0 & 0
\end{cases} u(k) + \begin{cases}
0 & 0 \\
0 & 0 \\
0 & 0
\end{cases} \quad \text{if } x_2(k) \leq 1 \\
1 & 0.2 & 0.3 \\
0 & 0.5 & 1 \\
0 & 0 & 1
\end{cases} + \begin{cases}
0 & 0 \\
0 & 0 \\
0 & 0
\end{cases} u(k) + \begin{cases}
0 & 0 \\
0 & 0 \\
0.5 & 0
\end{cases} \quad \text{if } x_2(k) \geq 1
\]

(8.10)

Subject to constraints $-10 \leq x_1(k) \leq 10$, $-5 \leq x_2(k) \leq 5$, $-10 \leq x_3(k) \leq 10$, and $-1 \leq u(k) \leq 1$. Again, weights in the cost function are $Q = I$, $R = 0.1$. 


Example 8.3. Consider the 4-dimensional PWA system introduced in [MR03]:

\[
x(k + 1) = \begin{cases} 
    \begin{bmatrix} 
    1 & 0.5 & 0.3 & 0.5 \\
    0 & 1 & 1 & 1 \\
    0 & 0 & 1 & 1 \\
    1 & 0.2 & 0.3 & 0.5 \\
    0 & 0.5 & 1 & 1 \\
    0 & 0 & 1 & 1 \\
    0 & 0 & 0 & 1 \\
    \end{bmatrix} x(k) \\
    + \begin{bmatrix} 
    0 \\
    0 \\
    1 \\
    0 \\
    0 \\
    0 \\
    0 \\
    \end{bmatrix} u(k) \\
    + \begin{bmatrix} 
    0 \\
    0 \\
    0.3 \\
    0.5 \\
    1 \\
    0 \\
    0 \\
    \end{bmatrix} & \text{if } x_2(k) \leq 1 \\
    \begin{bmatrix} 
    1 & 0 & 1 & 1 \\
    0 & 0 & 0 & 1 \\
    \end{bmatrix} x(k) \\
    + \begin{bmatrix} 
    0 \\
    0 \\
    0 \\
    1 \\
    \end{bmatrix} u(k) \\
    + \begin{bmatrix} 
    0 \\
    0 \\
    0 \\
    0 \\
    \end{bmatrix} & \text{if } x_2(k) \geq 1 \\
\end{cases}
\]

Subject to constraints \(-10 \leq x_1(k) \leq 10, -5 \leq x_2(k) \leq 5, -10 \leq x_3(k) \leq 10, -10 \leq x_4(k) \leq 10, \) and \(-1 \leq u(k) \leq 1\). Weighting matrices in the cost function are \(Q = I, R = 0.1\).

Once the set \(O_{PW}^{\infty}\) is computed, Algorithms 8.2–8.5 are applied to Examples 8.1–8.3. A runtime comparison of the computation procedures as well as complexity of the resulting solutions are reported in Table 8.1. As can be seen in Table 8.1, the proposed Algorithms

<table>
<thead>
<tr>
<th>Algorithm 8.2</th>
<th>Algorithm 8.4</th>
<th>Algorithm 8.5</th>
<th>Algorithm [BCM03b]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 8.1</td>
<td>61 sec.</td>
<td>40 sec.</td>
<td>53 sec.</td>
</tr>
<tr>
<td>Example 8.2</td>
<td>1153 sec.</td>
<td>755 sec.</td>
<td>286 sec.</td>
</tr>
<tr>
<td>Example 8.3</td>
<td>92 hours</td>
<td>2.2 hours</td>
<td>1413</td>
</tr>
</tbody>
</table>

Table 8.1: Off-line CPU-time \(t\) and number of controller regions \(#R\) for different algorithms. The * symbol denotes that the stability analysis procedure failed. The computation was run on a 2.8GHz Pentium IV CPU running the Windows version of MATLAB 6.5 along with the NAG foundation LP solver.

did not work well for Example 8.3. Even though the proposed computation schemes are significantly more efficient than existing approaches, it is easy to come up with examples where the associated computation time is prohibitive. For Example 8.2, the controller regions are depicted in Figures 8.4(a)–8.4(c).

In order to compare low complexity control strategies discussed in this chapter with the cost optimal approach of [BCM03b] described in Chapter 7, we generated 10 random PWA systems with 2 states, 1 input and 4 piecewise-affine dynamics. All elements in the state space matrices were assigned random values between \([-2, 2]\) (i.e., stable and unstable systems were considered). Each of the random PWA systems consists of 4 different affine dynamics which are defined over non-overlapping random sets whose union covers the square \(X = [-5,5] \times [-5,5]\). The origin was chosen to be on the boundary of multiple dynamics. All simulation runs as well as the random system generation was performed with the MPT toolbox [KGBM04].

\[1\] For random PWA systems mpt_randPWAsys was called.
Algorithms 8.2, 8.4 and 8.5, as well as the cost-optimal strategy of [BCM03b] were applied to these systems. Complexity of the resulting solution and run time of each algorithm are depicted graphically in Figures 8.5(a) and 8.5(b).

Figure 8.5: Complexity and runtime for 10 random PWA systems.

To further investigate the behavior of different control strategies, another test on a set of 10 random PWA systems was performed to show how the complexity of Algorithms 8.2, 8.4 and 8.5 scales with increasing volume of the exploration space. A comparison with the approach in [BCM03b] is depicted in Figures 8.6(a) and 8.6(b). For the random systems considered here, the necessary runtime is reduced by two orders of magnitude and the solution complexity is reduced by one order of magnitude, on average. In addition, these differences become larger with increasing size of the state constraints. Although we have not come across
any examples where the proposed schemes are inferior to the approaches in [BBBM03a, KM02], we are not able to proof that no such cases exist.

Figure 8.6: Complexity and runtime versus size of exploration space (average over 10 random PWA systems).

However, none of the algorithms presented in this chapter guarantee optimal closed-loop performance in the sense of the cost-objective (8.5). In order to assess the degradation in performance, equidistantly spaced data points inside the set $\mathcal{K}^{\text{PWA}}(O^{\text{PWA}})$ were generated as feasible initial states. Subsequently, the closed-loop trajectory cost for these initial states was computed according to the performance index (8.5a). The average decrease in performance with respect to the cost-optimal solution of [BCM03b] is summarized in Figures 8.7(a) and 8.7(b). It can be seen that closed-loop performance gets better with increasing size of the exploration space. The intuitive explanation of this observation is as follows: if the state is far away from the origin, going at “full throttle” will be the optimal strategy, since the contribution of the state penalty term in (8.5a) is much higher than the term which penalizes the control action. This statement holds regardless of the objective matrices $Q$ and $R$, i.e. for any $Q$ and $R$, there exists a distance $d$ such that for any $\|x\| \geq d$ the state penalty term in (8.5a) dominates. Therefore almost the same performance is achieved with low complexity strategies as with cost-optimal algorithms for a majority of the controllable state-space, resulting in good average performance.

8.5.2 Stability Analysis

In this subsection, we will analyze the likelihood of finding a PWA or PWQ Lyapunov function for a controller partition obtained with the method proposed in Section 8.4.3. The PWA analysis scheme was implemented as described in Section 8.4.3 and the PWQ analysis is based on the scheme in [GLPM03]. Specifically, 200 random PWA systems with $n = 2$ states and $m = 1$ inputs were created. The random PWA systems were created as described in
the previous section. Subsequently, Algorithm 8.5 was applied to the random PWA systems and the resulting controller partition was analyzed for stability. The results are given in Table 8.2. It is clear from Table 8.2 that neither the PWA nor PWQ Lyapunov analysis is superior, which justifies the need to consider the LP based analysis presented in Section 8.4.3. Furthermore, the results suggest that the likelihood of finding a Lyapunov function for the one-step controller is fairly large, in general.

### 8.6 Conclusion

A scheme to compute terminal sets (along with the associated ‘cost-to-go’) for generic PWA systems was presented, which may be used in the context of receding horizon control to obtain asymptotic stability guarantees for the closed-loop system. These sets are subsequently used to derive three novel algorithms to compute low complexity feedback controllers for constrained PWA systems. All controllers guarantee constraint satisfaction for all time as well as asymptotic stability. The computation scheme iteratively solves a series of multi-parametric programs such that a feedback controller is obtained which drives the state into a target set in minimum time. An alternative controller which aims at reducing the number of

<table>
<thead>
<tr>
<th>Number of Systems</th>
<th>PWA and PWQ</th>
<th>PWA only</th>
<th>PWQ only</th>
<th>none</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>8</td>
<td>2</td>
<td>27</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: Stability analysis results for controller partitions obtained with Algorithm 8.5 for 200 random PWA systems. PWA and PWQ denote whether a PWA and/or PWQ Lyapunov function was found for the partition. The analyzed partitions consisted of 26 - 187 regions.
switches between different dynamics is also presented and the provided examples suggest that this approach may further reduce complexity. Furthermore, a search tree for efficient on-line identification of the optimal feedback law is automatically constructed by both algorithms. A third computation scheme (referred to as one-step control) is also presented, which separately deals with the requirement of constraint satisfaction and asymptotic stability. In the one-step scheme, stability is not enforced but merely verified a posteriori. While the resulting controller is of even lower complexity than the previous two, there is no a priori guarantee that the closed-loop system will be asymptotically stable. In order to analyze stability, in this chapter we introduced a method of computing PWA Lyapunov functions for a given autonomous PWA system. The proposed method is based on linear programming and is guaranteed to find a PWA Lyapunov function for a given partition, if it exists.

In an extensive case study, it is observed that all three algorithms reduce complexity versus optimal controllers [BBBM03a, KM02] by several orders of magnitude. The proposed procedures make problems tractable that were previously too complex to be tackled by the methods in [BBBM03a, KM02]. Although we have not come across any examples where the proposed schemes are inferior to the approaches in [BBBM03a, KM02], we are not able to prove that no such cases exist.

The presented algorithms are contained in the MPT toolbox [KGBM04].
9

Stability Analysis of PWA Systems with a Linear Performance Index

We consider the constrained finite time optimal control (CFTOC) problem for the class of discrete-time linear hybrid systems. For a linear performance index the solution to the CFTOC problem is a time-varying piecewise affine (PWA) function of the state. However, when a receding horizon control strategy is used stability and/or feasibility (constraint satisfaction) of the closed-loop system is not guaranteed. In this chapter we present an algorithm that by analyzing the CFTOC solution extracts regions of the state-space for which closed-loop stability and feasibility can be guaranteed. The algorithm computes the maximum positive invariant set and stability region (in the classical Lyapunov stability sense) of a piecewise affine system by combining reachability analysis with some basic polyhedral manipulation. The simplicity of the overall computation stems from the fact that in all steps of the algorithm only linear programs need to be solved.

9.1 Introduction

In the last few years several different techniques have been developed for the analysis and controller synthesis for hybrid systems [Son81, BZ00, BBM00c, LTS99, Bor03]. A significant amount of the research in this field has focused on solving constrained optimal control problems, both for continuous-time and discrete-time hybrid systems.

We consider the class of discrete-time linear hybrid systems, in particular, the class of constrained piecewise affine (PWA) systems that are obtained by partitioning the extended state+input space into polyhedral regions and associating with each region a different affine state update equation, cf. [Son81, HDB01]. For such a class of systems the constrained finite time optimal control (CFTOC) problem can be solved by means of multi-parametric programming [Bor03]. The solution is a piecewise affine state feedback control law and can be computed by using multi-parametric mixed-integer quadratic programming for a quadratic performance index and multi-parametric mixed-integer linear programming for a linear performance index, cf. [Bor03, DP00].

As recently shown in [BBBM03a] for a quadratic performance index and in [KM02, BCM03a] for a linear performance index, it is possible to solve the CFTOC problem without the use of integer programming. The authors propose algorithms based on a dynamic
programming strategy combined with multi-parametric quadratic or linear program solvers, depending on the performance index being used.

However, when a receding horizon control strategy is employed stability and feasibility (constraint satisfaction) of the closed-loop system are not guaranteed. To remedy this deficiency various schemes have been proposed in the literature. For constrained linear systems stability can be (artificially) enforced by introducing the proper terminal set constraints and/or terminal cost to the formulation of the CFTOC problem [MRRS00]. For the class of constrained PWA systems very few and restrictive stability criteria are known, e.g. [BM99,MRRS00]. Only recently ideas used for enforcing closed-loop stability of the CFTOC problem for constrained linear systems have been extended to PWA systems [GBTM03]. Unfortunately the technique presented in [GBTM03] introduces a certain level of sub-optimality in the solution.

Another way to guarantee closed-loop stability for the whole feasible state-space is to attain a solution to the Hamilton-Jacobi-Bellman equation. A technique to obtain such a solution, i.e., to solve the constrained infinite time optimal control (CITOC) problem with linear performance index for constrained PWA systems was recently presented in [BCM03b].

In this chapter we focus on the a posteriori analysis of the CFTOC solution in order to extract the regions of the state-space where closed-loop stability and feasibility can be guaranteed. We present a technique to compute the maximum positive invariant set and a Lyapunov stability region (based on the linear cost function) for constrained piecewise affine systems. The algorithm combines a reachability analysis with some basic polyhedral manipulations. In the end we illustrate the applicability of the proposed algorithm with several numerical examples.

### 9.2 Constrained Finite Time Optimal Control

Consider the class of linear discrete-time hybrid systems that can be described as constrained piecewise affine systems of the following form

\[
x(t + 1) = f_{\text{PWA}}(x(t), u(t)) = A_i x(t) + B_i u(t) + f_i, \quad \text{if} \ [x(t) \ u(t)] \in D_i
\]  

(9.1)

where \( D_i := \{ [x \ u] \mid D_i^x x + D_i^u u \leq D_i^0 \}, \ t \geq 0, \ x \in \mathbb{R}^n \) is the state, \( u \in \mathbb{R}^m \) is the control input, and \( \{ D_i \}_{i=1}^n \) is the polyhedral partition of the sets of the extended state+input space \( \mathbb{R}^{n+m} \). Furthermore let the union of the polyhedral partitions be \( \mathcal{D} := \cup_{i=1}^n D_i \). Note that linear state and input constraints of the general form \( C^x x + C^u u \leq C^0 \) can be incorporated in the description of \( \mathcal{D}_i \).

Throughout this chapter we will consider the cost that penalizes the deviation of the state and control action from the origin in the extended state+input space. However, all presented results also hold for any non-zero equilibrium point since such problems are easily translated to the “steer-to-the-origin” problem by a simple substitution of the variables. Hence we
define the following linear cost function
\[ J(U_T, x(0)) := \|P x(T)\|_p + \sum_{t=0}^{T-1} \|Q x(t)\|_p + \|R u(t)\|_p \]  
(9.2)
and consider the constrained finite time optimal control (CFTOC) problem
\[ J^*(x(0)) := \min_{U_T} J(U_T, x(0)), \]  
(9.3)
subj. to
\[ \begin{align*}
x(t+1) &= f_{\text{PWA}}(x(t), u(t)), \\
x(T) &\in \mathcal{X}^f 
\end{align*} \]  
(9.4)
where the column vector \( U_T := [u(0)', \ldots, u(T-1)']' \in \mathbb{R}^{mT} \) is the optimization vector, \( T \) is the time horizon, \( \mathcal{X}^f \) is the terminal target region and \( \|Q x\|_p \) with \( p \in \{1, \infty\} \) in (9.2) denotes the corresponding standard vector 1-norm or \( \infty \)-norm. Note that although the problem (9.1)–(9.4) can be posed for any choice of the matrices \( P, R, \) and \( Q \), from a practical point of view (if we want to steer the state to the origin) only the choice of \( R \) and \( Q \) being of full column rank makes sense.

We summarize the main result of the solution to the CFTOC problem (9.1)–(9.4) which is proved in [May01, Bor03].

**Theorem 9.1** (Solution to CFTOC). The solution to the optimal control problem (9.1)–(9.4) with \( p \in \{1, \infty\} \) is a time-varying piecewise affine state feedback control law of the form
\[ u^*(x(t)) = F^*_t x(t) + G^*_t, \quad \text{if} \quad x(t) \in \mathcal{P}_t^i \]  
(9.5)
and the value function is a time-varying piecewise affine function of the state
\[ J^*_t(x(t)) = \Phi^*_t x(t) + \Gamma^*_t, \quad \text{if} \quad x(t) \in \mathcal{P}_t^i \]  
(9.6)
where \( \mathcal{P}_t^i = \{ x \in \mathbb{R}^n \mid P_i^{x,t} x \leq P_i^{0,t} \} \), \( i = 1, \ldots, N^t \), is a polyhedral partition of the set \( \mathcal{X}^t \) of feasible states \( x(t) \) at time \( t \) with \( t = 0, \ldots, T-1 \).

As shown by the authors in [BCM03a] the CFTOC problem (9.1)–(9.4) can be solved in an efficient way by solving an equivalent *dynamic program* (DP) backwards in time. The DP has the following form
\[ J_t^*(x(t)) := \min_{u(t)} \|Q x(t)\|_p + \|R u(t)\|_p + J_{t+1}^*(f_{\text{PWA}}(x(t), u(t))), \]  
(9.7)
subj. to
\[ f_{\text{PWA}}(x(t), u(t)) \in \mathcal{X}^{t+1} \]  
(9.8)
for \( t = T-1, \ldots, 0 \), with boundary conditions
\[ \mathcal{X}^T = \mathcal{X}^f, \quad \text{and} \quad J_T^*(x(T)) = \|P x(T)\|_p, \]  
(9.9)
where

$$\mathcal{X}^t := \{ x \in \mathbb{R}^n \mid \exists u, f_{\text{PWA}}(x, u) \in \mathcal{X}^{t+1}\}$$

(9.10)

is the set of all states for which the problem (9.7)–(9.8) is feasible.

Since $p \in \{1, \infty\}$ the dynamic programming problem (9.7)–(9.8) can be reformulated as a multi-parametric linear program, cf. [Bor03], where the state $x(t)$ is treated as a parameter and the control input $u(t)$ as an optimization variable. By solving such a program at each iteration step $t$ we obtain the PWA optimal control law (9.5) and the PWA value function (9.6) that represents the so called ‘cost-to-go’.

In the case that the receding horizon (RH) policy is used the control is given as a time-invariant state feedback control law of the form

$$u_{\text{RH}}(x(t)) = F^0_i x(t) + G^0_i, \quad \text{if} \quad x(t) \in \mathcal{P}^0_i$$

(9.11)

and a time-invariant cost function\(^1\)

$$J_{\text{RH}}(x(t)) = \Phi^0_i x(t) + \Gamma^0_i, \quad \text{if} \quad x(t) \in \mathcal{P}^0_i$$

(9.12)

for $t \geq 0$ and thus only $N := N^0$ (possibly different) control laws have to be stored.

To simplify the notation in the rest of the chapter we will discard the superscript $0$ for the matrices and sets of region $i = 1, \ldots, N$ in the receding horizon solution, i.e. $F_i := F^0_i$, $G_i := G^0_i$, $\mathcal{P}_i := \mathcal{P}^0_i$, $\Phi_i := \Phi^0_i$, and $\Gamma_i := \Gamma^0_i$. In the following calligraphic letters always denote sets, such as e.g. $\mathcal{P} := \bigcup_{i=1}^N \mathcal{P}_i$.

### 9.3 Closed-Loop Stability & Feasibility

As mentioned in the introduction, even for linear systems the receding horizon control based on the solution to the CFTOC problem does not guarantee closed-loop stability for the whole (initial open-loop) feasible state-space $\mathcal{X}^0$, cf. [MRRS00]. Furthermore, receding horizon control might drive the state outside of $\mathcal{X}^0$. Therefore closed-loop stability and feasibility for constrained PWA systems for the whole set $\mathcal{X}^0$ cannot be guaranteed either.

Without loss of generality we may assume that

$$\forall i \in \{1, \ldots, N\}, \quad \exists! d = d(i), \quad d \in \{1, \ldots, n_d\} \quad \text{such that} \quad \forall x \in \mathcal{P}_i, \quad \begin{bmatrix} x \\ u_{\text{RH}}(x) \end{bmatrix} \in \mathcal{D}_d.$$  

(9.13)

Assumption (9.13) guarantees that the closed-loop system trajectories are uniquely defined.

The autonomous closed-loop (CL) system is then given by

$$x(t+1) = f_{\text{PWA}}^\text{CL}(x(t)) = A^\text{CL}_i x(t) + a^\text{CL}_i, \quad \text{if} \ x(t) \in \mathcal{P}_i$$

(9.14)

\(^1\)Note that $J_{\text{RH}}(\cdot)$ in (9.12) does not represent the value function of the closed-loop system when the receding horizon control law $u_{\text{RH}}(\cdot)$ is applied.
with

\[ A_i^{\text{CL}} := A_{d(i)} + B_{d(i)} F_i, \quad a_i^{\text{CL}} := f_{d(i)} + B_{d(i)} G_i, \quad i = 1, \ldots, N. \] (9.15)

**Definition 9.1** (Maximum Positive Invariant Set \( I \)). Let \( u_{\text{RH}}(\cdot) \), as in Equation (9.11), be a given control law for the PWA system (9.1). The set of states \( I \subseteq X^0 \subseteq \mathbb{R}^n \), with

\[ I := \{ x(0) \in X^0 \mid x(t + 1) = f_{\text{PWA}}^\text{CL}(x(t)) \in X^0, \forall t \geq 0 \} \] (9.16)

is called maximum positive invariant set for the closed-loop system (9.14)–(9.15).

Closed-loop feasibility for all time can be guaranteed if one can confirm that the initial state belongs to the maximum positive invariant set \( I \) [Bla99]. It is easy to see that for a given CITOCS solution (if such a solution exists)\(^2\) the maximum positive invariant set is equal to the region of closed-loop (asymptotic) stability which is in turn equal to the set \( X^0 \) [BCM03b].

Obtaining the CITOCS solution for linear or PWA systems might be computationally prohibitive due to a large (possibly infinite) prediction horizon and the complexity of the optimal solution itself. In the worst case the complexity of the problem increases exponentially with increasing prediction horizon. Just recently in [BCM03b] the authors proposed a computationally efficient algorithm to compute the CITOCS solution for constrained PWA systems with a linear performance index.

However, in many cases the numerical computation of the CITOCS solution might not be possible or might not even be desired due to the complexity of the solution. As observed in [GLPM03] for constrained linear systems with a quadratic cost function one can often neglect the difference in performance between the sub-optimal CFTOC solution with a specified minimal prediction horizon and the optimal CITOCS solution but gains a tremendous complexity reduction. This behavior is very likely also to be expected for most (if not all) constrained PWA systems with a linear performance index, confer also the example in Section 9.4.2 (Figure 9.11). It is of major importance, however, to know for which subset of the open-loop feasible region \( X^0 \) the computed sub-optimal controller can guarantee closed-loop stability and feasibility.

\(^2\)The CITOCS solution is obtained from the CFTOC problem (9.1)–(9.4) by letting \( T \to \infty \).
9.3.1 Computation of the Maximum Positive Invariant Set

In order to present our algorithm, we need the following definition:

**Definition 9.2** (Region of Attraction $\mathcal{A}$). Let $u_{RH}(\cdot)$, as in Equation (9.11), be a given control law for the PWA system (9.1). The set of states $\mathcal{A} \subseteq \mathcal{X}^0 \subseteq \mathbb{R}^n$, with

$$\mathcal{A} := \left\{ x(0) \in \mathcal{X}^0 \mid \lim_{t \to \infty} x(t) \to 0 \right\} \quad (9.17)$$

is the region of attraction (for the origin) for the closed-loop system (9.14)–(9.15). □

In the rest of the chapter we assume that the closed-loop system (9.14)–(9.15) does not exhibit chaotic behavior.

**Remark 9.1.** From the definition of the maximum positive invariant set it immediately follows that $\mathcal{A} \subseteq \mathcal{I}$. However, one can also deduce that for the bounded maximum positive invariant set $\mathcal{I} \subseteq \mathbb{R}^n$ the following holds $\mathcal{I} = \mathcal{A} \cup \{\text{limit cycles}\} \cup \{\text{stationary points } x_{\text{stat}} \neq 0 \text{ and any trajectory in } \mathcal{X}^0 \text{ leading to such points}\}$. □

Figure 9.1 shows a typical arrangement of the open-loop feasible set $\mathcal{X}^0$, the maximum positive invariant set $\mathcal{I}$, the region of attraction $\mathcal{A}$, some Lyapunov stable region $\mathcal{L}$ as well as the typical behavior for a trajectory $x(t)$ starting in these respective sets (dashed lines) for constrained PWA systems.

In [GLPM03] the authors compute the invariant set in an iterative procedure where at each iteration step a one-step reachability analysis is performed to extract the parts of the state-space that remains closed-loop feasible. The algorithm has converged when the feasible
stability analysis of PW A systems with a linear performance index

Figure 9.2: Schematical arrangement of the regions being used in Algorithm 9.1 in iteration step \([r]\). The dashed arrow denotes that the target set is reached in one step with the given control law.

state-space remains constant. Our approach to compute the maximum positive invariant set \(I\) for a given PWA state feedback control law for constrained PWA systems can be considered as complementary to the algorithm presented in [GLPM03].

We also use an iterative approach but, in contrast to [GLPM03], we focus on the computation of the parts of the open-loop feasible state-space that lead to infeasibility regions, denoted with \(U_i\). One of the benefits of our approach is that there is no need for the computation of the union of polyhedral regions in intermediate steps. The detailed procedure of the maximum positive invariant set computation is given in Algorithm 9.1. Potential speed-ups of the algorithm are not mentioned here. Note that in the following for simplicity we only mention operations on (possibly non-convex) sets and polyhedral regions but in fact to every polyhedral region a cost function and state feedback control law as in (9.12) and (9.11), respectively, are assigned.

The algorithm is divided into an initialization part and a main part. A schematical arrangement of the considered regions and sets in iteration step \([r]\) used in Algorithm 9.1 is depicted in Figure 9.2. The dashed arrow denotes the reachability from the ‘source’-set to the ‘target’-set in one step using the given PWA state feedback control law.

In the initialization part a one-step reachability analysis is performed and the possible mappings from the polyhedral region \(P_i\) to the region \(P_j\) are recorded in the mapping matrix \(M\), i.e. if parts of the region \(P_j\) can be reached from parts of region \(P_i\) in one step by the given piecewise affine control law (9.11) then the entry \(M_{i\rightarrow j}\) in the mapping matrix is set to \texttt{true}. See Figure 9.2 for a schematical explanation. \(P_{i\rightarrow j}\) denotes the part of \(P_i\) that is mapped into region \(P_j\) in one step. Additionally, regions that lead to infeasibility, denoted with \(U_k^{[0]}\), in one step are computed.

In the main part at every iteration step \([r]\), we perform a one-step reachability analysis from
the ‘feasible’ region $\mathcal{P}_i^{[r]}, i = 1, \ldots, N^r_P$, to the ‘infeasible’ regions $^3 \mathcal{U}_k^{[r-1]}, k = 1, \ldots, N_{\mathcal{U}}^{[r-1]}$, from the previous iteration step. The part of $\mathcal{P}_i^{[r]}$ that is mapped into the infeasible region $\mathcal{U}_k^{[r-1]}$ in one step is denoted with $\mathcal{P}_i^{[r]} \rightarrow \mathcal{U}_k^{[r-1]}$, cf. Figure 9.2. The union of all $\mathcal{P}_i^{[r]} \rightarrow \mathcal{U}_k^{[r-1]}$ is the new infeasible set $\mathcal{U}^{[r]}$. At the end of every iteration step $[r]$, the set difference of the feasible set $\mathcal{P}^{[r-1]}$ from the beginning of the iteration step $[r]$ and the newly computed closed-loop infeasible set $\mathcal{U}^{[r]}$ of the state-space is performed. This set-difference is the ‘new’ feasible set $\mathcal{P}^{[r]}$.

The algorithm converged when no regions of the feasible set are leading to infeasibility, i.e. when $\mathcal{U}^{[r]} = \emptyset$. The remaining feasible set is the maximum positive invariant set $\mathcal{I}$ because all states starting in this remaining set will remain in $\mathcal{I}$ for all time by construction.

Note that, by design, every region $\mathcal{P}_j^{[r]}$ is a subset of some original region $\mathcal{P}_i$. Therefore the control law and cost expressions of $\mathcal{P}_j^{[r]}$ are the same as for the original region but the region index of the subsets is changed from iteration step to iteration step. (The same applies for $\mathcal{U}_k^{[r]}$.) Thus a simple but very efficient speed-up is performed by using the matrix of the possible transitions (mappings) $M$, computed in the initialization step, and the function $\text{org}(\cdot) : X^0 \rightarrow N$, defined implicitly as

$$\text{org}(\mathcal{P}_j^{[r]}) = i \iff \mathcal{P}_j^{[r]} \subseteq \mathcal{P}_i.$$  

(9.18)

So it is easy to check in iteration step $[r]$ which regions give a possibility to lead into feasibility or infeasibility in one step, and which transitions are impossible with the computed control law, e.g. if

$$M_{\text{org}(\mathcal{P}_j^{[r]}), \text{org}(\mathcal{U}_k^{[r-1]})} = \text{false}$$

then the subset of $\mathcal{P}_j^{[r]}$ that will lead the state in one step into $\mathcal{U}_k^{[r-1]}$ is empty because it was computed in the initialization step that there is no one step transition from the original region $\mathcal{P}_i$ to $\mathcal{P}_{\text{org}(\mathcal{U}_k^{[r-1]})}$. This reduces the number of possible combinations tremendously as the iterations evolve.

From the description above it is clear that an efficient computation of the set difference has a major impact on the implementation of the algorithm. In this work we were computing the set difference with the procedure presented in [BT03], since it involves only linear programs, and the number of regions it generates for the description of the set difference is very low (see [BT03] for more details).

---

$^3$With ‘infeasible’ regions $\mathcal{U}^{[r-1]}$ we mean all the states in $X^0$ that are driven into the infeasibility set in $r$ steps.
Algorithm 9.1 (Max. Positive Invariant Set $\mathcal{I}$).

**INPUT** CFTOC solution: $f_{\text{PWA}}(x, u), u_{\text{RH}}(x), J_{\text{RH}}(x)$

**OUTPUT** Max. Positive Invariant Set $\mathcal{I}$

——— Initialization ———

\[ \mathcal{U}^{[0]} := \emptyset \]

for $i = 1$ to $N$

compute $A_i^{\text{CL}}$ and $a_i^{\text{CL}}$ according to (9.15)

for $j = 1$ to $N$

\[ \mathcal{P}_{i-j} := \left\{ x \in \mathbb{R}^n \mid \begin{bmatrix} P_i^f \cr P_i^g A_i^{\text{CL}} \end{bmatrix} x \leq \begin{bmatrix} P_i^0 \cr P_i^0 - P_i^g a_i^{\text{CL}} \end{bmatrix} \right\} \]

if \( \mathcal{P}_{i-j} \neq \emptyset \) then $M_{i-j}$ is true, else $M_{i-j}$ is false, end

if $\mathcal{P}_i \cup \bigcup_{k=1}^j \mathcal{P}_{i-k} = \emptyset$ then

$M_{i-k}$ is false for $k = j + 1, \ldots, N$, next $i$

end

\[ \mathcal{U}^{[0]} := \mathcal{U}^{[0]} \cup \left( \mathcal{P}_i \setminus \bigcup_{k=1}^N \mathcal{P}_{i-k} \right) \]

end

——— Main ———

\[ r := 0, \mathcal{U}^{[r]} := \bigcup_{j=1}^{N_{\mathcal{P}^{[r]}}} \mathcal{U}^{[r]} \]

while $\mathcal{U}^{[r]} \neq \emptyset$

\[ r := r + 1, \mathcal{U}^{[r]} := \emptyset \]

for $i = 1$ to $N_{\mathcal{P}^{[r]}}$

\[ M' := \{ k \mid M_{\text{org}^{[r]-1}} \rightarrow \text{org}^{[r]-1} \} \]

for $j = 1$ to $M'_{\mathcal{R}^{[r]-1}}$

\[ \mathcal{P}^{[r-1]}_{i-j} := \left\{ x \in \mathbb{R}^n \mid \begin{bmatrix} P_i^{[r-1],x} \cr U_j^{[r-1],x} A_i^{\text{CL}} \end{bmatrix} x \leq \begin{bmatrix} P_i^{[r-1],0} \cr U_j^{[r-1],0} - P_i^{[r-1],0} \end{bmatrix} \right\} \]

\[ \mathcal{U}^{[r]} := \mathcal{U}^{[r]} \cup \mathcal{P}^{[r-1]}_{i-j} \]

if $\mathcal{P}^{[r-1]}_{i-j} = \bigcup_{k=1}^{N_{\mathcal{P}^{[r]}}} \mathcal{P}^{[r-1]}_{i-j}$ then next $i$

end

end

\[ \mathcal{U}^{[r]} := \bigcup_{j=1}^{N_{\mathcal{P}^{[r]}}} \mathcal{U}^{[r]} \]

end

\[ I := \mathcal{P}^{[r]} \]

**Remark 9.2.** In the case that the open-loop feasible set $\mathcal{X}^0$ is open, the Algorithm 9.1 will possibly take an infinite number of iteration steps. However, this is hardly a limitation for ‘practical’ problems where usually a bounded set of interest is considered.
9.3.2 Computation of the Lyapunov Stability Region

Next we present an algorithm to compute a Lyapunov stability region for a given CFTOC solution when the receding horizon control strategy is applied. The algorithm is based on the linear cost function and additionally uses reachability analysis for the computation. Unlike techniques presented in the literature, cf. e.g. [FTCM02, JR98], we are not looking for a piecewise quadratic Lyapunov function that provides overall stability guarantees but we are checking if the given value function of the CFTOC solution is a piecewise linear Lyapunov function. Therefore no LMI techniques are needed, and no possible conservatism is introduced.

In analogy to energy arguments it is natural to assume that the PWA value function (9.12) is a valid candidate for a PWA Lyapunov function for a region around the origin, confer [BCM03b]. In contrast to the optimal infinite time solution where the whole feasible state-space is stabilizing, we are considering the finite time solution with a receding horizon control strategy. Therefore it is not to be expected (especially for PWA systems) that with standard Lyapunov stability arguments with this choice of Lyapunov function, we are actually obtaining the whole asymptotically stable region of the closed-loop system, but only a subset of it.

The detailed description is given in Algorithm 9.2. The notation is analog to the notation used in Algorithm 9.1. Also here, note that for simplicity in the following we only mention operations on (possibly non-convex) sets and polyhedral regions; however, we also have to keep track of the cost function and state feedback control law (as in (9.12) and (9.11), respectively) assigned to each region. Speed-ups of the algorithm are again not discussed for simplicity.

The initialization part of the algorithm is completely analog to the initialization part of Algorithm 9.1 but in addition we compute the ‘Lyapunov’ set $\mathcal{L}^{[0]}$ where in one step the cost
function $J_{RH}$ is decreased, i.e.
\[
L^0 = \{ x \in \mathcal{X}^0 \mid J_{RH}(f_{PWA}^CL(x)) < J_{RH}(x) \}.
\]

Starting with $L^0$ as initialization in the second part of the algorithm, we extract the parts of $L^0$ for which the value function decays in one step. The remaining part is the new set $L^1$ and the procedure continues in an iterative way until in two consecutive steps the polyhedral partition of the Lyapunov stability region does not change, i.e. $L^r = L^{r+1} =: \mathcal{L}$.

Third part: To provide a good fix for the aforementioned deficiency of the Lyapunov function of our choice, i.e. that we (possibly) do not cover the whole asymptotic stability region, we perform a one-step reachability analysis into the Lyapunov region $L$. The newly attained regions are denoted with $A^{[1]}$. Confer also Figure 9.3. Then, in an iterative way, again a one-step reachability analysis into the regions $A^{[r-1]}$ is performed to obtain the regions $A^{[r]}$ until no new regions can be found. It is clear from the construction that the set $A^r$ is the set of states from which a trajectory is driven (with the given control law) into the computed Lyapunov stable region $L$ in $r$ steps. The union of all $A^r$ together with the Lyapunov stability region $L$ is the region of attraction $\mathcal{A}$, which itself is a Lyapunov stability region in the classical sense.

Algorithm 9.2 (Lyapunov Stability Region $L$, Region of Attraction $\mathcal{A}$).

**INPUT** CFTOC solution: $f_{PWA}(x, u), u_{RH}(x), J_{RH}(x)$

**OUTPUT** Lyapunov Stability Region $L$, Region of Attraction $\mathcal{A}$

--- Initialization ---

\[
L^{[0]} := \emptyset
\]

for $i = 1$ to $N$

compute $A_i^{CL}, a_i^{CL}$, according to (9.15)

for $j = 1$ to $N^0$

\[
P_{i \rightarrow j} := \left\{ x \in \mathbb{R}^n \mid \begin{bmatrix} P_i^x \\
A_i^{CL} \
\end{bmatrix} x \leq \begin{bmatrix} P_0^x \\
- P_j^x a_i^{CL} 
\end{bmatrix} \right\}
\]

if $P_{i \rightarrow j} \neq \emptyset$ then

$M_{i \rightarrow j} = \text{true}$,

$S_{i \rightarrow j}^{[0]} := \{ x \in \mathbb{R}^n \mid (\Phi_j A_i^{CL} - \Phi_i) x < (\Gamma_i - \Gamma_j - \Phi_j A_i^{CL}) \}$,

$L^{[0]} := L^{[0]} \cup (P_{i \rightarrow j} \cap S_{i \rightarrow j}^{[0]})$

else

$M_{i \rightarrow j} = \text{false}$

end

if $P_{i} = \cup_{j=1}^{i} P_{i \rightarrow k}$ then

$M_{i \rightarrow k} = \text{false}$ for $k = j + 1, \ldots, N$, next $i$

end

end

--- Lyapunov Stability Region $L$ ---

\[ r := 0, \quad L^{[r]} =: \bigcup_{j=1}^{N^r} \left\{ L_j^{[r]} \right\}, \quad \Delta L^{[0]} := P \setminus L^{[0]} \]
9 Stability Analysis of PWA Systems with a Linear Performance Index

while $\Delta L^{[r]} \neq \emptyset$
  $r := r + 1, \quad L^{[r]} := \emptyset, \quad \Delta L^{[r]} := \emptyset$
for $i = 1$ to $N^{[-1]}_L$
  $\mathcal{M} := \left\{ k \mid M_{\text{org}(L^{[i-1]}_i)} \rightarrow \text{org}(L^{[i-1]}_{k-1}) = \text{true} \right\}$
  for $j \in \mathcal{M}$
    $L^{[r]}_{i \rightarrow j} := \left\{ x \in \mathbb{R}^n \left| \begin{bmatrix} L^{[r-1],x}_{j-1} A^{\text{CL}}_{\text{org}(L^{[r-1]}_i)} \end{bmatrix} x \leq \begin{bmatrix} L^{[r-1],0}_{j-1} - L^{[r-1],0}_{j-1} A^{\text{CL}}_{\text{org}(L^{[r-1]}_i)} \end{bmatrix} \right. \right\}$
    if $L^{[r]}_{i \rightarrow j} \neq \emptyset$ then
      $S^{[r]}_{i \rightarrow j} := \left\{ x \in \mathbb{R}^n \left| \left( \Phi_j A^{\text{CL}}_{\text{org}(L^{[r-1]}_i)} \right) - \Phi_i \right. \right\} x < \left( \Gamma_i - \Gamma_j + \Phi_j A^{\text{CL}}_{\text{org}(L^{[r-1]}_i)} \right)$,
      $L^{[r]} := L^{[r]} \cup \left( L^{[r]}_{i \rightarrow j} \cap S^{[r]}_{i \rightarrow j} \right)$
    end
  if $L^{[r-1]}_{i} = \bigcup_{k=M_1}^{N^{[r-1]}_i} \left\{ L^{[r]}_{i \rightarrow k} \right\}$ then next $i$, end
end
$L := L^{[r]}$

Region of Attraction $A$

$r := 0, \quad A := L, \quad A^{[0]} := L = \bigcup_{j=1}^{N^{[0]}_A} \left\{ A^{[0]}_j \right\}$,
$\mathcal{R}^{[0]} := P \backslash L = \bigcup_{j=1}^{N^{[0]}_L} \left\{ R^{[0]}_j \right\}$
while $A^{[r]} \neq \emptyset$ and $\mathcal{R}^{[r]} \neq \emptyset$
  $r := r + 1, \quad A^{[r]} := \emptyset$
for $i = 1$ to $N^{[-1]}_R$
  $\mathcal{M} := \left\{ k \mid M_{\text{org}(R^{[i-1]}_{i-1})} \rightarrow \text{org}(A^{[i-1]}_{k-1}) = \text{true} \right\}$
  for $j \in \mathcal{M}$
    $R^{[r-1]}_{i \rightarrow A^{[i-1]}_j} := \left\{ x \in \mathbb{R}^n \left| \begin{bmatrix} R^{[r-1],x}_{j-1} A^{\text{CL}}_{\text{org}(R^{[r-1]}_{i-1})} \end{bmatrix} x \leq \begin{bmatrix} R^{[r-1],0}_{j-1} - R^{[r-1],0}_{j-1} A^{\text{CL}}_{\text{org}(R^{[r-1]}_{i-1})} \end{bmatrix} \right. \right\}$
    $A^{[r]} := A^{[r]} \cup \left( R^{[r-1]}_{i \rightarrow A^{[i-1]}_j} \right)$
    if $R^{[r-1]}_{i} = \bigcup_{k=M_1}^{N^{[r-1]}_r} \left\{ R^{[r-1]}_{i \rightarrow A^{[i-1]}_k} \right\}$ then next $i$, end
end
$\mathcal{R}^{[r]} := \bigcup_{j=1}^{N^{[r-1]}_\mathcal{R}} \left\{ R^{[r]}_j \right\} \backslash A^{[r]} = \bigcup_{j=1}^{N^{[r]}_\mathcal{R}} \left\{ R^{[r]}_j \right\}$,
Remark 9.3. Note that the computation of the union of regions in the various steps of Algorithm 9.1 and Algorithm 9.2 is very simple. Since the regions do not intersect we can simply represent the union as a collection of the regions.

Remark 9.4. For the case that \( 0 \in \text{int}(\mathcal{L}) \), the set \( \mathcal{A} \) computed with Algorithm 9.2 is the (maximum) region of attraction as in Definition 9.2. However, if \( 0 \not\in \text{int}(\mathcal{L}) \) then in general Algorithm 9.2 computes a subset of the (maximum) region of attraction. See Example (9.20) and the related Figure 9.10. In such a case one could either increase the prediction horizon \( T \) and check if for the new CFTOC solution \( 0 \in \text{int}(\mathcal{L}) \), or compute the region of attraction to an \( \epsilon \)-size hypercube around the origin.

Remark 9.5. As in Algorithm 9.1, it should be noted that in the case that the open-loop feasible set \( \mathcal{A}^0 \) is open, the Algorithm 9.2 will possibly not terminate in a finite number of iteration steps. However, under rather strong assumptions (e.g. exponential stability) as it was shown in [GLPM03] that a finite termination of such an algorithm can be guaranteed. For the considered case where no assumptions on the stability of the closed-loop trajectories are made, we believe that no guarantee of finite termination can be given.

Theorem 9.2 (Asymptotic Stability of \( \mathcal{A} \)). The closed-loop system (9.14) is asymptotically stabilizing to the origin for every state \( x(t) \) starting in the region of attraction \( \mathcal{A} \).

Proof The proof follows by the construction of the region of attraction \( \mathcal{A} \) (Algorithm 9.2) and standard Lyapunov stability arguments. As mentioned before, the set \( \mathcal{A} \) is itself a Lyapunov stability region in the classical sense.

Corollary 9.1. Let \( \mathcal{A} \) be the region of attraction and \( \mathcal{I} \) be the maximum positive invariant set of the closed-loop system (9.14).

(a) If \( \mathcal{A} \equiv \mathcal{I} \) then all states \( x \in \mathcal{I} \) will converge asymptotically to the origin, i.e. no limit cycles or stationary point other than the origin exist.

(b) If \( \mathcal{A} \subset \mathcal{I} \subseteq \mathbb{R}^n \) and the system is in the class of constrained PWA systems then stationary points other than the origin and/or limit cycles can lie in the set \( \mathcal{I} \setminus \mathcal{A} \).

(c) If \( \mathcal{A} \subset \mathcal{I} \subset \mathbb{R}^n \) and the system is in the class of constrained PWA systems then stationary points other than the origin and/or limit cycles exist and lie in the set \( \mathcal{I} \setminus \mathcal{A} \).

Proof The proof for Corollary 9.1 is straightforward and is omitted here. We would just like to point out that the difference in part (b) and part (c) of Corollary 9.1 is a consequence of the fact that for unbounded sets invariance does not imply stability, cf. Remark 9.1.
9.4 Examples

9.4.1 Linear System: Constrained Double Integrator

Consider the constrained double integrator

\[
\begin{align*}
  x(t+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u(t), \\
  x(t) &\in [-20, 20] \times \mathbb{R}, \\
  u(t) &\in [-1, 1].
\end{align*}
\]  
(9.19)

The constrained finite time optimal control problem (9.2)–(9.4) is solved with \( Q = I, R = 1, P = 0 \), and \( \mathcal{X}^f = [-20, 20] \times [-20, 20] \) for \( p = \infty \). \( T = 5 \) was chosen as prediction horizon. We report the solution to the CFTOC problem in Figure 9.4 (cost function \( J_{RH}(x) \)), Figure 9.5 (receding horizon control law \( u_{RH}(x) \)), and Table 9.1. For \( T = 5 \) the region of attraction \( \mathcal{A} \) and the maximum positive invariant set \( \mathcal{I} \) are identical, cf. Figure 9.6 and Figure 9.7. This means that no limit cycle or stationary points other than the origin exist. Furthermore, the stability region is a strict subset of the feasible region, i.e. there exist regions of the CFTOC solution for \( T = 5 \) which lead to infeasibility when the receding horizon policy is applied (green marked regions in Figure 9.7).

<table>
<thead>
<tr>
<th>Computation for</th>
<th>( T = 5 )</th>
<th>( T = 10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyapunov stability region ( \mathcal{L} )</td>
<td>8 iterations</td>
<td>6 iterations</td>
</tr>
<tr>
<td></td>
<td>178 regions</td>
<td>556 regions</td>
</tr>
<tr>
<td>Region of attraction ( \mathcal{A} )</td>
<td>11 iterations</td>
<td>8 iterations</td>
</tr>
<tr>
<td></td>
<td>1256 regions</td>
<td>928 regions</td>
</tr>
<tr>
<td>Max. positive invariant set ( \mathcal{I} )</td>
<td>7 iteration</td>
<td>4 iteration</td>
</tr>
<tr>
<td></td>
<td>266 regions</td>
<td>444 regions</td>
</tr>
<tr>
<td>CFTOC solution</td>
<td>82 regions</td>
<td>446 regions</td>
</tr>
</tbody>
</table>

Table 9.1: Computational results for Example (9.19).

9.4.2 Constrained PWA System

Consider the piecewise affine system [BM99]

\[
\begin{align*}
  x(t+1) &= 0.8 \begin{bmatrix} \cos \alpha(t) & -\sin \alpha(t) \\ \sin \alpha(t) & \cos \alpha(t) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \\
  \alpha(t) &= \begin{cases} \frac{\pi}{3} & \text{if } [1 \ 0]x(t) \geq 0, \\
                      \frac{-\pi}{3} & \text{if } [1 \ 0]x(t) < 0, \end{cases} \\
  x(t) &\in [-10, 10] \times [-10, 10], \\
  u(t) &\in [-1, 1].
\end{align*}
\]  
(9.20)
The constrained finite time optimal control problem (9.2)–(9.4) is solved with \(Q = I\), \(R = 1\), \(P = 0\), and \(\mathcal{X}^f = [-10, 10] \times [-10, 10]\) for \(p = \infty\). \(T = 1\) was chosen as prediction horizon. We report the results in Figure 9.8–9.10 and in Table 9.2.

From [BCM03a, BCM03b] we know that the CITOC solution (comprising 252 polyhedral regions) for the problem (9.20) is obtained with a prediction horizon \(T = T_\infty = 11\). Nevertheless, as can be seen from Table 9.2 even for the small prediction horizon of \(T = 1\) the
maximum positive invariant set is $\mathcal{I} = \mathcal{X}^0 \subset \mathbb{R}^n$ which means that the overall system is stable. The set $\mathcal{X}^0$ for $T = 1$ partitioned into only 10 polyhedral regions compared to the CITOOC solution with 252 polyhedral regions, cf. Figure 9.8. In Figure 9.10 we see that the Lyapunov stability region $\mathcal{L}$, as computed in Section 9.3.2, does not cover $\mathcal{X}^0$ and therefore (in this case) the cost function is not the best choice for a Lyapunov function. Note that $0 \not\in \text{int}(\mathcal{L})$ which, as pointed out in Remark 9.4, means that Algorithm 9.2 may (and in this case does) return a subset of the (maximum) region of attraction as seen from the trajectory in Figure 9.10. By increasing the prediction horizon to $T = 2$ we obtain $\mathcal{X}^0 = \mathcal{A}$, confer
Remark 9.4.

The performance decay index \( e_J \) and the maximum performance deviation \( e_J^{\max} \) from the optimal infinite time solution as a function of the prediction horizon \( T \) are depicted in Figure 9.11. Both are computed as follows: the feasible state-space is gridded, and for each equidistant grid point \( x_{0,i} \), \( i = 1, \ldots, N_{x0} \), as initial state the closed-loop trajectory is computed with the CITOC solution and the sub-optimal CFTOC solution for different prediction horizons \( T \), respectively. The performance decay index and the maximum performance deviation are defined as

\[
e_J(T) := \frac{1}{N_{x0}} \sum_{i=1}^{N_{x0}} \left| J_T^*(x_{0,i}) - J_\infty^*(x_{0,i}) \right| \quad \text{and} \quad e_J^{\max}(T) := \max_{x_{0,i}} \left| \frac{J_T^*(x_{0,i}) - J_\infty^*(x_{0,i})}{J_\infty^*(x_{0,i})} \right|.
\]

One can see that already for \( T = 3 \) the (averaged) relative error in the cost is less than 1.5\%. And for increasing prediction horizon the error vanishes quickly.
9.4.3 Constrained PWA System

Consider the piecewise affine system [MR03]

\[
x(t + 1) = \begin{cases} 
    \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{if } x_1 \leq 1, \\
    \begin{bmatrix} 0.5 & 0.2 \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} & \text{if } x_1 \geq 1,
\end{cases}
\]

\[
\begin{bmatrix}
-1 & 1 \\
-3 & -1 \\
0.2 & 1 \\
-1 & 0 \\
1 & 0 \\
0 & -1 \\
u(t) \in [-1, 1]
\end{bmatrix}
\begin{bmatrix}
x(t) \\
15 \\
9 \\
6 \\
8 \\
10
\end{bmatrix}
\leq
\begin{bmatrix}
15 \\
25 \\
9 \\
6 \\
8 \\
10
\end{bmatrix},
\]

The constrained finite time optimal control problem (9.2)–(9.4) is solved with \( Q = I \), \( R = 0.1 \), \( P = 0 \), and \( \mathcal{X}^f = [-10, 10] \times [-10, 10] \) for \( p = \infty \). \( T = 4 \) was chosen as prediction horizon. The results are reported in Figures 9.12–9.15 and in Table 9.3.

For \( T = 4 \) the region of attraction \( \mathcal{A} \) and the maximum positive invariant set \( \mathcal{I} \) are identical\(^4\), cf. Figure 9.14 and Figure 9.15. This means that no limit cycle or stationary points other than the origin exists. Furthermore, the stability region is a strict subset of the feasible region, i.e. there exist regions of the CFTOC solution for \( T = 4 \) which lead to infeasibility when the receding horizon policy is applied (green marked regions in Figure 9.15).

<table>
<thead>
<tr>
<th>Computation for</th>
<th>( T = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyapunov stability region ( \mathcal{L} )</td>
<td>7 iterations 201 regions</td>
</tr>
<tr>
<td>Region of attraction ( \mathcal{A} )</td>
<td>9 iterations 370 regions ( \mathcal{A} = \mathcal{I} \subset \mathcal{X}^0 )</td>
</tr>
<tr>
<td>Max. positive invariant set ( \mathcal{I} )</td>
<td>8 iteration 123 regions ( \mathcal{A} = \mathcal{I} \subset \mathcal{X}^0 )</td>
</tr>
<tr>
<td>CFTOC solution</td>
<td>119 regions</td>
</tr>
</tbody>
</table>

Table 9.3: Computational results for Example (9.22).

\(^4\)The solution for \( T = 3 \) was not computed. The mentioned results for \( T = 4 \) might also hold for lower prediction horizons.
Figure 9.12: Cost $J_{RH}(x)$ for Example (9.22) with $T = 4$. Same color corresponds to the same cost value.

Figure 9.13: Control law $u_{RH}(x)$ for Example (9.22) with $T = 4$. Blue ($u = -1$) and red-brown ($u = 1$) marked regions correspond to the saturated control action.

Figure 9.14: Lyapunov stability region $\mathcal{L}$ (red) and region of attraction $\mathcal{A}$ (red+grey-shades) for Example (9.22) with $T = 4$.

Figure 9.15: Maximum positive invariant set $\mathcal{I}$ for Example (9.22) with $T = 4$. The same color corresponds to the same cost value. Green marked regions lead to infeasibility.
Figure 9.16: Simulation for Example (9.23) with $T = 3$. Red marked trajectories lead to infeasibility. The purple star marks the origin.

### 9.4.4 3-dimensional Constrained PWA System

Consider the piecewise affine system [MR03]

$$
\begin{align*}
x(t + 1) &= \begin{cases}
\begin{bmatrix}
1 & 0.5 & 0.3 \\
0 & 1 & 1 \\
1 & 0.2 & 0.3 \\
0 & 0.5 & 1 \\
0 & 0 & 1
\end{bmatrix} x(t) + \begin{bmatrix}
0 \\
0 \\
1 \\
0 \\
1
\end{bmatrix} u(t) + \begin{bmatrix}
0 \\
0 \\
0 \\
0.3 \\
0
\end{bmatrix} & \text{if } x_2 \leq 1, \\
\begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & -1 \\
0 & 0 & 1
\end{bmatrix} x(t) + \begin{bmatrix}
10 \\
10 \\
5 \\
10 \\
10
\end{bmatrix} & \text{if } x_2 \geq 1,
\end{cases}
\end{align*}
$$

(9.23)

The constrained finite time optimal control problem (9.2)–(9.4) is solved with $Q = I$, $R = 0.1$, $P = 0$, and $\mathcal{X}^f = [-10, 10] \times [-10, 10] \times [-10, 10]$ for $p = \infty$. $T = 3$ ($T = 4$) was chosen as prediction horizon. The results are reported in Table 9.4. There we see that during the computation of the maximum positive invariant set 2632 (665) regions of the CFTOC solution for $T = 3$ ($T = 4$) lead to infeasibility when the receding horizon policy is applied (red marked trajectories in Figure 9.16). This implies that the region of attraction $\mathcal{A} \subseteq \mathcal{I} \subset \mathcal{X}^0$. 
Table 9.4: Computation results for Example (9.23).

9.5 Conclusion

We have presented an algorithm that by analyzing the CFTOC solution (based on the linear cost function) for constrained piecewise affine systems extracts regions of the state-space for which closed-loop stability and feasibility can be guaranteed. The algorithm combines reachability analysis with some basic polyhedral manipulation for the computation of the maximum positive invariant set and for the computation of a Lyapunov stability region. The simplicity of the overall computation stems from the fact that in all steps of the algorithm only linear programs need to be solved. Applicability of the algorithm was illustrated with several numerical examples.
Part III

CASE STUDIES
Multi-object Adaptive Cruise Control

Cruise control is a common and well known automotive driver assistance system in which the driver sets a reference speed and the engine is controlled so that this reference speed is maintained regardless of external loads such as wind, road slope or changing vehicle parameters. Adaptive Cruise Control (ACC) additionally takes into account the traffic in front of the car. In a multi-object adaptive cruise control problem the optimal acceleration of the driver’s car is to be found respecting traffic rules, safety distances and driver intentions. The control objectives are

- to track the reference speed
- to respect safety distance if a neighboring car is in the same lane, and
- not to overtake on the right side of a neighboring car

while constraining acceleration and changes in acceleration and in deceleration. Loosely speaking we would like to maintain a comfortable drive for our car and respect traffic rules. The hybrid nature of the problem arises from the multiple objectives which include switches.

The optimal state-feedback control law (for a quadratic objective function) is found by solving the underlying Constrained Finite Time Optimal Control problem via Dynamic Programming [BBBM03a]. The optimal state-feedback control law was tested on a research car Mercedes E430. Special interfaces to throttle and brakes, sensor fusion, visualization and the ACC controller were running in a real-time environment with an 80 ms cycle time on an Intel Pentium4 1.4GHz machine with 500 MByte RAM.

10.1 Introduction

Cruise control is a common and well known automotive driver assistance system. The driver sets a reference speed and the engine is controlled so that this reference speed is maintained regardless of external loads such as wind, road slope or changing vehicle parameters. Further development led to the so called Adaptive Cruise Control (ACC) that takes into account the traffic in front of the car. Good acceleration and deceleration control is essential for the ACC systems and over the last decades it has been subject of extensive research. The reader is referred to [LRWK93, HTB89, WH01] for an overview.
To obtain information about the distance between vehicles the ACC system uses infrared or radar sensors. If another car crosses into the driver’s lane, and the distance is less than a certain safety distance (a separation interval of 1 to 2 seconds), the control system slows down the car by reducing the throttle or applying the brakes. If the leading car speeds up or goes out of the driver’s lane the controller accelerates the vehicle to the cruising speed. In all those maneuvers the ACC system has to deal with the constraints imposed on acceleration and deceleration. Acceleration limitation increases comfort, while with deceleration limitation we avoid unnecessary emergency brakes that can be caused by sensor noise, limited sensor range, or the imperfections of the traffic scene modeling. However, limiting deceleration has its drawbacks, since, in rare but dangerous situations, an accident could be avoided if higher deceleration is used. Additionally, the ACC system should respect traffic rules (e.g. overtaking only on the left) and lane changers have to be considered early to avoid dangerous situations. To fulfill all those requirements, sensors and processing systems should be used, that can detect and represent the complete traffic scene ahead of the car. In that way we can achieve a better prediction and design a better controller in a multi-object scenario.

In this chapter we present a way to model and estimate a complex traffic scene sensed by radar, infrared laser and video sensors and compute the optimal acceleration as a state-feedback control law by using hybrid system theory. Estimation and representation of the traffic scene were implemented and tested. Simulation and experimental results of the designed controller are reported.

### 10.2 Modelling and Estimation

#### 10.2.1 Sensors

To get a good representation of the actual traffic scene radar, infrared and stereo vision systems are mounted on a research car. With these external sensors, the road ahead, as well as position and velocity of the neighboring cars, can be estimated. Internal sensors deliver information about speed, acceleration and yaw rate of the driver’s vehicle (ego-car). Figure 10.1 shows the ranges of sight of the different external sensors.

The radar sensor measures distance and relative speed independently based on different physical principles. The sensor’s range is up to 150 m with an aperture angle of 7° but a rather poor angular resolution. The infrared sensors scans nearly 180° up to about 50 m. The stereo vision system has the best lateral resolution but the object recognition based on disparity detection and clustering is sensitive to noise. The idea is to use all these sensors and to benefit from the best of each one of them.

---

1Radar sensors at the same time provide distance and relative speed measurement.

2Throughout this chapter with the word/subscript ego we refer to the driver’s vehicle, while with the word/subscript obj we refer to the vehicle(s) that is in the driver’s range of sight.
10.2.2 Lane Recognition

For an accurate ACC system it is not enough to know the relative positions and velocities of the neighboring cars. Only with additional information about the road curvature good ACC performance can be achieved. For this reason a vision-based lane recognition system is implemented in the car. Optical lane recognition was introduced in the 1980s [ZD86] and has often been implemented and refined since. The system is thoroughly described in many publications [ZD86, FGG+01, FGG+98], but the modeling shall be repeated here briefly. According to recommendations for highway construction [Ric84], highways are built with the constraint of slowly changing curvature. Therefore the lane recognition system is based on a clothoidal lane model, that is given by

\[
c(L) = c_0 + c_1 L. \tag{10.1}
\]

where \( c(L) \) describes the curvature at arclength \( L \) of the clothoid, \( c_0 \) the initial curvature and \( c_1 \) the curvature rate or clothoid parameter. The clothoid is defined as \( c = R^{-1} \), with \( R \) the radius of the curve. With this road model and assuming only small angles and curvatures (which is true for highways and highway-like roads) a simple model of the vehicle in the lane can be stated as

\[
\dot{x}_L = A_L x_L + B_L u_L \tag{10.2}
\]

where

\[
x_L = \begin{bmatrix} x_{\text{off}} \\ \Delta \Psi \\ c_0 \\ c_1 \end{bmatrix}, \quad A_L = \begin{bmatrix} 0 & v_{\text{ego}} & 0 & 0 \\ 0 & 0 & -v_{\text{ego}} & 0 \\ 0 & 0 & 0 & v_{\text{ego}} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_L = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad u_L = \begin{bmatrix} v_{\text{lat}} \\ \dot{\Psi} \end{bmatrix}. \tag{10.3}
\]

Here \( x_{\text{off}} \) denotes the lateral offset of the car in the lane, \( \Delta \Psi \) the yaw angle, i.e. the angle between road direction and vehicle direction, \( \dot{\Psi} \) the yaw rate i.e. the rotational speed of the
car, $v_{ego}$ the longitudinal and $v_{lat}$ the lateral speed of the car. To determine the offset of the car from the middle of the lane, the width of the lane $w$ is needed as well. To estimate $x_L$ (for illustration see Figure 10.2), image processing algorithms detect the white lane markings. With these lane markings and internal sensor data the states from (10.2) can be estimated via Kalman filtering. The lane width $w$ is treated as a disturbance variable.

### 10.2.3 Object Recognition

With all sensors having different ranges, measuring principles and accuracies, various algorithms are needed to process collected sensor data. Therefore, we set up the following model of longitudinal and lateral dynamics of the ego-vehicle and the obj-vehicle

\[
\dot{x}_M = A_M x_M
\]

where

\[
x_M = \begin{bmatrix} d_i \\ v_{ego} \\ a_{ego} \\ v_{obj, i} \\ a_{obj, i} \\ d_{lat, i} \\ v_{lat, i} \end{bmatrix}, \quad A_M = \begin{bmatrix} 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.
\]

Here $v_{ego}$, $a_{ego}$ and $v_{obj, i}$, $a_{obj, i}$ refer to velocity and acceleration of the ego-vehicle and the $i^{th}$ neighboring car respectively, $d_i$ denotes the longitudinal distance between the ego-vehicle and the $i^{th}$ neighboring car, $d_{lat, i}$ and $v_{lat, i}$ describe relative lateral distance and speed of the $i^{th}$ neighboring car.

The dimension of system (10.5) varies with the number of neighboring cars that are taken into account. This model is then used for the Kalman filter based estimators. With measurements for $d_i$, $v_{ego}$, $a_{ego}$, $v_{rel, i} = v_{obj, i} - v_{ego}$ and $d_{lat, i}$ system (10.5) is observable. The
measurement noise parameter of the Kalman filter is used to weight measurements of the three sensor types differently. Distance measurements from more trustworthy sensors are respected with smaller measurement noise than measurements from less trustworthy sensors. To achieve good data association and tracking behavior in the presence of noise, probabilistic data association filters (PDAF) with gating techniques are used [BSF88, BSB00, Bla86]. Further, it was observed that most of the time a neighboring car’s motion can be described well either with zero acceleration or with a rather strong acceleration. For this reason multiple model filters are used in the estimation process [Mag65, BBS88]. For longitudinal modeling one filter assumes constant speed and another filter constant acceleration; for lateral dynamics one filter runs with a zero velocity and one with a constant velocity model. The two filters then interact according to their likelihood ratio.

10.2.4 Control Model

For control a model is needed to predict the future system behavior, given the actual states. With small changes, system (10.5) can be used. In system (10.5) the relative lateral distance \( d_{lat_i} \) was introduced. In the ACC control problem, however, the difference in lane offset \( \delta_i \) is important rather than \( d_{lat_i} \). In exaggerated form this can be seen in Figure 10.3. Here \( d_{lat_i} \) is zero since the advancing car is exactly in the direction of the ego-vehicle. The difference in lane offset, \( \delta_i \), however, is clearly not zero. The states that are needed to calculate \( \delta_i \) and its derivative \( \nu_i \) are all known from lane recognition and object recognition. The calculation is as follows (see Figure 10.4):

\[
\delta_i = x_{off} - x_{off_i} = -d_{lat_i} - \Delta \Psi d_i + \frac{1}{2} c_0 d_i^2 + \frac{1}{6} c_1 d_i^3, \tag{10.6}
\]

\[
\nu_i = \frac{\partial}{\partial t} \delta_i = -v_{lat_i} + \frac{1}{2} c_1 v_{ego} d_i^2 - (\Psi - c_0 v_{ego}) d_i. \tag{10.7}
\]

In (10.7) we assume \( d_i \) to be constant. As control variable, the reference acceleration, \( a_{ref} \), of the ego-vehicle is introduced. We will assume that a low level acceleration controller, which manipulates the vehicle engine, gear shift and brakes, can be described as a first order lag with time constant \( \tau \).

Introducing the control variable \( a_{ref} \) and substituting \( d_{lat_i} \) and \( v_{lat_i} \) with \( \delta_i \) and \( \nu_i \), respectively, we obtain the continuous model of the system

\[
\dot{x} = A_c x + B_c u \tag{10.8}
\]

where

\[
x = \begin{bmatrix}
    d_i \\
v_{ego} \\
a_{ego} \\
v_{obj_i} \\
a_{obj_i} \\
\delta_i \\
\nu_i
\end{bmatrix}, \quad u = \begin{bmatrix} a_{ref} \end{bmatrix}, \quad A_c = \begin{bmatrix} 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{\tau} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\
0 \\
\frac{1}{\tau} \\
0 \\
0 \\
0 \\
0 \end{bmatrix}. \tag{10.9}
\]
Figure 10.3: Illustration of a difference in the lane offset.

Figure 10.4: Computation of the difference in the lane offset $\delta$. 
For control purpose we are interested in the discrete representation

\[ x(k + 1) = A_d x(k) + B_d u(k) \] (10.10)

that is obtained by sampling system (10.8) with the Zero Order Hold (ZOH).

### 10.3 Problem Formulation

Control objectives for a traffic scene depicted in Figure 10.5 are the following

- track reference speed \( v_{\text{ref}} \) if possible
- respect safety distance \( d_{\text{min}} \) if neighboring car is in the same lane
- do not overtake on the right side of a neighboring car

while constraining

- maximal acceleration
- changes in acceleration
- changes in deceleration

Loosely speaking we would like to maintain a comfortable drive for the ego-car and prevent any obj-car from entering the shaded area in Figure 10.5. By inspection of the above mentioned objectives - inherent logic (traffic rules and lane assignment) and constraints on system states and input - it is straightforward to see that we are dealing with a constrained hybrid optimal control problem.

Whenever obj-car is outside of the shaded area (Figure 10.5), only the reference speed should enter the cost function

\[ J = (v_{\text{ego}}(k) - v_{\text{ref}})^2 Q_v, \]
but if \( \text{obj-car} \) is closer than a certain \( d_{\text{min}} \) and it is in the same lane or on the left of the \( \text{ego-car} \), the safety distance violation should enter the cost

\[
J = (v_{\text{ego}}(k) - v_{\text{ref}})^2 Q_v + (d_{\text{min}} - d(k))^2 Q_d,
\]

where \( Q_v \) and \( Q_d \) are appropriately chosen weights.

Note that the problem has two different cost functions defined over two regions, one of which is a non-convex polyhedron. Hence, to formulate one overall objective function we have to introduce some binary information.\(^3\) For this purpose two new states are introduced. One of those states corresponds to the past manipulated variable \( u(k-1) = a_{\text{ref}}(k-1) \), and is needed if changes in acceleration and deceleration are to be constrained. The second additional state incorporates the above mentioned logic condition

\[
h_i(k) = \begin{cases} 
  d_{\text{min}} - d_i(k) & \text{if } d_i(k) \leq d_{\text{min}} & \delta_i(k) \geq \delta_{\text{min}} \\
  0 & \text{if } \text{otherwise}
\end{cases}
\]

so that the overall cost can be expressed simply as

\[
J = (v_{\text{ego}}(k) - v_{\text{ref}})^2 Q_v + \sum_{i=1}^{\#\text{obj}} (h_i(k))^2 Q_d
\]

where \( \delta_{\text{min}} \) is lateral distance for which \( \text{obj-car} \) is considered to be in the same lane as \( \text{ego-car} \), and \( \#\text{obj} \) is a total number of \( \text{obj-cars} \).

In this chapter we will focus on a simplified scenario when \( \text{ego-car} \) is always driving in the middle lane and there is only one \( \text{obj-car} \). Although simplified, this model still captures all important ingredients of the ACC control problem.

### 10.4 PWA Model of the System

We use a discrete-time piecewise affine (PWA) model of the system [Son81]

\[
x(t+1) = A_i x(t) + B_i u(t) + f_i \\
\text{if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \mathcal{C}_i \\
i = 1, \ldots, s
\]

where \( x \in \mathbb{R}^{n_c} \times \{0,1\}^{m_c}, u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_r}, \{C_i\}_{i=1}^s \) is a polyhedral partition of the sets of state+input space \( \mathbb{R}^{n+m} \), \( n := n_c + n_{t} \), \( m := m_c + m_{t} \).

The equivalent PWA model is derived from the continuous model (10.8) after incorporating additional states \( u(k-1) \) and \( h(k) \). For simplicity, constant safety distance \( d_{\text{min}} \), constant lateral distance \( \delta_{\text{min}} \) and a constant reference speed \( v_{\text{ref}} \) are assumed. The maximum acceleration and deceleration are constrained to \( \pm a_{\text{ref max}} \) and the change in acceleration is constrained to \( \pm \Delta a_{\text{ref max}} \). Assuming that a time constant \( \tau \) of the low level acceleration

\(^3\)If lateral dynamics of the \( \text{obj-car} \) is neglected, or, equivalently, if \( \text{ego-car} \) is driving in the rightmost lane, the problem can be simplified even further. In such a case, using argumentation from [BBBM02], the problem can be reformulated as a multi-parametric Quadratic Program (mp-QP).
controller is known, by sampling the system (10.8) with Zero Order Hold (ZOH) with the sampling time $T_s$ we get the following discrete-time PWA model

$$
x(t + 1) = A_i x(t) + B_i u(t) + f_i \quad \text{if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in C_i := \left\{ \begin{bmatrix} x \\ u \end{bmatrix} : H_i x + L_i u \leq K_i \right\}
$$

(10.14)

where the state vector is

$$
x(k) = [d(k), v_{ego}(k), a_{ego}(k), v_{obj}(k), a_{obj}(k), \delta(k), \nu(k), a_{ref}(k - 1), h(k)]^T
$$

and input is defined as $u(k) = a_{ref}(k)$. Having in mind that there are two cost functions we are trying to define, it seems natural to use only 2 regions in the PWA model. However, in the PWA model (10.14) we have 3 regions. The reason for such a choice is illustrated in Figure 10.6. Note that the region where contribution of distance $d_i$ to the overall cost (10.12) is zero has a non-convex shape. Hence, it has to be sliced into two convex polyhedra. Numerical values of the parameters and system matrices $A_i$, $B_i$, $f_i$, $H_i$, $L_i$ and $K_i$ are given in the Appendix. Note that although original system has only one dynamics we had to introduce some logic condition to describe overall objective. In our example logic is incorporated in the auxiliary state $h(k)$ and in the state+input space partition. As a consequence we obtain a PWA model of the system.
10.5 Optimal State Feedback Control Law for PWA Systems

Consider the PWA system (10.13) subject to input and state constraints

\[ E_c x(t) + L_c u(t) \leq M_c \quad (10.15) \]

for \( t \geq 0 \).

Denote by constrained PWA system (CPWA) the restriction of the PWA system (10.13) over the set of states and inputs defined by (10.15),

\[ x(t + 1) = A_i x(t) + B_i u(t) + f_i \quad \text{if } \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \in \tilde{C}_i \quad (10.16) \]

where \( \{\tilde{C}_i\}_{i=1}^s \) is the new polyhedral partition of the sets of state+input space \( \mathbb{R}^{n+m} \) obtained by intersecting the polyhedrons \( C_i \) in (10.13) with the polyhedron described by (10.15). Let \( \tilde{C} := \bigcup_{i=1}^s \tilde{C}_i \).

In the following we will denote the CPWA system equations (10.16) with the shorter form

\[ x(k + 1) = \tilde{f}_{PA}(x(k), u(k)) \quad (10.17) \]

Define the following cost function

\[ J(U_0^{T-1}, x(0)) := \|P(x(T) - x_e)\|_2 + \sum_{k=0}^{T-1} \|Q(x(k) - x_e)\|_2 + \|R(u(k) - u_e)\|_2 \quad (10.18) \]

and consider the constrained finite-time optimal control problem (CFTOC)

\[ J^*(x(0)) := \min_{U_0^{T-1}} J(U_0^{T-1}, x(0)) \quad (10.19) \]

s.t. \[ \left\{ \begin{array}{l} x(t + 1) = \tilde{f}_{PA}(x(t), u(t)) \\ x(T) \in \mathcal{X}^f \end{array} \right. \quad (10.20) \]

where the column vector \( U_0^{T-1} := [u'(0), \ldots, u'(T-1)]' \in \mathbb{R}^{nT} \) is the optimization vector, \( T \) is the time horizon and \( \mathcal{X}^f \) is the terminal region. In (10.18), \( \|Qx\|_2 = x'Qx \) and \( R = R' > 0, \ P = P' \geq 0 \). We denote by \( \mathcal{X}^0 \subseteq \mathbb{R}^n \) the set of initial states \( x(0) \) for which the optimal control problem (10.18)-(10.20) is feasible. Similarly, \( \mathcal{X}^k \) denotes the set of feasible states \( x(k) \), \( k = 1, \ldots, T \) at time \( k \) for the optimal control problem (10.18)-(10.20).

10.6 Simulation Results

We design a receding horizon controller based on the optimization problem (10.18)-(10.20) for a system (10.14)-(10.8), with \( T_s = 0.5 \), \( T = 3 \), \( Q = diag([0 1 0 0 0 0 10]) \), \( R = 1 \), \( P = Q \), and desired velocity \( v_{ref} = 30m/s \). The optimal control law is computed off-line with a dynamic programming procedure described in Chapter 6.
The corresponding optimal solution is composed of 2740 polyhedra in $\mathbb{R}^9$ (the dimension of the state space is 9). In Figure 10.7 we show one slice of the optimal state feedback control law partition for the fixed states $a_{ego} = 0\, m/s^2$, $v_{obj} = 30\, m/s$, $a_{obj} = 0\, m/s^2$, $a_{ref}(k-1) = 0\, m/s^2$, $\delta = 2\, m$, $\nu = 0\, m/s$. In Figure 10.8, starting with initial state $x(0) = [45\, 30\, 2\, 0\, 0\, 0\, 0\, 0\, 0]^T$, we report the following traffic scenario. An ego-car and an obj-car are driving in the middle lane (see Figure 10.5). obj-car has a constant speed $v_{obj} = 28\, m/s$ and it is moving to the right lane with the constant lateral velocity $\nu = -0.1$. Since $v_{obj} < v_{ref} = 30$ we notice that at $t = 3s$ obj-car reaches the safety distance and from that point onward it is maintaining the same speed as the obj-car. At $t = 20s$ obj-car leaves the middle lane and ego-car speeds up to the reference cruising speed $v_{ref}$.

### 10.7 Experimental Results

For the experiment we simplify problem (10.18)–(10.20) for a system (10.14)–(10.8) by neglecting the lateral dynamics of the obj-car. We design a receding horizon with $T = 3$, $Q = diag([0\, 1\, 0\, 0\, 0\, 0\, 10])$, $R = 1$, $P = Q$, $v_{ref} = 30\, m/s$ and $d_{min} = 40\, m$. The corresponding optimal solution is composed of 753 polyhedra in $\mathbb{R}^7$ (dimension of the state space is 7). In Figure 10.9 we report the response of the system starting from $x(0) = [45\, 23\, 0\, 20\, 0\, 0\, 0\, 0\, 0]^T$ when the optimal state-feedback control law is used. Similarly to Figure 10.8, we see that ego-car reaches and keeps a safety distance $d_{min} = 40\, m$ while tracking the speed of obj-car. The optimal state-feedback control law was tested on a research car Mercedes E430 (Figure 10.10). Special interfaces to throttle and brakes, sensor fusion, visualization and the ACC controller were running in a real-time environment with an 80 ms cycle time on an Intel Pentium4 1.4GHz machine with 500 MByte RAM. In Figure 10.11 experimen-
10 Multi-object Adaptive Cruise Control

Figure 10.8: Simulation of the ACC system with implemented optimal state-feedback controller for initial state $x(0) = [45 \ 30 \ 0 \ 28 \ 0 \ 2 - 0.1 \ 0 \ 0]^T$ and with obj-car slowly moving out of ego-car’s lane.

Figure 10.9: Simulation of the ACC system with implemented optimal state-feedback controller for initial state $x(0) = [45 \ 23 \ 0 \ 20 \ 0 \ 0 \ 0]^T$. Lateral dynamics is neglected and the obj-car has a constant speed $v_{obj} = 20$. 
The plots start with initial states approximately \( x(0) = [28 \ 26 \ -0.5 \ 23.5 \ 0 \ -0.7 \ 12]^T \). As time increases, the distance between ego-car and obj-car approaches \( d_{\text{min}} = 40\text{m} \) and \( v_{\text{ego}} \) approaches \( v_{\text{obj}} \), as can be seen in Figure 10.11(a). The corresponding accelerations \( a_{\text{ego}} \) and \( a_{\text{obj}} \) and controller action \( a_{\text{ref}} \) are shown in Figure 10.11(b).

In Section 10.3 and Section 10.4 the piecewise affine problem formulation was introduced to incorporate the lane assignment into the controller design. In this approach the controller decides implicitly whether an obj-car is relevant for the controller or not. In a second experiment lane assignment is calculated every cycle by an external logic before the controller is called. If the obj-car changes its lane to the right it is declared not relevant and thus it is not taken into account by the controller. In Figure 10.12 results of this strategy are shown. The scenario starts with \( v_{\text{ego}} > v_{\text{obj}} \) and \( d > d_{\text{min}} \). In the sequel the distance between ego-car and obj-car approaches \( d_{\text{min}} = 40\text{m} \) and \( v_{\text{ego}} \) approaches \( v_{\text{obj}} \). At time instant \( t = 6\text{s} \) the obj-car changes its lane to the right and is thus declared not relevant. As time goes on \( v_{\text{ego}} \) approaches \( v_{\text{ref}} = 30\frac{\text{m}}{\text{s}} \). Note that although from \( t = 6\text{s} \) the obj-car is in the right lane (and neglected by the controller) it is still seen by the sensors until \( t = 12.7\text{s} \), when it completely moves out of the sight. This can be seen as a drop of the distance \( d \) to the zero value in Figure 10.12(a).

### 10.8 Conclusion

We have shown that the Multi-Object Adaptive Cruise Control problem can be solved via hybrid system theory. The objective function is modelled as a quadratic cost function for the discrete-time piecewise affine system. The optimal state-feedback control law is found by solving the underlying constrained finite time optimal control problem via dynamic programming. Experimental results were presented for the car-following scenario.
Figure 10.11: Experimental results of the ACC system with implemented optimal state-feedback controller. Initial state is approximately $x(0) = [28 \ 26 \ -0.5 \ 23.5 \ 0 \ -0.7 \ 12]^T$. After a transient ego-car manages to track speed of obj-car while maintaining the safety distance.

Figure 10.12: Experimental results of the ACC system with implemented optimal state-feedback controller. Until $t = 7s$ obj-car is in the same lane as ego-car, but then it goes into the right lane. From that point controller has only reference speed as an objective. At $t = 12.7s$ obj-car leaves the road completely.
Appendix

In the following we report numerical values for the PWA model (10.14):

\[ d_{\text{min}} = 40 \text{m}, \quad \delta_{\text{min}} = 3 \text{m}, \quad v_{\text{ref}} = 30 \frac{\text{m}}{\text{s}}, \]

\[ a_{\text{ref max}} = 1.5 \frac{\text{m}}{\text{s}^2}, \quad \Delta a_{\text{ref max}} = 0.02 \frac{\text{m}}{\text{s}^3}, \]

\[ \tau = 0.1 \text{s}, \quad T_s = 2.7 \text{s}. \]

\[
A_1 = \begin{bmatrix}
1 & -2.7 & -0.26 & 2.7 & 3.645 & 0 & 0 & 0 & 0 \\
0 & 1 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 2.7 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 2.7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad B_1 = \begin{bmatrix}
-3.385 \\
2.6 \\
1 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
\end{bmatrix}, \quad f_1 = \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix},
\]

\[
H_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad L_1 = \begin{bmatrix}
0 \\
0 \\
-3.385 \\
2.6 \\
1 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}, \quad K_1 = \begin{bmatrix}
d_{\text{min}} \\
-\delta_{\text{min}} \\
\end{bmatrix}.
\]

\[
A_2 = \begin{bmatrix}
1 & -2.7 & -0.26 & 2.7 & 3.645 & 0 & 0 & 0 & 0 \\
0 & 1 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 2.7 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 2.7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad B_2 = \begin{bmatrix}
-3.385 \\
2.6 \\
1 \\
0 \\
0 \\
0 \\
1 \\
0 \\
0 \\
\end{bmatrix}, \quad f_2 = \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix},
\]

\[
H_2 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad L_2 = \begin{bmatrix}
0 \\
0 \\
-3.385 \\
2.6 \\
1 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}, \quad K_2 = \begin{bmatrix}
d_{\text{min}} \\
-\delta_{\text{min}} \\
\end{bmatrix}.
\]

\[ A_3 = A_2, \quad B_3 = B_2, \]

\[ f_3 = f_2, \]

\[ H_3 = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad L_3 = \begin{bmatrix}
0 \\
\end{bmatrix}, \quad K_3 = \begin{bmatrix}
-d_{\text{min}} \\
\end{bmatrix}. \]

Matrices \( E_c, L_c \) and \( M_c \) in (10.15) are easily derived from general constraints

\[-a_{\text{ref max}} \leq u(k) \leq a_{\text{ref max}}, \]

\[-\Delta a_{\text{ref max}} \leq u(k) - x_8(k) \leq \Delta a_{\text{ref max}}.\]
11

Constrained Finite Time Optimal Control of an Electronic Throttle

The overall car performance is strongly influenced by the quality of the control of the electronic throttle – a DC motor driven valve that regulates the inflow of air to the vehicle’s engine. The controller design for the throttle system is a challenging task since the controller has to cope with two strong nonlinear effects: the gearbox friction and the so-called “limp-home” nonlinearity of the return spring. In this chapter we address this problem by solving a constrained optimal control problem formulated for the discrete-time piecewise affine model of the throttle. The look-up table like solution to the optimal control problem – constructed in an off-line, dynamic programming procedure – allows the controller implementation in real-time, which would otherwise be impossible to achieve due to the small sampling time for the application at hand.

11.1 Introduction

The electronic throttle has gradually become an essential element of many engine control systems. Compared to its mechanical counterpart, the electronic throttle leads to significant improvements in vehicle emission, fuel economy and drivability, if it is well controlled. Since, one way or another, all before mentioned goals can be formulated as a constrained optimal control problems, it is natural to consider the implementation of a Model Predictive Control (MPC) scheme in order to achieve those goals.

Model Predictive Control with its receding horizon policy has proven itself trustworthy numerous times in the past few decades, especially so when dealing with discrete-time linear systems with constraints (see for instance [MRRS00, Mac02]). Unfortunately, there are several major obstacles that prevents us from simply applying the existing MPC framework for our particular problem. Namely, massive use of electronic throttle in automotive industry also means that the throttle is usually composed of relatively cheap components. As a consequence, inside of the throttle body high gear friction occurs. On top of that, the throttle has an embedded mechanical safety feature the so-called Limp-Home (LH) position nonlinearity, which guarantees a specific level of air inflow even in the case of total power failure. Nevertheless, a major obstacle for the successful implementation of MPC turns out not to be the process nonlinearity but rather the time within which the optimal control law
has to be computed. The sampling time of the electronic throttle system is in the range of milliseconds and that puts a huge strain on the MPC implementation.

To meet the demands for the small computation time and compensation of the existing process nonlinearities, we consider an optimal control of a discrete-time linear hybrid model of the throttle system with a quadratic performance criterion, which results with a state-feedback controller.

This work is mostly experimental verification of the ideas presented in Chapter 6. We begin by defining a continuous time nonlinear model of the electronic throttle in Section 11.2, from which we derive a discrete-time PWA model of the system in Section 11.3. Finally, in Section 11.4 we show the experimental and simulation results of the optimal controller.

### 11.2 Electronic Throttle Model

The throttle valve, shown in Figure 11.1, regulates the air inflow to the vehicle engine. It is a DC servo system (Figure 11.2), where the DC drive is supplied from a bipolar chopper. Motor shaft rotation is transmitted through a gearbox to the shaft with the throttle plate. The movement of the plate continues until the motor torque is balanced with the torque generated by the return spring which is attached to the plate’s shaft.

The dynamical behavior of the throttle system can be described with the following equations

\[
L_a \frac{di_a}{dt} + R_a i_a = u_a - K_v \omega_m, \tag{11.1}
\]

\[
u_a = K_{ch} u, \tag{11.2}
\]

\[
m_m = K_i i_a, \tag{11.3}
\]

\[
m_{app} = m_m - m_S - m_L, \tag{11.4}
\]

\[
\frac{d\theta}{dt} = K_v \omega_m, \tag{11.5}
\]

\[
m_f = m_f(m_{app}, w_m), \tag{11.6}
\]
\[ m_S = m_S(\theta) \]  

Figure 11.2: Electronic throttle system.

Figure 11.3: Nonlinear model of the throttle system.

where we denote with \( u \) the input control voltage, \( u_a \) the DC motor armature voltage, \( K_{ch} \) the chopper gain, \( i_a \) the DC motor armature current, \( m_m \) the motor torque, \( m_S \) the return spring torque, \( m_L \) the load torque, \( m_{app} \) the applied torque, \( m_f \) the friction torque, \( \omega_m \) the motor angular velocity, \( \theta \) the position (angle) of the throttle plate, \( R_a \) the overall armature resistance, \( L_a \) the armature inductance, \( K_t \) the motor torque constant, \( K_v \) the electromotive force constant, \( K_l \) the gear ratio.

The block diagram of the full nonlinear model of the process (for more details see [PDJP02]) is shown in Figure 11.3. In our problem setup the dynamics of the armature current \( i_a \) can be neglected since the time constant \( T_a = L_a/R_a \) is very small compared to the sampling
time $T_s$. Therefore equation (11.1) is replaced with

$$i_a = K_a (u_a - K_v \omega_m) \quad (11.8)$$

with $K_a = 1/R_a$.

For the complete model of the throttle two more mathematical models, those of friction (11.6) and the LH nonlinearity (11.7) are needed. They are derived in the following subsections.

### 11.2.1 Static Friction Model

As mentioned before, (relatively) inexpensive mass-produced mechanical components, such as gearboxes and bearings, introduce significant friction in an electronic throttle. To model such phenomena, in this chapter we use a static Karnopp friction model [Kar85].

Depending on the motor speed the Karnopp model (see Figure 11.4) is either in the stiction or in the sliding region. To distinguish between these two cases an additional internal state variable $\omega_m^*$ is introduced

$$J \frac{d\omega_m^*}{dt} = m_{app} - m_f \quad (11.9)$$

where $\omega_m^*$ represents a fictitious motor speed and $J$ is the overall moment of inertia. Denoting with $\Delta \omega$ the angular velocity very close to zero, we say that the friction model is in the stiction region if $|\omega_m^*| \leq \Delta \omega$, otherwise it is in the sliding region

$$\omega_m = \begin{cases} 0 & \text{if } |\omega_m^*| \leq \Delta \omega, \\ \omega_m^* & \text{if } |\omega_m^*| > \Delta \omega. \end{cases} \quad (11.10)$$

In the stiction mode $m_f$ is equal to the applied torque $m_{app}$ if the applied torque is within
the region \([-M_S, M_S]\), otherwise it is saturated to the maximum value

\[
m_f = \begin{cases} 
-M_S & \text{if } |\omega_m| \leq \Delta \omega & \text{and } m_{app} < -M_S, \\
m_{app} & \text{if } |\omega_m| \leq \Delta \omega & \text{and } |m_{app}| \leq M_S, \\
M_S & \text{if } |\omega_m| \leq \Delta \omega & \text{and } m_{app} > M_S
\end{cases}
\] (11.11)

where \(M_S\) is the maximum friction torque in the stiction.

In the sliding region the friction torque is expressed as

\[
m_f = [M_C + (M_S - M_C) e^{-(\omega_m - \Delta \omega) / \omega_s}] + b|\omega_m| \text{sgn}(\omega_m)
\] (11.12)

where \(M_C\) is the Coulomb friction, \(\omega_s\) is the Striebeck angular velocity, \(\delta\) is the Striebeck coefficient, and \(b\) is the viscous friction coefficient. In this chapter, for simplicity, the Striebeck effect and the viscous friction are neglected

\[
m_f = \begin{cases} 
M_C & \text{if } \omega_m > \Delta \omega, \\
-M_C & \text{if } \omega_m < -\Delta \omega.
\end{cases}
\] (11.13)

### 11.2.2 Limp-Home Nonlinearity

Following the manufacturer specification, in the case of total power failure the throttle valve has to be placed in a so-called Limp-Home (LH) position. This position enables the driver to “limp” to the nearest repairing facility since there is some inflow of air to the car engine. The LH position is ensured with the highly nonlinear stress-strain curve of the return spring. The LH nonlinearity is shown in detail around the LH position in Fig. 11.5. Note that the LH nonlinearity is affine

\[
m_S(\theta) = \begin{cases} 
K_i (K_{S1j} \theta + K_{S0j}) & \text{if } \theta \leq \theta_{LH}, \\
K_i (K_{S12} \theta + K_{S02}) & \text{if } \theta_{LH} < \theta \leq \bar{\theta}_{LH}, \\
K_i (K_{S13} \theta + K_{S03}) & \text{if } \theta > \bar{\theta}_{LH},
\end{cases}
\] (11.14)

where \(K_{S1j}\) and \(K_{S0j}\) are coefficients of the \(j\)-th affine law, \(j \in \{1, 2, 3\}\), and \(\theta_{LH}\) and \(\bar{\theta}_{LH}\) are the respective angles where the affine law changes.

### 11.3 PWA Model of the Throttle

After obtaining the nonlinear continuous time model of the throttle system (equations (11.2)–(11.14)), the next step is to derive a discrete-time model which can be used in an optimization procedure. Several modeling frameworks have been introduced for discrete-time hybrid systems. Among them, piecewise affine (PWA) systems [Son81] are defined by partitioning the state space into polyhedral regions, and associating with each region a different affine state-update equation

\[
x(t + 1) = A_i x(t) + B_i u(t) + f_i \\
\text{if } \begin{bmatrix} x(t) \\
u(t) \end{bmatrix} \in C_i, \ i = 1, \ldots, s
\] (11.15)
where \( x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_f}, \ u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_f}, \ {C_i}^s \) is a polyhedral partition of the state+input space \( \mathbb{R}^{n+m}, \ n := n_c + n_f, \ m := m_c + m_f. \)

In our case the states are \( x_1 = \omega^* \) and \( x_2 = \theta \), and the input is the voltage \( u \). Note that we use \( \omega^* \) and not \( \omega \) as the state of the system, although \( \omega \) can be measured/estimated.

In this way the total number of PWA dynamics is reduced. Furthermore, \( \omega \) and \( \omega^* \) differ only in a small stiction region, and in practical applications this difference can be neglected.

All nonlinearities of the electronic throttle are PWA in continuous time. It is reasonable to consider the following approach when constructing the corresponding discrete-time PWA model. We fix all possible combinations of affine parts of the nonlinearities (5 from the friction model (11.11)–(11.13), and 3 from the LH model (11.14) giving 15 affine dynamics in total), and for each of those combinations we form an affine continuous time description of the throttle system. Sampling each of the continuous time affine systems with a Zero Order Hold (ZOH) gives an equivalent discrete-time representation. Combining them gives 15 discrete-time PWA dynamics, each of which is defined over a certain part of the state+input space

\[
H_i x + L_i u \leq K_i, \tag{11.16}
\]

As an example we show one such continuous time affine region with its corresponding dynamics

\[
\Delta \omega \leq \omega^* \leq \Delta \omega, \tag{11.17}
\]

\[
\theta_{LH} \leq \theta \leq \overline{\theta}_{LH}, \tag{11.18}
\]

\[
m_{app} \leq -M_S, \tag{11.19}
\]

\[
\begin{bmatrix}
-1 & 0 \\
1 & 0 \\
0 & -1 \\
0 & 1 \\
0 & -K_lK_{S12}
\end{bmatrix} x(k) +
\begin{bmatrix}
0 \\
0 \\
K_aK_{ch}K_l
\end{bmatrix} u(k) \leq
\begin{bmatrix}
\Delta \omega \\
\Delta \omega \\
\theta_{LH} \\
\overline{\theta}_{LH} \\
-M_S + K_lK_{S02}
\end{bmatrix}, \tag{11.20}
\]
11.4 Constrained Optimal Control of the Throttle

A PWA description of the throttle was obtained with the procedure explained in previous section. The sampling time was set to $T_s = 10$ ms with the following constraints on the system variables: $i_a(k) \in [-2, 2]$, $\omega_m(k) \in [-100, 100]$, $\theta(k) \in [13, 90]$ and $u(k) \in [-5, 5]$, $k = 0, \ldots, T$. Additionally, in order to make the real-time implementation easier $\theta(k)$ was constrained to $[13, 40]$ which resulted in a smaller number of control laws generated by the algorithm.

We formulate the optimization problem (10.18)–(10.20) for the discrete-time model of the throttle in the same fashion it was done for the previous application (see Section 10.5). Here we have $T = 5$, $Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0 \end{bmatrix}$, $R = 0.1$, $P = Q$, $X_f = [\begin{bmatrix} -0.2 & 24.95 \\ 0.2 & 25.05 \end{bmatrix}]$, $x_e = [0 \ 25]'$, $u_e = 1.098$. The corresponding optimal solution, depicted in Figure 11.6, is composed of 568 regions in the state-space.

### 11.4.1 Simulation Results

In Figure 11.7 we report the response of the system starting from $x(0) = [0 \ 13.74]'$ when the optimal state-feedback control law is used. More detailed simulation results are reported in [BVMP02].
11.4.2 Experimental Results

The optimal set point control scheme is applied to the real electronic throttle process. Experiments were carried out with a Pentium III 1.7 GHz machine running MATLAB® 5.3 and using Real-Time Workshop® with an A/D–D/A card used as a computer-process interface. We use the same system/optimization settings as in the simulation in Section 11.4.1, and the same resulting controller. Note that the optimal controller is applied in a receding horizon fashion with a sampling time of $T_s = 10$ ms.

Obtained experimental results are shown in Figure 11.8, and the detailed transient behavior in Figure 11.9. The starting point $x(0) = [0 13.74]'$ is reached using a step change of the control input at $t = 2.5$ s. The optimal controller action is applied to the process from $t = 5$ s onwards. However, variable $u$ represents the controller action throughout the whole experiment. Only position signal $\theta$ is available while the motor angular speed $\omega_{mrec}$ is reconstructed from the position signal using a simple differentiator.

11.5 Conclusion

In this chapter we have seen how to construct a state-feedback optimal control law for a specific hybrid system - an electronic throttle. After modeling the electronic throttle as a PieceWise Affine (PWA) system, we derived an optimal control law via dynamic programming. Results indicate that the constrained finite time optimal control of small/medium
Figure 11.8: Throttle response (experiment). Optimal controller is applied after $t = 5$ s.

sized PWA systems with fast sampling times can be successfully implemented.
Figure 11.9: Detail of the throttle response from Figure 11.8 (experiment).
12

Low Complexity Control of an
Electronic Throttle

In this chapter we extend results from the previous chapter (see also [BVMP03, VBPP04])
where the constrained finite time optimal control (CFTOC) problem for the electronic throt-
tle was solved, but only one set-point was considered. Here we address the case of the
throttle valve angle reference tracking. Unfortunately in the reference tracking case the
solution to the CFTOC problem becomes too large. Therefore we consider a modification
of the objective function, in particular the constrained time-optimal control (CTOC) prob-
lem [GKBM04] is solved with the procedure described in Chapter 8, which results with the
stabilizing, lower-complexity state-feedback controller.

This chapter is organized as follows. In Section 12.1 we derive a discrete-time piecewise
affine model of the throttle. The off-line and on-line computation procedure for the time-
optimal control of hybrid systems is presented in Section 12.2. In Section 12.3 we report
the experimental results on a real electronic throttle using the time-optimal control strategy.
Finally, the contributions are summarized in Section 12.4.

12.1 Discrete-Time Piecewise Affine Model of an
Electronic Throttle

Discrete-time PWA systems (for more details see [Son81, HSB01]) entail several state-update
equations, each defined over a polyhedron in the extended state+input space

\[ x_{k+1} = A_i x_k + B_i u_k + f_i \]
\[ y_k = C_i x_k + D_i u_k \]

if \( \begin{bmatrix} x_k \\ u_k \end{bmatrix} \in D_i \), \( i = 1, \ldots, s \) (12.1)

where \( x \in \mathbb{R}^n \) is the state, \( y \in \mathbb{R}^p \) is the output, \( u \in \mathbb{R}^m \) is the control input, \( \{D_i\}_{i=1}^s \) is
polyhedral partition of the state+input space \( \mathbb{R}^{n+m} \), and \( k \) denotes the sampling instant.

12.1.1 Electronic Throttle Model

The discrete-time PWA model is derived from the continuous-time nonlinear model of the
electronic throttle (11.2)–(11.8) (see Section 11.2).

The piecewise affine approximation of the limp-home nonlinearity is given by (11.14). In
the rest of this section we will derive the appropriate dynamic friction model.
12.1.2 Dynamic Friction Model

Probably the easiest way to model the friction effect is to describe it as a simple static characteristic. The Karnopp friction model [Kar85] is an example of a simple static friction model that allows only two working regimes: stiction and sliding. However, friction in the throttle’s gearbox is characterized by a significant presliding effect [DPP04] where the valve motion begins well before the applied torque reaches the value of the static friction torque. The presliding motion continues until the full presliding displacement (≈ 0.3°) is reached. At this point the applied torque is equal to the static friction torque. Modelling this phenomenon is crucial since our tracking system has to be accurate up to the measurement resolution (≈ 0.1°). For the valve movements in the presliding displacement range the controller that uses the static Karnopp friction model would overcompensate the friction effect. Consequently, for a given set point such a controller can exhibit oscillations in the steady state position of the throttle valve, cf. [BVMP03]. Therefore in this chapter we use a dynamic friction model. In particular we use a reset-integrator model [HF91] that is essentially modeling the friction nonlinearity (11.6) as a linear switched system of the following form (for more details see [HF91])

\[
\frac{dm_f}{dt} = \begin{cases} 
K_f \omega m & \text{if } \alpha M_C \leq |m_f| \leq M_C, \\
0 & \text{otherwise},
\end{cases}
\]

(12.2)

where \(M_C\) denotes the static Coulomb friction and the coefficient \(\alpha\), with \(0 \leq \alpha < 1\), defines the friction torque at which presliding movement begins\(^1\). Note that the reset-integrator model (12.2) is well suited for the derivation of a discrete-time PWA friction model which is not necessarily the case with other dynamic friction models. For example, the LuGre friction model [dWOsL95] can accurately describe presliding effect but it is governed by a nonlinear differential equation that is difficult to linearize.

The choice of a continuous-time reset-integrator friction model [HF91] is just a first step in deriving a PWA friction model in the form (12.1). All continuous-time linear dynamics of the friction model (12.2) are sampled with ZOH (with the same sampling time) to obtain their corresponding discrete-time representations. Consider now the discrete-time PWA system that comprises only in such fashion generated affine dynamics and compare it to the original continuous-time nonlinear system. Even though ZOH is used for each sub-system there is no guarantee that the output of a discrete-time and continuous-time systems will not differ at sampling instances (unlike in the linear systems case). This discrepancy is caused by the fixed sampling time of a discrete-time model (12.1) which implies that the switching between two dynamics can happen only at multiples of a sampling time. On the other hand, in the continuous-time model (12.2) switching between two modes can happen at any moment. Thus, if a sampling time is too big, it could happen that the state update equation \(x_{k+1} = A_i x_k + B_i u_k + f_i\) “changes” the state too much. Unfortunately, in our application the above mentioned effect cannot be neglected since the sampling time has to be large enough to predict the whole transient response of the system. We cope with this problem by introducing additional affine dynamics in the PWA model (12.1) as follows.

\(^1\)In the throttle case \(\alpha = 0.05\). It is beneficial for \(\alpha\) to be small since this makes the control action smoother but there is no need for its experimental identification.
It is critical to accurately model the friction in the PWA form around zero-velocity \((\omega_m = 0)\) since at that point affine models obtained by discretizing linear subsystems of (12.2) with ZOH differ drastically. We predict the \(\omega_m\)-zero-crossing in the PWA model in order to use different affine dynamics in the zero-crossing and non-zero-crossing case. This is achieved by introducing additional partitioning in the state-input space, defined by the borders of \(\omega_{m,k+1}^i = 0\), where \(\omega_{m,k+1}^i\) denotes the one-step ahead prediction of \(\omega_m\) at the discrete instant \(k\) using affine dynamics indexed with \(i\). The borders of the friction model in \(\omega_m - m_f\) plane are depicted in Fig. 12.1, where \(\varepsilon\) denotes a small positive number. Note that the borders introduced by the one-step-ahead prediction of \(\omega_m\) are not depicted, since they involve all the states and inputs and cannot be presented in 2D. Note also, that this additional partitioning happens only in the presliding and sliding dynamics, i.e. when the valve moves.

\[
\begin{align*}
\omega_{m,k+1}^i &= K_m m_{\text{app},k}, \\
\theta_{k+1} &= \theta_k + \beta \frac{180K_i T}{\pi} \frac{1}{2} \omega_{m,k}, \\
m_{f,k+1} &= m_{\text{app},k},
\end{align*}
\]

Figure 12.1: Borders for the friction model in the \(\omega_m - m_f\) plane (thick lines).

The linearization of the reset-integrator friction model in sliding and presliding regimes, when zero-crossing of \(\omega_m\) is not detected, is straightforward. In the case of stiction or \(\omega_m\) zero-crossing we use the following state-update

\[
\begin{align*}
\omega_{m,k+1} &= K_m m_{\text{app},k}, \\
\theta_{k+1} &= \theta_k + \beta \frac{180K_i T}{\pi} \frac{1}{2} \omega_{m,k}, \\
m_{f,k+1} &= m_{\text{app},k},
\end{align*}
\]
where \( K_m = 1 \, \text{rad Nms} \), so that the torques of around 0.005 Nm observed in practice produce a very small speed for stiction. We use Equation (12.3) for the \( \omega_m \)-update because resetting \( \omega_m \) to the value 0 would cause problems at the next time step since such a state would belong to more than one region \( D_i \) in (12.1). Coefficient \( \beta \), experimentally determined at value 0.7, is used to model the change of the valve position when entering the stiction regime.

### 12.1.3 Discrete-Time PWA Model of an Electronic Throttle

The discrete-time PWA model (12.1), with \( m = 1 \), \( n = 3 \), \( p = 1 \), the state vector \( x = [\omega_m \, \theta \, m_f]^T \) and the output \( y = \theta \), is constructed by combining each of the affine models of friction with each of the affine models of the LH nonlinearity. The PWA model of the electronic throttle consists of 30 affine dynamics. For numerical reasons, the internal friction torque state \( m_f \) is scaled to fall within the order of magnitude of 100.

### 12.2 Time-Optimal Electronic Throttle Reference Tracking

#### 12.2.1 Extended System and Constraints

The optimal reference tracking problem for the throttle is defined in the augmented state+input space

\[
\bar{x}_k = \begin{bmatrix} x_k \\ u_{k-1} \\ r_k \end{bmatrix}, \quad \bar{u}_k = u_k - u_{k-1}, \quad (12.6)
\]

where \( r_k \in \mathbb{R}^p \) denotes the reference that the system output \( y_k \) should follow, and \( \bar{u}_k \) is the change of a controller action, which is, conveniently, equal to zero at steady-state for any value of the reference. The extended state-space is \( \bar{X} \subset \mathbb{R}^5 \). The output \( \bar{y}_k \) of such an extended system is the tracking error itself:

\[
\bar{y}_k = r_k - y_k. \quad (12.8)
\]

The extended PWA system can be constructed from (12.1) by using Equations (12.6)–(12.8) and \( r_k = \text{const.} \), i.e.

\[
\bar{x}_{k+1} = \bar{A}_i \bar{x}_k + \bar{B}_i \bar{u}_k + \bar{f}_i, \quad (12.9a)
\]

\[
\bar{y}_k = \bar{C}_i \bar{x}_k \quad (12.9b)
\]

if \( [\bar{x}_k \, \bar{u}_k]^T \in \bar{D}_i, \quad i = 1, \ldots, s. \) \quad (12.9c)

In the following we will denote Equations (12.9a)–(12.9c) with the shorter form

\[
\bar{x}_{k+1} = \bar{f}_{\text{PWA}}(\bar{x}_k, \bar{u}_k). \quad (12.10)
\]
Constraints on the throttle states and input are introduced to preserve outwearing of the plant

$$C^x \bar{x}_k + C^u \bar{u}_k \leq C^c,$$  \hspace{1cm} (12.11)

where $C^x$, $C^u$, and $C^c$ are properly dimensioned matrices. Using the algorithm presented here, it is guaranteed that those constraints are fulfilled. In our experimental setup, physically possible throttle angles are in the range of 12.8° to 103.4° with a LH-position at 20.0° and 0.2° symmetrical LH-area around it. The constraints used in the model are the same as the physical constraints as we don’t want the valve to hit the mechanical stops. The same constraints are used for the reference $r_k$. The angular velocity on the motor side $\omega_m$ is constrained between $-150 \text{ rad/s}$ and $150 \text{ rad/s}$, thus enabling maximal change of the angle of $\pm 2.5^\circ$ in 5 ms = $\pm 500^\circ$ s. The input signal $u_k$ is physically limited to $\pm 5$ V, due to D/A-card and chopper. The same constraint is used for $\bar{u}$. To extend the working life of the armature circuit the armature current is limited to $\pm 2$ A.

### 12.2.2 Off-Line Computation

The main idea of the time-optimal algorithm is to steer the system states as quickly as possible, while fulfilling the constraints, to the subset of the state-space where a control law exists that makes the closed-loop system asymptotically stable and invariant. This asymptotically stable, invariant subset is denoted with $\mathcal{X}^I$ and it is defined as follows (for more details see Chapter 8 and [RGK+04, GKB04])

$$\mathcal{X}^I = \{ \bar{x}_0 \mid \bar{x}_k \in \mathcal{X}^I, C^x \bar{x}_k + C^u u_I(\bar{x}_k) \leq C^c, \bar{x}_{k+1} = f_{\text{PWA}}(\bar{x}_k, u_I(\bar{x}_k)), \forall k > 0 \},$$ \hspace{1cm} (12.12)

The invariant set is computed in two steps: firstly, a piecewise affine control law $u_I(\bar{x})$ is computed, and then the set $\mathcal{X}^I$ is constructed with the iterative procedure described in [RGK+04]. The computation of the control law $u_I$ in the throttle case is hard because the affine dynamics that describe stiction are non-controllable. Thus, neither LQR nor any other optimal control strategy for affine systems can be used to derive the control law for them.

We define a tiny polytopic region around the hyperplane $r = y, \omega_m = 0$, which we refer to as the tracking origin

$$T = \{ \bar{x} \in \mathbb{R}^5 : |\omega_m| < 0.1, |r - \theta| < 0.01 \}.$$ \hspace{1cm} (12.13)

We find the controller $u_I(\bar{x})$ using a one-step reachability analysis with $T$ as the target region. For each controllable dynamics a multi-parametric quadratic program (mp-QP) is solved to find a PWA control law and polyhedral partition. Since for non-controllable dynamics this procedure is inapplicable we define an ad-hoc affine control law that resets $m_{\text{app}}$ to 0 at the current time instant, and then use polytopic algebra to find the polytopic domain of this law. The obtained polytopic regions with their corresponding affine control laws, form the initial set for the iterative computation of the invariant set [RGK+04]. In our application the invariant set consists of approximately 50 polytopes in $\mathbb{R}^5$. Asymptotic stability of the
obtained invariant set has not yet been tested. Minimum-time control problem with known invariant set $\mathcal{X}^I$ is posed as follows:

$$J^*(\bar{x}_0) = \min_{U_k,k} \kappa$$

subject to

$$\begin{align*}
\bar{x}_{k'} + 1 &= f_{\text{PWA}}(\bar{x}_{k'}, \bar{u}_{k'}) , \\
C^\alpha \bar{x}_{k'} + C^u \bar{u}_{k'} &\leq C^c , \\
\bar{x}_k &\in \mathcal{X}^I , \\
0 &\leq k' \leq k ,
\end{align*}$$

(12.14)

where $U_k = \{\bar{u}_0, ..., \bar{u}_{k-1}\}$. The solution to this problem using the dynamic programming approach is described in Chapter 8. The cost-to-go is equal to the number of time steps to reach the invariant set, which significantly reduces the offline computation time and the controller complexity compared to the piecewise linear or piecewise quadratic cost-to-go functions used in e.g. [BVMP03]. The algorithm to solve (12.14) is as follows: at the first iteration step ($z = 1$) the invariant set consist of the convex hulls $R_{0,i,k}$ formed at the previous iteration step ($z = 0$)

$$\mathcal{X}^I = \bigcup_{1 \leq i \leq s} \bigcup_{1 \leq k \leq k_{i,0}} R_{0,i,k}$$

(12.15)

where $k_{i,z}$ denotes the number of convex hulls attached to dynamics $i$ in iteration $z$. Using all dynamics $1 \leq j \leq s$, we try to enter into any of the convex hulls $R_{i,k-1}$ in one step, while satisfying the imposed constraints. This one-step reachability analysis can be solved using multi-parametric program. We state it as an mp-QP

$$V(\bar{x}_0) = \min_{\bar{u}_0} \begin{bmatrix} \bar{y}_1^T Q \bar{y}_1 + \bar{y}_0^T Q \bar{y}_0 + \bar{u}_0^T R \bar{u}_0 \\
C^\alpha \bar{x}_0 + C^u \bar{u}_0 &\leq C^c , \\
A_j \bar{x}_0 + B_j \bar{u}_0 + f_j &\in R_{i,k-1} \\
\bar{y}_0 &= C_j \bar{x}_0 , \\
\bar{y}_1 &= C_i \bar{x}_1.
\end{bmatrix}$$

(12.16)

For each triplet $(j, i, k)$, $1 \leq k \leq k_{i,z-1}$, we find a convex feasibility region (convex hull) $R$ by solving (12.16). Convex hull $R$ contains several polytopes with an affine control law over each polytope. If it is totally covered with previously generated hulls or empty (in case of an infeasible mp-QP), we discard it. Otherwise, it is kept and denoted with $R_{j,k}^z$, where $k$ successively grows as new hulls in dynamics $j$ in step $z$ are added. After all the transitions are made, we construct a one-cost-to-go set of all kept regions

$$\mathcal{X}^I = \bigcup_{1 \leq i \leq s} \bigcup_{1 \leq k \leq k_{i,1}} R_{i,k}^1$$

(12.17)

The iteration step $z$ is increased to 2 and the whole computation is repeated. This procedure is algorithm-wise presented in Chapter 8, with a simple proof of the theorem that states asymptotical stability for the obtained minimum-time controller. Algorithm that minimizes the switchings between dynamics through the predicted trajectory by sacrificing the minimum-time property is also presented in that paper. We propose a mix of those two
approaches: reduced switching whilst keeping the minimum-time property. It is achieved by ordering the dynamics indices $j$ and $i$ in triplets $(j, i, k)$ for each iteration $z$, in such a way that transitions with $j = i$ are processed first. Additionally, indices $k$ are ordered in such a way that hulls $R_{i,k}^{-1}$ with a bigger inner Chebyshev radius come first. Since larger target regions create larger hulls the smaller hulls created at later stage are more likely to be covered and discarded.

The off-line computation stops when at some iteration $z_m + 1$ no new convex feasibility regions are found. Then all the sets $\mathcal{X}^z$, $1 \leq z \leq z_m$, together with $\mathcal{X}^t$ form maximal controllable set [RGK+04]

$$\mathcal{K}_{\text{PWA}}^\infty = \mathcal{X}^t \cup \bigcup_{1 \leq z \leq z_m} \mathcal{X}^z \subset \bar{\mathcal{X}}.$$ (12.18)

The maximal controllable set $\mathcal{K}_{\text{PWA}}^\infty$ contains polytopes and the associated affine control laws, sorted by the cost-to-go penalty, and within the same penalty sorted by the dynamics.

12.2.3 On-Line Computation

On-line computation of the control action is very simple. For a given $\bar{x}$ we define the set of active dynamics

$$\mathcal{A}(\bar{x}) = \{ i \mid 1 \leq i \leq s, \bar{x} \in \bar{P}_i \},$$ (12.19)

where $\bar{P}_i$ denotes the projection of $\bar{D}_i$ on $\bar{X}$ space. Starting from an invariant (zero-cost-to-go) set towards the higher cost-to-go sets we search for a first polytope among active dynamics that contains the state $\bar{x}$ and compute the control input $\bar{u}$ by using the affine control law corresponding to that polytope.

12.3 Experimental Results

The derived time-optimal control strategy is tested on a real electronic throttle with a computer system running Real-Time Linux. Since the MPC strategy requires all system states to be known, the unmeasured angular velocity $\omega_m$ and internal friction variable $m_f$ are estimated using the Unscented Kalman Filter (UKF). For details on the estimator implementation see in [VPP03]. The on-line control scheme is depicted in Fig. 12.2.

We introduce a prefilter (with a 15 ms time constant) in the reference path in order to eliminate an overshoot in the angle response and to reduce the number of the cost-to-go sets around the invariant set needed for the on-line computation. Namely, only three cost-to-go sets around the invariant set are used in on-line control input computation and for the abrupt reference changes the extended state could fall outside the space covered with those sets. Overall, there are 17780 polyhedral regions in $\mathbb{R}^5$ with an affine control law in each region. For storage of such a controller structure around 5 MB of RAM is needed. In the subsequent Figures: Fig. 12.3, Fig. 12.4, and Fig. 12.5 we report the experimental results. Very good performance of the control system is visible: fast transients (30 ms for all reference steps less than $2^\circ$ - more than two times faster than by a tuned PID controller with a feedforward compensation of nonlinearities [DPP+04]), small overshoot or no overshoot at
all, fast convergence to the one-bit limit cycle around the given reference\(^2\) (comparable to nonlinear PID).

\(\text{Figure 12.2: On-line control scheme.}\)

\(\text{Figure 12.3: Response to the ramp reference through LH area.}\)

\(^2\)Static accuracy within the measurement resolution.
12.4 Conclusion

A strong friction effect, nonlinear return spring characteristics and high demands on the overall vehicle performance make the control of an electronic throttle a challenging task. In this chapter we derive a time-optimal model predictive control strategy for the electronic throttle. Our strategy guarantees (i) time-optimal reference tracking and (ii) satisfaction of all constraints imposed on the process states and inputs. The electronic throttle is modeled as a discrete-time piecewise affine (PWA) system. Starting from a continuous-time reset-integrator friction model, we derive a discrete-time friction model that can capture the presliding effect and is suitable for the model predictive control strategy. We outline the procedure for an off-line computation of the optimal control law for the range of system states and references. The procedure is using recently reported results in time-optimal control of hybrid systems. We propose slight modifications that reduce the switching between dynamics of a PWA model along the predicted transient whilst preserving the time-optimality. The optimal control law is PWA over polytopes and thus simply implementable on-line. We report the experimental results on a real electronic throttle and compare them with a tuned PID controller with a feedforward compensation of nonlinearities. Our controller achieves more than two times faster transients, while preserving one-bit limit cycle phenomenon and having no overshoot.

Figure 12.4: Response to the square reference of $2^\circ$. 
Figure 12.5: Response to the square reference of 0.2°.
Part IV

SOFTWARE
13

The MPT Toolbox

The Multi-Parametric Toolbox (MPT) is a Matlab toolbox for computing explicit optimal or sub-optimal feedback control laws for constrained linear and piecewise affine systems. The toolbox offers a broad spectrum of algorithms compiled in a user friendly and accessible format: starting from different performance objectives (linear, quadratic, minimum time) to the handling of systems with persistent additive and polytopic uncertainties. The algorithms included in the toolbox are a collection of results from recent publications in the field of constrained optimal control of linear and piecewise affine systems. The MPT toolbox is available from

http://control.ee.ethz.ch/~mpt

and is being updated on a regular basis. For an in-depth introduction to MPT, we refer the reader to the MPT web-page, where the original paper [KGBM04] and a detailed latest version of software manual [KGBC04] is available for download.

We would like to thank all contributors to the toolbox which are not on the author list [KGBM04]. Specifically (in alphabetical order): Miroslav Barić, Alberto Bemporad, Pratik Biswas, Francesco Borrelli, Raphael Cagienard, Frank J. Christoffersen, Tobias Geyer, Eric Kerrigan, Arne Linder, Marcel Leutenegger, Marco Lüthi, Saša V. Raković, Raphael Suard, Fabio D. Torrisi, Kari Unneland and Mario Vašak. A special thanks goes to Komei Fukuda (cdd), Johan Löfberg (Yalmip) and Colin Jones (ESP) for allowing us to include their respective packages in the distribution. Thanks to their help we are able to say that MPT truly is an 'unpack-and-use' toolbox.

13.1 Overview of The Toolbox

Optimal control of constrained linear and piecewise affine (PWA) systems has garnered great interest in the research community due to the relative ease with which complex problems can be stated and solved. The aim of the Multi-Parametric Toolbox (MPT) is to provide an efficient computational means of obtaining feedback controllers for these types of constrained optimal control problems. The MPT toolbox is implemented in a MATLAB [The03] programming environment and consists of three software-blocks:

- Polytope Library
The MPT Toolbox

- Multi-Parametric Programming Solvers
- Computation of Feedback Controllers for Constrained Systems

Specifically, the toolbox contains efficient implementations of all polytope- and P-collection-operations described in Chapter 3. In addition, mp-LP and mp-QP solvers are provided which are applied in various feedback controller computation schemes contained in MPT. In short: MPT contains a large part of all algorithms which were developed at the Automatic Control Laboratory (ETH Zürich) during the last two years as well as a plethora of standard functions which are often needed in the context of controller computation for constrained systems.

Furthermore, the MPT software package includes several state of the art solvers (CDD [Fuk97], ESP [Jon04], SeDuMi [Stu99], Yalmip [Löf00]) such that the toolbox is truly unpack and use. In addition to these freeware solvers which are distributed as part of MPT, additional solvers are also supported. Namely MATLAB's linprog and quadprog, the LP and QP solvers of the Numerical Algorithms Group (NAG) [Num02] as well as the CPLEX [ILO03] and GLPK [Mak01] solvers. If the user wishes to utilize another LP or QP solver, a simple modification to the functions mpt_solveLP and mpt_solveQP will do.

Aside from the strong functionality of MPT, a lot of work has gone into making it easily accessible, so that people with little background in control will (hopefully) be able to apply it without too many difficulties.

References

Aside from standard polytopic-manipulation functions, the toolbox contains several state-of-the-art algorithms, some of which are given in the following list:

- Set difference computation [BT03] and convexity recognition of the union of polyhedra [BFT01].
- Multi-parametric solvers for control of constrained linear [Bao02, GBTM03, GBTM04] and PWA [BBBM03a, BCM03a, BCM03b] systems.
- Invariant Set computation of linear [GT91] and PWA systems [GPM03].
- Computation of invariant sets for linear [KG98] and PWA systems [RGK+04] subject to bounded disturbances.
- Stability analysis of PWA systems through LMIs [GLPM03] or LPs [GKBM05].
- Robust stability analysis of PWA systems through LMIs [GPM03].
- Multi-parametric computation of robust feedback controllers for constrained linear systems [BBM03b].
- Multi-parametric computation of low complexity controllers for constrained linear [GM03] and PWA [GKBM04] systems.
13.1.1 Classes and Basic Polytope Manipulations

\[
P = \text{polytope}(P_x, P_c) \quad \text{Constructor for creating the polytope}
\]
\[
P = \{ x \in \mathbb{R}^n \mid P_x x \leq P_c \};
\]
\[
\text{double}(P) \quad \text{Access internal data of the polytope, e.g. } [P_x, P_c] = \text{double}(P);
\]
\[
\text{display}(P) \quad \text{Displays details about the polytope } \mathcal{P};
\]
\[
\text{nx} = \text{dimension}(P) \quad \text{Returns dimension of a given polytope } \mathcal{P};
\]
\[
\text{nc} = \text{nconstr}(P) \quad \text{For a polytope } \mathcal{P} = \{ x \in \mathbb{R}^n \mid P_x x \leq P_c \} \text{ returns number of constraints defining } \mathcal{P} \text{ (i.e. number of rows of the } P_x \text{ matrix)};
\]
\[
[~,~] \quad \text{Horizontal concatenation of polytopes into an array, e.g. } PA = [P_1, P_2, P_3];
\]
\[
( ) \quad \text{Subscripting operator for } P\text{-collections, e.g. } PA(i) \text{ returns the } i\text{-th polytope in } PA;
\]
\[
\text{length}(PA) \quad \text{Returns number of elements in a } P\text{-collection } PA;
\]
\[
\text{end} \quad \text{Indexing function which returns the final element of a } P\text{-collection};
\]
\[
P == Q \quad \text{Check if two polytopes are equal } (P = Q);
\]
\[
P ~== Q \quad \text{Check if two polytopes are not-equal } (P \neq Q);
\]
\[
P >= Q \quad \text{Check if } P \supseteq Q;
\]
\[
P <= Q \quad \text{Check if } P \subseteq Q;
\]
\[
P > Q \quad \text{Check if } P \supset Q;
\]
\[
P < Q \quad \text{Check if } P \subset Q;
\]
\[
P & Q \quad \text{Intersection of two polytopes, } P \cap Q;
\]
\[
P \mid Q \quad \text{Union of two polytopes, } P \cup Q.
\]
\[
\text{If the union is convex, the polytope } P \cup Q \text{ is returned, otherwise the } P\text{-collection (polytope array) } [P \ Q] \text{ is returned;}
\]
\[
P + Q \quad \text{Minkowski sum, } P \oplus Q;
\]
\[
P - Q \quad \text{Pontryagin difference, } P \ominus Q;
\]
\[
P \setminus Q \quad \text{Set difference operator. Works with polytopes and } P\text{-collections;}
\]

Table 13.1: Short overview of overloaded operators for the class polytope.

The toolbox defines a new class polytope inside the MATLAB programming environment along with overloaded operators which are presented in Table 13.1. The functions for polytope manipulations are given in Table 13.2.

Note that MPT does not handle polyhedral sets and is designed for use with bounded sets only. All functions take either polytopes or P-collections as an input argument which is illustrated in the following example:

Example 13.1.

\[
>> P = \text{polytope}([\text{eye}(2); -\text{eye}(2)], [1 1 1 1]'); \quad \% \text{Create Polytope } P
\]
The MPT Toolbox

- `B=bounding_box(P)` Computes minimal hyper-rectangle containing a polytope $\mathcal{P}$;
- `[c,r]=chebyball(P)` Returns center $c$ and radius $r$ of the Chebychev ball inside $\mathcal{P}$;
- `V=extreme(P)` Computes extreme points (vertices) of a polytope $\mathcal{P}$;
- `E=envelope(P,Q)` Computes envelope $\mathcal{E}$ of two polytopes $\mathcal{P}$ and $\mathcal{Q}$;
- `[c,r]=facetcircle(P,i)` Computes the center $c$ and radius $r$ of the largest lower dimensional ball inside facet $i$ of polytope $\mathcal{P}$;
- `P=hull(PA)` Computes hull of a $\mathcal{P}$-collection $\mathcal{P}_A$;
- `P=hull(V)` or hull of an array of vertices $V$;
- `bool=isfulldim(P)` Checks if polytope $\mathcal{P}$ is full-dimensional;
- `bool=isinside(P,x)` Checks if $x \in \mathcal{P}$. Also works for $\mathcal{P}$-collections.
- `plot(P)` Plots a given polytope or $\mathcal{P}$-collection in 2D or 3D;
- `P=range(Q,A,f)` Affine transformation of a polytope $\mathcal{Q}$
  $$\mathcal{P} = \{ Ax + f \in \mathbb{R}^n \mid x \in \mathcal{Q} \};$$
- `P=domain(Q,A,f)` Compute polytope that is mapped to $\mathcal{Q}$
  $$\mathcal{P} = \{ x \in \mathbb{R}^n \mid Ax + f \in \mathcal{Q} \};$$

<table>
<thead>
<tr>
<th>Table 13.2: Functions defined for class polytope.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&gt;&gt; [r,c]=chebyball(P)</code></td>
</tr>
<tr>
<td><code>r=[0 0]'</code></td>
</tr>
<tr>
<td><code>c=1</code></td>
</tr>
<tr>
<td><code>&gt;&gt; W=polytope([eye(2);-eye(2)],0.1*[1 1 1 1]')</code>;</td>
</tr>
<tr>
<td><code>&gt;&gt; DIF=P-W;</code></td>
</tr>
<tr>
<td><code>&gt;&gt; ADD=P+W;</code></td>
</tr>
<tr>
<td><code>&gt;&gt; plot(ADD, P, DIF, W);</code></td>
</tr>
</tbody>
</table>

The resulting plot is depicted in Figure 13.1. When a polytope object is created, the constructor automatically normalizes its representation and removes all redundant constraints. Note that all elements of the polytope class are private and can only be accessed as described in the tables. Furthermore, all information on a polytope is stored in the internal polytope structure. More functions on polytopes are given in Table 13.2 and are illustrated in the following example.
**Example 13.2.**

```matlab
>> P=polytope([eye(2);-eye(2)],[1 1 1 1]'); %Create Polytope P
>> Q=polytope([eye(2);-eye(2)],0.1*[1 1 1 1]'); %Create Polytope Q
>> D=P\Q; %Compute set difference between P and Q
>> length(D) %D is a P-collection with 4 elements
    ans=4
>> U=D|Q; %Compute union of D and Q
>> length(U) %Union is again a polytope
    ans=1
>> U==P %Check if two polytopes are equal
    ans=1
```

(a) The sets $P$ and $Q$ in Example 13.2.

(b) The sets $P \setminus Q$ in Example 13.2.

The polytopes $P$ and $Q$ are depicted in Figure 13.1.1. The following will illustrate the `hull` and `extreme` functions.

**Example 13.3.**

```matlab
>> P=polytope([eye(2);-eye(2)],[0 1 1 1]'); %Create Polytope P
>> Q=polytope([eye(2);-eye(2)],[1 1 0 1]'); %Create Polytope Q
>> VP=extreme(P); %Compute extreme vertices of P
>> VQ=extreme(Q); %Compute extreme vertices of Q
>> D1=hull([P Q]); %Create convex Hull of P and Q
>> D2=hull([VP;VQ]); %Create convex Hull of vertices VP and VQ
>> D1==D2 %Check if hulls are equal
    ans=1
```

The `hull` function is overloaded such that it takes both elements of the `polytope` class as well as matrices of points as input arguments.
13.1.2 Control Functions

This section will give a brief overview of the main control functions which are provided with the MPT toolbox. All functions may be called by using the accessor function

\[
[\text{ctrlStruct}] = \text{mpt\_Control(} \text{sysStruct,probStruct,Options})
\]

which takes the structures defined in Chapter 13.3 as parameters and automatically calls one of the functions described below depending on the parameters which were passed. Every function returns a P-collection which contains the regions over which the feedback law is unique, i.e. \( u = F_i x + G_i \) if \( x \in P_i \). The different functions for obtaining these solutions are:

\[
[\text{ctrlStruct}] = \text{mpt\_optControl(} \text{sysStruct,probStruct,Options})
\]

This function solves a constrained optimal control problem as defined in (8.3) for linear and quadratic cost objectives for linear systems with the method proposed in [Bao02, BBM00b].

\[
[\text{ctrlStruct}] = \text{mpt\_optControlPWA(} \text{sysStruct,probStruct,Options})
\]

Calculates a solution to the Constrained Finite-Time Optimal Control problem for PWA systems. The cost objective must be linear [BCM03a].

\[
[\text{ctrlStruct}] = \text{mpt\_optInfControl(} \text{sysStruct,probStruct,Options})
\]

This function computes a solution to the constrained infinite-time optimal control problem as defined in for quadratic cost objective for linear systems using an algorithm described in [GBTM03, GBTM04].

\[
[\text{ctrlStruct}] = \text{mpt\_optInfControlPWA(} \text{sysStruct,probStruct,Options})
\]

Solution to the Constrained Infinite Time Optimal Control problem for PWA systems. The cost objective must be linear [BCM03b].

\[
[\text{ctrlStruct}] = \text{mpt\_iterative(} \text{sysStruct,probStruct,Options})
\]

This function applies the minimum time computation scheme described in [GM03]. The function returns a stabilizing minimum time controller. The controller partition obtained at the final iteration is also returned as a part of the \text{ctrlStruct} structure. The final controller guarantees feasibility for all time, though stability should still be checked with \text{mpt\_getPWQLyapFunction} (see Section 13.1.3). This scheme can also be used to obtain robust solutions by setting the appropriate flags [GPM03].

\[
[\text{ctrlStruct}] = \text{mpt\_iterativePWA(} \text{sysStruct,probStruct,Options})
\]

This function applies the minimum time or low-complexity computation scheme described in [GKBM04]. The function returns the stabilizing controller. This scheme can also be used to obtain robust solutions for PWA systems affected by additive disturbance by setting the appropriate flags [KM02].

As mentioned before, the solution to an optimal control problem is obtained by a call to the \text{mpt\_control} function. This function takes the system and problem description as
input arguments and calls one of the functions above to calculate the state feedback controller. There is no need to call the individual functions directly. The controller structure `ctrlStruct` encompasses the control law $u = F_i x + G_i$ as well as the polyhedral partition $\{P_i\}_{i=1}^{NP}$ over which this PWA control law is defined. Consult Chapter 13.3 for a detailed description of the mentioned structures.

### 13.1.3 Analysis Functions

Various scripts which serve to plot the obtained results as well as analysis functions are included in the toolbox (e.g., [GLPM03]). Some of these function are vital for obtaining low complexity controllers [GM03,GPM03,GKBM04]. The corresponding functions are given in the following Table:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpt_getPWLyapFct</code></td>
<td>Computes a PWQ Lyapunov function for a given closed-loop system;</td>
</tr>
<tr>
<td><code>mpt_getCommonLyapFct</code></td>
<td>Computes a common quadratic Lyapunov function for a set of linear systems;</td>
</tr>
<tr>
<td><code>mpt_infset</code></td>
<td>Calculates the maximal (robust) positively invariant set for an LTI system;</td>
</tr>
<tr>
<td><code>mpt_infsetPWA</code></td>
<td>Computes the maximal (robust) positive invariant subset for PWA systems;</td>
</tr>
<tr>
<td><code>mpt_plotPartition</code></td>
<td>Plots controller partitions obtained with mpt_control. The output of this function usually looks better than the standard <code>plot</code>;</td>
</tr>
<tr>
<td><code>mpt_plotTrajectory</code></td>
<td>Graphical interface for plotting closed-loop trajectories in state-space;</td>
</tr>
<tr>
<td><code>mpt_plotTimeTrajectory</code></td>
<td>Plots closed-loop trajectories of states, inputs and outputs as a function of time;</td>
</tr>
<tr>
<td><code>mpt_plotU</code></td>
<td>For a given explicit controller, plots value of the control input over a given polyhedral state-space partition;</td>
</tr>
<tr>
<td><code>mpt_plotPWA</code></td>
<td>Plots a PWA function in 3D;</td>
</tr>
<tr>
<td><code>mpt_plotPWQ</code></td>
<td>Plots a PWQ function in 3D;</td>
</tr>
<tr>
<td><code>mpt_plotArrangement</code></td>
<td>Plots a hyperplane arrangement of a polytope in $\mathcal{H}$-representation;</td>
</tr>
</tbody>
</table>

### 13.2 Polytope Library

As already mentioned in Chapter 3, a polytope is a convex bounded set which can be represented either as an intersection of a finite number of half-spaces ($\mathcal{H}$-representation) or as a convex hull of vertices ($\mathcal{V}$-representation). Both ways of defining a polytope are allowed in MPT and you can switch from one representation to the other one. However, by default all polytopes are generated in $\mathcal{H}$-representation only to avoid unnecessary computation.
13.2.1 Creating a polytope

A polytope in MPT is created by a call to the polytope constructor as follows:

\[ P = \text{polytope}(P_x, P_c) \]

creates a polytope by providing its \( \mathcal{H} \)-representation, i.e. the matrices \( P_x \) and \( P_c \) which form the polytope \( P = \{ x \in \mathbb{R}^n \mid P_x x \leq P_c \} \). If input matrices define some redundant constraints, these will be automatically removed to form a minimal representation of the polytope. In addition, center and diameter of the Chebychev ball (3.10) are computed as well and the \( \mathcal{H} \)-representation is normalized to avoid numerical problems. The constructor then returns a polytope object. A polytope can also be defined by its vertices:

\[ P = \text{polytope}(V) \]

where \( V \) is a matrix which contains vertices of the polytope in the following format:

\[
V = \begin{bmatrix}
v_{1,1} & \cdots & v_{1,n} \\
\vdots & \ddots & \vdots \\
v_{k,1} & \cdots & v_{k,n}
\end{bmatrix}
\]  \hspace{1cm} (13.1)

where \( k \) is the total number of vertices and \( n \) is the state dimension. Hence vertices are stored row-wise. Before the polytope object is created, \( \mathcal{V} \)-representation is first converted to half-space description by eliminating all points from \( V \) which are not extreme points. The convex hull of the remaining points is then computed to obtain the corresponding \( \mathcal{H} \)-representation. Extreme points will be stored in the polytope object and can be returned upon request without additional computational effort.

13.2.2 Accessing data stored in a polytope object

Each \texttt{polytope} object is internally represented as a structure, but because of the Object-Oriented approach, this information cannot be directly obtained by using structure referencing through the . (dot) operator. Special functions have to be called in order to retrieve individual fields.

To access the \( \mathcal{H} \)-representation of the polytope \( P = \{ x \in \mathbb{R}^n \mid P_x x \leq P_c \} \), one has to use the command \texttt{double} as described below.

\[
[P_x, P_c] = \text{double}(P)
\]

The center and radius of the Chebyshev ball can be obtained by:

\[
[x_{\text{Cheb}}, R_{\text{Cheb}}] = \text{chebyball}(P)
\]

The polytope is bounded if

\[
\text{flag} = \text{isbounded}(P)
\]
returns 1 as the output. Dimension of a polytope can be obtained by
\[ d = \text{dimension}(P) \]
and
\[ nc = \text{nconstr}(P) \]
will return number of constraints (i.e. number of half-spaces) defining the given polytope \( P \).
The vertex representation of a polytope can be obtained by:
\[ V = \text{extreme}(P) \]
which returns vertices stored row-wise in the matrix \( V \). As enumeration of extreme vertices is an expensive operation, the computed vertices can be stored in the polytope object. To do this, the function must be called as
\[ [V, R, P] = \text{extreme}(P) \]
which returns extreme points \( V \), extreme rays \( R \) and the update polytope object with vertices stored inside \( P \). Since polytopes are bounded sets (i.e. there are no unbounded directions) the matrix \( R \) should be empty. To check if a given point \( x \) lies in a polytope \( P \), use the following call:
\[ \text{flag} = \text{isinside}(P, x) \]
The function returns 1 if \( x \in P \), 0 otherwise. If \( P \) is a P-collection (see Definition 3.7), the function call can be extended to provide additional information:
\[ [\text{flag}, \text{inwhich}, \text{closest}] = \text{isinside}(P, x) \]
which returns a 1/0 flag which denotes if the given point \( x \) belongs to any polytope of a P-collection \( P \). If the given point lies in more than one polytope, \( \text{inwhich} \) contains indices of the regions which contain \( x \). If there is no such region, the index of the region which is closest to the given point \( x \) is returned in \( \text{closest} \). A more detailed (though not complete) overview of the polytope library is given in Table 13.3.

13.2.3 P-collections

polytope objects can be concatenated into arrays and it does not matter if the elements are stored row-wise or column-wise. A P-collection is created using standard Matlab concatenation operators \([,] \), e.g. \( A = [B \ C \ D] \).

It does not matter whether the concatenated elements are single polytopes or P-collections. To illustrate this, assume we’ve defined polytopes \( P_1, P_2, P_3, P_4, P_5 \) and P-collections: \( A = [P_1 \ P_2] \) and \( B = [P_3 \ P_4 \ P_5] \). Then the following P-collections \( M \) and \( N \) are equivalent:
\[ M = [A \ B] \]
\[ N = [P_1 \ P_2 \ P_3 \ P_4 \ P_5] \]
P = polytope(Px, Pc) Constructor for creating the polytope \( \mathcal{P} = \{ x \in \mathbb{R}^n \mid P^x x \leq P^c \} \); 

P = polytope(V) Constructor for creating the polytope out of extreme points; 

double(P) Access internal data of the polytope, e.g. \([P_x, P_c] = \text{double}(P)\); 

display(P) Displays details about the polytope \( \mathcal{P} \); 

nx = dimension(P) Returns dimension of a given polytope \( \mathcal{P} \); 

nc = nconstr(P) For a polytope \( \mathcal{P} = \{ x \in \mathbb{R}^n \mid P^x x \leq P^c \} \) returns number of rows of the \( P^x \) matrix; 

[ , ] Horizontal concatenation of polytopes into an array, e.g. \( \mathbf{P}_A = [\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3] \); 

( ) Subscripting operator for polytope arrays, e.g. \( \mathbf{P}_A(i) \) returns the \( i \)-th polytope in \( \mathbf{P}_A \); 

length(PA) Returns number of elements in a polytope array \( P_A \); 

end In indexing functions returns the final element of an array; 

\([c, r] = \text{chebyball}(\mathcal{P})\) Returns center \( c \) and radius \( r \) of the Chebychev ball of \( \mathcal{P} \); 

V = extreme(\( \mathcal{P} \)) Computes extreme points (vertices) of a polytope \( \mathcal{P} \); 

bool = isfulldim(\( \mathcal{P} \)) Checks if polytope \( \mathcal{P} \) is full dimensional; 

bool = isinside(\( \mathcal{P}, x \)) Checks if \( x \in \mathcal{P} \). Works also for polytope arrays.

<table>
<thead>
<tr>
<th>Table 13.3: Functions defined for class polytope.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual elements of a P-collection can be obtained using the standard referencing (i) operator, i.e.</td>
</tr>
</tbody>
</table>

\[ \mathbf{P} = \mathbf{M}(2) \]

will return the second element of the P-collection \( \mathbf{M} \) which is equal to \( \mathbf{P}_2 \) in this case. More complicated expressions can be used for referencing:

\[ \mathbf{Q} = \mathbf{M}([1, 3:5]) \]

will return a P-collection \( \mathbf{Q} \) which contains first, third, fourth and fifth element of P-collection \( \mathbf{M} \).

If we want to remove some element from a P-collection, we use the referencing command as follows:

\[ \mathbf{M}([1 \ 3]) = [] \]

which will remove the first and third element from the P-collection \( \mathbf{M} \). If some element of a P-collection is deleted, the remaining elements are shifted towards the start of the P-collection. This means that for \( \mathbf{N} = [\mathbf{P}_1 \ \mathbf{P}_2 \ \mathbf{P}_3 \ \mathbf{P}_4 \ \mathbf{P}_5] \), the command

\[ \mathbf{N}([1 \ 3]) = [] \]

will yield the P-collection \( \mathbf{N} = [\mathbf{P}_2 \ \mathbf{P}_4 \ \mathbf{P}_5] \) and the length of the array is 3. No empty positions in a P-collection are allowed. Analogously, empty polytopes are not being added to a P-collection.

A P-collection is still a polytope object, hence all functions which work on polytopes also support P-collections. This is an important feature in many geometric functions. The length of a given P-collection is obtained by
13.3 MPT Functions and Structures

As indicated in Section 13.1.2, the solution to an optimal control problem can be obtained by a simple call to `mpt_control`. The general syntax is given below:

\[
\text{ctrlStruct} = \text{mpt\_control(sysStruct, probStruct, Options)}
\]

Based on the system definition `sysStruct` and problem description `probStruct`, the main control routine `mpt_control` automatically calls one of the functions reported in Section 13.1.2 to calculate the explicit solution to a given problem. Once the control law is calculated, the solution is returned in form of the controller structure `ctrlStruct`. The system-, problem- and control- objects will be discussed in this chapter.

MPT provides a variety of control routines which are being called from `mpt_control`. Solutions to the following problems can be obtained

1. Constrained Finite Time Optimal Control (CFTOC) for LTI and PWA systems,
2. Constrained Infinite Time Optimal Control (CITOC) for LTI and PWA systems,
3. Constrained Minimum Time Optimal Control (CMTOC) for LTI and PWA systems,
4. Low Complexity (LowComp) Scheme for LTI and PWA systems.

The problem which will be solved depends on parameters of the system and problem structure, namely on type of the system (LTI or PWA), prediction horizon (finite or infinite) and the level of sub-optimality (optimal solution, minimum-time solution, low complexity). Different combinations of these three parameters lead to different optimization procedure, as reported in Table 13.4. See the documentation of the individual functions for implementation details.

As indicated in the previous sections, the optimization problem to be solved is described by two parameters:

\[
l = \text{length}(N)
\]
• description of the system to be controlled
• description of the optimization problem to solve

The dynamical system which is subject to control is defined by means of the system structure `sysStruct`. Properties of the optimal control problem are specified by the problem structure `probStruct`. The following subsections reveal detailed structure of these two descriptions. Consult also Section 13.3.4 for numerical examples.

### 13.3.1 System Structure sysStruct

The system object `sysStruct` is a structure which describes the system to be controlled. MPT can deal with two types of systems:

1. Discrete-time linear time-invariant (LTI) systems
2. Discrete-time Piecewise Affine (PWA) Systems

Both system types can be subject to constraints on control inputs, system states and/or outputs. In addition, constraints on slew rate of the control inputs can also be given.

#### LTI systems

In general, a constrained linear time-invariant system is defined by the following relations:

\[
\begin{align*}
x(k+1) &= Ax(k) + Bu(k) \\
y(k) &= Cx(k) + Du(k) \\
\text{subj. to} & \begin{cases} 
y_{\text{min}} \leq y(k) \leq y_{\text{max}} \\
u_{\text{min}} \leq u(k) \leq u_{\text{max}}
\end{cases}
\end{align*}
\]

Such an LTI system is defined by the following mandatory fields:

```plaintext
sysStruct.A = A;  
sysStruct.B = B;  
sysStruct.C = C;  
sysStruct.D = D;  
sysStruct.ymax = ymax;  
sysStruct.ymin = ymin;  
sysStruct.umax = umax;  
sysStruct.umin = umin;
```

Constraints on the slew rate of the control input \( u(k) \) can also be imposed by:

```plaintext
sysStruct.dumax = dumax;  
sysStruct.dumin = dumin;
```
which enforces \( \Delta u_{\text{min}} \leq u(k) - u(k-1) \leq \Delta u_{\text{max}} \). If order to deactivate a constraint, simply set the associated limiter to \( \text{Inf} \). An LTI system subject to parametric uncertainty and/or additive disturbances is described by the following set of relations:

\[
x(k+1) = A_{\text{unc}}x(k) + B_{\text{unc}}u(k) + w(k)
\]
\[
y(k) = Cx(k) + Du(k)
\]

where \( w(k) \) is an unknown, but bounded additive disturbance, i.e.

\[ w(k) \in W \quad \forall k \in \mathbb{N} \]

To specify an additive disturbance, set \( \text{sysStruct.noise} = W \) where \( W \) is a polytope object bounding the disturbance. A polytopic uncertainty can be specified by a cell array of matrices \( \text{Aunc} \) and \( \text{Bunc} \) as follows:

\[
\text{sysStruct.Aunc} = \{A_1, \ldots, A_n\};
\]
\[
\text{sysStruct.Bunc} = \{B_1, \ldots, B_n\};
\]

where \( \text{Aunc} \) and \( \text{Bunc} \) denote the vertices of the polytopic uncertainty.

**PWA Systems**

PWA systems are widely used to model hybrid and non-linear systems. The dynamical behavior of such systems is captured by relations of the following form:

\[
x(k+1) = A_i x(k) + B_i u(k) + f_i
\]
\[
y(k) = C_i x(k) + D_i u(k) + g_i
\]

subj. to

\[
\begin{align*}
y_{\text{min}} & \leq y(k) & \leq y_{\text{max}} \\
u_{\text{min}} & \leq u(k) & \leq u_{\text{max}} \\
\Delta u_{\text{min}} & \leq u(k) - u(k-1) & \leq \Delta u_{\text{max}}
\end{align*}
\]

Each dynamic \( i \) is active in a polyhedral partition bounded by the so-called guardlines:

\[
\text{guardX}_i x(k) + \text{guardU}_i u(k) \leq \text{guardC}_i,
\]

which means dynamic \( i \) will be applied if the above inequality is satisfied. Fields of \( \text{sysStruct} \) describing a PWA system are listed below:

\[
\begin{align*}
\text{sysStruct.A} &= \{A_1, \ldots, A_n\} \\
\text{sysStruct.B} &= \{B_1, \ldots, B_n\} \\
\text{sysStruct.C} &= \{C_1, \ldots, C_n\} \\
\text{sysStruct.D} &= \{D_1, \ldots, D_n\} \\
\text{sysStruct.f} &= \{f_1, \ldots, f_n\} \\
\text{sysStruct.g} &= \{g_1, \ldots, g_n\} \\
\text{sysStruct.guardX} &= \{\text{guardX}_1, \ldots,\text{guardX}_R\} \\
\text{sysStruct.guardU} &= \{\text{guardU}_1, \ldots, \text{guardU}_R\} \\
\text{sysStruct.guardC} &= \{\text{guardC}_1, \ldots, \text{guardC}_R\}
\end{align*}
\]
Note that all fields have to be cell arrays of matrices of compatible dimensions. $R$ stands for total number of different dynamics. If $\text{sysStruct.gu} \text{ardU}$ is not provided, it is assumed to be zero. The system constraints are defined by:

```matlab
sysStruct.ymax = ymax;
sysStruct.ymin = ymax;
sysStruct.umax = umax;
sysStruct.umin = umin;
sysStruct.dumax = dumax;
sysStruct.dumin = dumin;
```

Constraints on slew rate are optional and can be omitted. MPT is able to deal also with PWA systems which are affected by bounded additive disturbances:

$$x(k+1) = A_i x(k) + B_i u(k) + f_i + w(k)$$

where the disturbance $w(k)$ is assumed to be bounded for all time instances by some polytope $W$. To indicate that your system is subject to such a disturbance, set

```matlab
sysStruct.noise = W;
```

where $W$ is a polytope object of appropriate dimension. Polytopic uncertainty in the dynamics cannot be treated by the control schemes for PWA systems. We leave it up to the user to implement the scheme in [RKM03], if this functionality is required. Mandatory and optional fields of the system structure are summarized in Tables 13.5 and 13.6, respectively.

| A, B, C, D, f, g | State-space dynamic matrices in (13.2) and (13.3). Set elements to empty if they do not apply. |
| unmin, umax | Bounds on inputs $\text{unmin} \leq u(t) \leq \text{umax}$. |
| ymin, ymax | Constraints on the outputs $\text{ymin} \leq y(t) \leq \text{ymax}$. |
| guardX, guardU, guardC | Polytope cell array defining where the dynamics are active (for PWA systems). $D_i = \{(x, u) \mid \text{guardX}\{i\} x + \text{guardU}\{i\} u \leq \text{guardC}\{i\}\}$. |

Table 13.5: Mandatory fields of the system structure `sysStruct`.  

| dumin, dumax | Bounds on $\text{dumin} \leq u(t) - u(t-1) \leq \text{dumax}$. |
| noise | A polytope bounding the additive disturbance. |
| Aunct, Bunct | Cell arrays containing the vertices of the polytopic uncertainty. |
| Pbnd | Polytope limiting the feasible state-space of interest. |

Table 13.6: Optional fields of the system structure `sysStruct`.  

13.3.2 Problem Structure probStruct

The problem object probStruct is a structure which defines the optimization problem to be solved by MPT. The probStruct object contains all information which does not directly originate from the dynamical system (control objective, etc.). Let us recall a standard finite time optimization problems as described in Chapter 5:

\[
\min_{u_0, \ldots, u_{N-1}} \|P_N x(N)\|_{\text{norm}} + \sum_{k=0}^{N-1} \|R u(k)\|_{\text{norm}} + \|Q x(k)\|_{\text{norm}} \\
\text{subj. to} \quad \begin{cases} \\
x(k+1) = f_{\text{dyn}}(x(k), u(k), w(k)) \\
u_{\text{min}} \leq u(k) \leq u_{\text{max}} \\
du_{\text{min}} \leq u(k) - u(k-1) \leq du_{\text{max}} \\
y_{\text{min}} \leq g_{\text{dyn}}(x(k), u(k)) \leq y_{\text{max}} \\
x(N) \in T_{\text{set}} \end{cases}
\]

The function \(f_{\text{dyn}}(x(k), u(k), w(k))\) is the state-update function as defined in Section 13.3.1. Here:

- \(N\) prediction horizon
- \(\text{norm}\) objective norm, can be 1, 2 or Inf
- \(Q\) weighting matrix on the states
- \(R\) weighting matrix on the manipulated variables
- \(P_N\) weight imposed on the terminal state
- \(T_{\text{set}}\) terminal set

are parameters which do not originate from the system dynamics and are defined in the probStruct object. Note that the entries \(N\), \(\text{norm}\), \(Q\) and \(R\) are mandatory. Optional fields are summarized in Table 13.7.

MPT provides different control strategies with different levels of optimality. Specifically, it is possible to trade off controller performance for controller complexity by manipulation of the .subopt_lev field, as described in the following:

1. The cost-optimal solution leads to a control law which minimizes a given performance index. This strategy is enforced by

   \[ \text{probStruct.subopt_lev} = 0 \]

2. Another possibility is to use the time-optimal solution, i.e. the control law will push a given state to an invariant set around the origin as fast as possible. This strategy usually leads to simpler control laws (i.e. less controller regions are generated). This approach is enforced by

   \[ \text{probStruct.subopt_lev} = 1 \]
probStruct.y0bounds  Boolean variable. If false, no constraints are imposed on the initial output $y(0)$ (default is 0);
probStruct.tracking Boolean variable, if set to 1, the problem will be formulated as a state-tracking problem (default is 0);
probStruct.P N Weight on the terminal state. If not specified, it is assumed to be zero for quadratic cost, or $P_N = Q$ for linear cost;
probStruct.Tset Polytope object describing the terminal set. If not provided and probStruct.norm is 2, the LQR set around the origin will be calculated automatically to guarantee stability properties;
probStruct.Tconstraint An integer (0, 1, 2) denoting which auxiliary stability constraint to apply. 0 - no terminal constraint, 1 - LQR terminal set 2 - user-provided terminal set constraint). Note that if probStruct.Tset is given, Tconstraint will be set to 2 automatically;
probStruct.feedback Boolean variable, if set to 1, the problem is augmented such that $U = K x + c$ where $K$ is a state-feedback gain (typically an LQR controller) and the optimization aims to identify the proper offset $c$ (default is 0);
probStruct.FBgain If the former option is activated, a specific state-feedback gain matrix $K$ can be provided (otherwise a LQR controller will be computed automatically);
probStruct.xref By default the toolbox designs a controller which forces the state vector to convert to the origin. If you want to track some a-priori given reference point, provide the reference state in this variable. probStruct.tracking has to be 0 (zero) to use this option!
probStruct.uref Similarly a reference point for the manipulated variable (i.e. the equilibrium $u$ for state probStruct.xref can be specified here. If it is not given, it is assumed to be zero.

Table 13.7: Optional field of the probStruct structure.
3. The last option is to use a low-complexity control scheme. This approach aims at constructing a one-step solution and subsequent PWQ or PWA Lyapunov function computation to verify stability properties. The approach generally results in a small number of regions and asymptotic stability as well as closed-loop constraint satisfaction is guaranteed. If you want to use this kind of solution, use:

\[
\text{probStruct.subopt_lev} = 2
\]

### 13.3.3 Controller structure `ctrlStruct`

The controller structure is an object which includes all information obtained while solving a given optimal control problem. In general, it describes the obtained control law and can be used both for analysis of the solution, as well as for implementation of the control law. The fields of the structure are summarized in Table 13.3.3.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_n)</td>
<td>The polyhedral partition over which the control law is defined is returned in this field. It is, in general, a polytope array.</td>
</tr>
<tr>
<td>(F_i, G_i)</td>
<td>The PWA control law for a given state (x(k) \in \mathcal{P}_r) is given by (u = F_i r x(k) + G_i r). (F_i) and (G_i) are cell arrays.</td>
</tr>
<tr>
<td>(A_i, B_i, C_i)</td>
<td>The value function (J^*(x)) is returned in these three cell arrays and for a given state (x(k)) can be evaluated as (J(x) = x(k)' A_i r x(k) + B_i r x(k) + C_i r) where the prime denotes a transpose and (r) is the index of the active region (i.e. the region of (P_n) containing the given state (x(k))).</td>
</tr>
<tr>
<td>(P_{final})</td>
<td>In this field, the maximum achieved feasible set is returned. It, in general, corresponds to a convex union of all polytopes in (P_n).</td>
</tr>
<tr>
<td>(dynamics)</td>
<td>A vector which denotes which dynamics is active in which region of (P_n). (only important for PWA systems)</td>
</tr>
<tr>
<td>(details)</td>
<td>More details about the solution (e.g. total run time).</td>
</tr>
<tr>
<td>(overlaps)</td>
<td>Boolean variable denoting whether regions of the controller partition overlap.</td>
</tr>
<tr>
<td>(sysStruct)</td>
<td>System description in the (sysStruct) format.</td>
</tr>
<tr>
<td>(probStruct)</td>
<td>Problem description in the (probStruct) format.</td>
</tr>
</tbody>
</table>

Table 13.8: Fields of the controller structure `ctrlStruct`.

### 13.3.4 Numerical Examples

In order to obtain a feedback controller, it is necessary to specify both a system as well as the problem. We demonstrate the procedure on a simple second-order double integrator, with bounded input \(|u| \leq 1\) and output \(\|y(k)\|_\infty \leq 5\):

**Example 13.4.**
For this system we will now formulate the problem with quadratic cost objective in (8.3) and a prediction horizon of $N = 5$:

```
>> probStruct.norm=2; %Quadratic Objective
>> probStruct.Q=eye(2); %Objective: \( \min_U J = \sum x'Qx + u'Ru \)
>> probStruct.R=1; %Objective: \( \min_U J = \sum x'Qx + u'Ru \)
>> probStruct.N=5; %...over the prediction horizon 5
>> probStruct.subopt_lev=0; %Compute optimal solution,
%not low complexity.
```

If we now call

```
>> [ctrlStruct]=mpt_Control(sysStruct,probStruct); %Compute controller
>> mpt_plotPartition(ctrlStruct)
```

the controller for the given problem is returned and plotted (see Figure 13.2(c)), i.e., if the state $x \in PA(i)$, then the optimal input for prediction horizon $N = 5$ is given by $u = F_i\{i\}x + G_i\{i\}$. If we wish to compute a low complexity solution, we can run the following:

```
>> probStruct.subopt_lev=2; %Compute low complexity solution.
>> [ctrlStruct]=mpt_Control(sysStruct,probStruct);
>> mpt_plotPartition(ctrlStruct)
>> Q = ctrlStruct.details.lyapQ;
>> L = ctrlStruct.details.lyapL;
>> C = ctrlStruct.details.lyapC;
>> mpt_plotPWQ(ctrlStruct.finalPn,Q,L,C);
```

The resulting partition and Lyapunov function is depicted in Figures 13.2(d) and 13.2(e) respectively. In the following we will solve the PWA problem introduced in [MR03] by defining two different dynamics which are defined in the left- and right half-plane of the state space respectively.

**Example 13.5.**

```
>> H=[-1 1; -3 -1; 0.2 1; -1 0; 1 0; 0 -1]; %Polytopic state
>> K=[ 15; 25; 9; 6; 8; 10]; % constraints $Hx(k) \leq K$
```
we can now compute the low complexity feedback controller by defining the problem

\[
\begin{align*}
&\text{probStruct.norm=2;} & \text{%Quadratic Objective} \\
&\text{probStruct.Q=eye(2);} & \text{%Objective: } J = \text{sum} x'Qx + u'Ru... \\
&\text{probStruct.R=0.1;} & \text{%Objective: } J = \text{sum} x'Qx + u'Ru... \\
&\text{probStruct.subopt_lev=1;} & \text{%Compute low complexity controller.}
\end{align*}
\]

and calling the control function,
The result is depicted in Figure 13.3.

Figure 13.3: Controller partition obtained for Example 13.5.
Part V

APPENDIX
A

Notation

Throughout this thesis, as a general rule, scalars and vectors are denoted with the lower case letters (e.g., \(a, b, \ldots\)), matrices are denoted with the upper case letters (e.g., \(A, B, \ldots\)), and sets are denoted with the upper case calligraphic letters (e.g., \(A, B, \ldots\)).

In the following let \(i, j, m, n \in \mathbb{N}\) be integers such that \(i \leq m, j \leq n\), let \(s \in \mathbb{R}^n\) be a column vector, \(S \in \mathbb{R}^{n \times n}\) be a square matrix, \(R \in \mathbb{R}^{m \times n}\) be a rectangular matrix, \(I \subseteq \{1, \ldots, m\}\) be a set of integers, and \(S \subseteq \mathbb{R}^n\) be a subset of an \(n\)-dimensional real vector space.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>general placeholder (for any variable)</td>
</tr>
<tr>
<td>\ldots</td>
<td>ellipsis, “and so forth”</td>
</tr>
<tr>
<td>:=</td>
<td>left-hand side defined by right-hand side</td>
</tr>
<tr>
<td>=:</td>
<td>right-hand side defined by left-hand side</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>such that</td>
</tr>
<tr>
<td>\in</td>
<td>is element of (belongs to)</td>
</tr>
<tr>
<td>\forall</td>
<td>for all</td>
</tr>
<tr>
<td>\exists</td>
<td>exists at least one</td>
</tr>
<tr>
<td>\exists!</td>
<td>exists exactly one</td>
</tr>
<tr>
<td>\notin, #, \ldots</td>
<td>/ denotes negation</td>
</tr>
<tr>
<td>\to</td>
<td>mapping</td>
</tr>
<tr>
<td>{*, \ldots}</td>
<td>a set</td>
</tr>
<tr>
<td>[* \ldots]</td>
<td>a matrix (or a row vector)</td>
</tr>
<tr>
<td>(*, \ldots)</td>
<td>a composite variable; internal elements of a composite variable may belong to different spaces, e.g., ((s, i, S) \in \mathbb{R}^n \times \mathbb{N} \times 2^{\mathbb{R}^n})</td>
</tr>
</tbody>
</table>

**Logic variables**

<table>
<thead>
<tr>
<th>\lor</th>
<th>or</th>
</tr>
</thead>
<tbody>
<tr>
<td>\land</td>
<td>and</td>
</tr>
<tr>
<td>\implies</td>
<td>implies</td>
</tr>
<tr>
<td>\iff</td>
<td>if and only if</td>
</tr>
</tbody>
</table>

**Spaces**
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^S$</td>
<td>power set (set of all subsets of $S$)</td>
</tr>
<tr>
<td>$\mathbb{N}$</td>
<td>positive integers, $\mathbb{N} = \mathbb{Z}_{\geq}$</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>real numbers</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>space of $n$-dimensional real (column) vectors</td>
</tr>
<tr>
<td>$\mathbb{R}^{m \times n}$</td>
<td>space of $m$ by $n$ real matrices</td>
</tr>
<tr>
<td>$\mathbb{R}_{\geq}$</td>
<td>non-negative real numbers</td>
</tr>
<tr>
<td>$\mathbb{R}_{&gt;}$</td>
<td>positive real numbers</td>
</tr>
<tr>
<td>$\mathbb{S}^n$</td>
<td>symmetric $n \times n$ matrices</td>
</tr>
<tr>
<td>$\mathbb{S}_{\geq}$</td>
<td>symmetric positive semidefinite $n \times n$ matrices</td>
</tr>
<tr>
<td>$\mathbb{Z}$</td>
<td>integers</td>
</tr>
<tr>
<td>$\mathbb{Z}_{\geq}$</td>
<td>non-negative integers</td>
</tr>
<tr>
<td>$\mathbb{Z}_{&gt;}$</td>
<td>positive integers</td>
</tr>
</tbody>
</table>

**Vectors**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{0}_n$</td>
<td>vector of zeros, $\mathbb{0}_n := [0 \ 0 \ldots \ 0]' \in \mathbb{R}^n$</td>
</tr>
<tr>
<td>$\mathbb{1}_n$</td>
<td>vector of ones, $\mathbb{1}_n := [1 \ 1 \ldots \ 1]' \in \mathbb{R}^n$</td>
</tr>
<tr>
<td>$&lt;$</td>
<td>element wise comparison, (similar for $\leq, =, \geq, &gt;$)</td>
</tr>
<tr>
<td>$s'$</td>
<td>row vector</td>
</tr>
<tr>
<td>$s_{(i)}$</td>
<td>$i$-th element of a vector</td>
</tr>
<tr>
<td>$s_{(\mathcal{I})}$</td>
<td>vector formed from the elements indexed by $\mathcal{I}$</td>
</tr>
<tr>
<td>$</td>
<td>s</td>
</tr>
<tr>
<td>$|s|$</td>
<td>any vector norm</td>
</tr>
<tr>
<td>$|s|_1$</td>
<td>sum of absolute elements of a vector</td>
</tr>
<tr>
<td>$|s|_2$</td>
<td>Euclidian norm</td>
</tr>
<tr>
<td>$|s|_{\infty}$</td>
<td>largest absolute element of a vector</td>
</tr>
</tbody>
</table>

**Matrices**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>identity matrix (of appropriate dimension)</td>
</tr>
<tr>
<td>$S \ (\succeq) &gt; 0$</td>
<td>positive (semi)definite matrix</td>
</tr>
<tr>
<td>$S \ (\preceq) &lt; 0$</td>
<td>negative (semi)definite matrix</td>
</tr>
<tr>
<td>$R'$</td>
<td>matrix transpose</td>
</tr>
<tr>
<td>$R_{(i)}$</td>
<td>$i$-th row of a matrix</td>
</tr>
<tr>
<td>$R_{(\mathcal{I})}$</td>
<td>matrix formed from the rows indexed by $\mathcal{I}$</td>
</tr>
<tr>
<td>$R_{(i,j)}$</td>
<td>$(i, j)$-th element of a matrix</td>
</tr>
<tr>
<td>$R_{(\cdot, j)}$</td>
<td>$j$-th column of a matrix</td>
</tr>
<tr>
<td>$S^{-1}$</td>
<td>inverse of the square matrix</td>
</tr>
<tr>
<td>$\text{det}(S)$</td>
<td>determinant of the square matrix</td>
</tr>
<tr>
<td>$\text{rank}(R)$</td>
<td>rank of a matrix</td>
</tr>
<tr>
<td>$\text{range}(R)$</td>
<td>the subspace spanned by the columns of $R$</td>
</tr>
<tr>
<td>$\text{null}(R)$</td>
<td>the (right) null space of $R$, ${s \in \mathbb{R}^n \mid Rs = 0}$</td>
</tr>
<tr>
<td>$\text{Tr}(S)$</td>
<td>the trace of a matrix, sum of diagonal elements of $S$</td>
</tr>
</tbody>
</table>
### Symbol Meaning

#### Sets
- ∅: the empty set
- (⊂) ⊆: (strict) subset
- (⊃) ⊇: (strict) superset
- ∩: intersection
- ∪: union
- \: set difference
- ⊕: Pontryagin difference
- ⊖: Minkowski sum
- |I|: number of elements (cardinality) of a set I
- S: the closure of S
- ∂S: the boundary of S
- aff(S): the affine hull of S
- co(S): the convex hull of S
- dim(S): the dimension of S
- affdim(S): the affine dimension of S
- int(S): the strict interior of S
- relint(S): the relative interior of S

#### Optimization
- inf: infimum
- max: maximum
- min: minimum
- sup: supremum

#### Functions
- dom(f): domain of the function f, i.e., set of values for which f is defined
- range(f): range of the function f, i.e., set of all values that f can take as its argument varies over domain

#### SPECIFIC
- $n_x$: the number of states, $n_x \in \mathbb{N}$
- $n_u$: the number of inputs, $n_u \in \mathbb{N}$
- $x$: the state vector, $x \in \mathbb{R}^{n_x}$
- $u$: the input vector, $u \in \mathbb{R}^{n_u}$
- $J$: the cost function, $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$
- $J^*$: the value function, i.e. the optimal cost function, $J^* : \mathbb{R}^{n_x} \to \mathbb{R}$
- $f_{PWA}$: piecewise affine (state update) function, $f_{PWA} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$
- $O$: order of complexity
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$lp(n,m)$</td>
<td>complexity of an LP with $n$ variables and $m$ constraints</td>
</tr>
</tbody>
</table>

**Acronyms**

ARE  Algebraic Riccati Equation  
CFTOC Constrained Finite Time Optimal Control  
CITOC Constrained Infinite Time Optimal Control  
DP Dynamic Program(ming)  
LMI Linear Matrix Inequality  
LP Linear Program(ming)  
LQR Linear Quadratic Regulator  
LTI Linear Time Invariant  
MILP Mixed Integer Linear Program  
MIQP Mixed Integer Quadratic Program  
MPC Model Predictive Control  
mp-LP multi-parametric Linear Program  
mp-MILP multi-parametric Mixed Integer Linear Program  
mp-MIQP multi-parametric Mixed Integer Quadratic Program  
mp-QP multi-parametric Quadratic Program  
PWA Piecewise Affine  
PWQ Piecewise Quadratic  
QP Quadratic Program(ming)  
RHC Receding Horizon Control  
SDP Semi Definite Program(ming)
B

Publication List

Here, all publications and technical reports written during my graduate studies the Automatic Control Laboratory, ETH Zürich, are listed in chronological order:

- **Efficient On-line Computation of Explicit Model Predictive Control;** F. Borrelli, M. Baotic, A. Bemporad and M. Morari; Tech-Report AUT01-15, 2001; Automatic Control Lab, ETH Zürich, Switzerland. [BBBM01b].

- **Efficient On-line Computation of Explicit Model Predictive Control;** F. Borrelli, M. Baotic, A. Bemporad and M. Morari; Conference on Decision and Control 2001, Orlando, Florida, USA. [BBBM01a].

- **An Efficient Algorithm for Computing the State Feedback Optimal Control Law for Discrete Time Hybrid Systems;** F. Borrelli, M. Baotic, A. Bemporad and M. Morari; Tech-Report AUT02-04, 2002; Automatic Control Lab, ETH Zürich, Switzerland. [BBBM02].

- **An Efficient Algorithm for Multiparametric Quadratic Programming;** M. Baotic; Tech-Report AUT02-05, 2002; Automatic Control Lab, ETH Zürich, Switzerland. [Bao02].

- **Hybrid System Theory Based Optimal Control of an Electronic Throttle;** M. Baotic, M. Vaska, N. Peric and M. Morari; Tech-Report AUT02-17, 2002; Automatic Control Lab, ETH Zürich, Switzerland. [BVMP02].

- **A new Algorithm for Constrained Finite Time Optimal Control of Hybrid Systems with a Linear Performance Index;** M. Baotic, F.J. Christophersen and M. Morari; Tech-Report AUT02-18, 2002; Automatic Control Lab, ETH Zürich, Switzerland. [BCM02].

- **Infinite Time Optimal Control of Hybrid Systems with a Linear Performance Index;** M. Baotic, F.J. Christophersen and M. Morari; Tech-Report AUT03-04, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [BCM03c].

- **Constrained Optimal Control of Discrete-Time Linear Hybrid Systems;** F. Borrelli, M. Baotic, A. Bemporad and M. Morari; Tech-Report AUT03-05, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [BBBM03b].

- **Multi-object adaptive cruise control;** R. Mobus, M. Baotic and M. Morari; Hybrid Systems: Computation and Control Conference 2003, Prague, Czech Republic. [MBM03].
• *An Efficient Algorithm for Computing the State Feedback Optimal Control Law for Discrete Time Hybrid Systems*; F. Borrelli, M. Baotić, A. Bemporad and M. Morari; American Control Conference 2003, Denver, USA. [BBBM03a].

• *Hybrid System Theory Based Optimal Control of an Electronic Throttle*; M. Baotić, M. Vašak, N. Perić and M. Morari; American Control Conference 2003, Denver, USA. [BVMP03].

• *Polycover*; M. Baotić and F. D. Torrisi; Tech-Report AUT03-11, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [BT03].

• *Low Complexity Control of Piecewise Affine Systems with Stability Guarantee*; P. Grieder, M. Kvasnica, M. Baotić and M. Morari; Tech-Report AUT03-13, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [GBK03].

• *Hybrid Theory Based Model Predictive Control of Electrical Drives with Friction*; I. Petrović, M. Baotić, L. Matić and N. Perić; Proc. 10th European Conference on Power Electronics and applications, EPE 2003, Toulouse, France. [PBMP03].

• *Hybrid Systems Modeling and Control*; M. Morari, M. Baotić and F. Borrelli; European Journal of Control, Volume 9, Numbers 2-3, September 2003, Pages 177–189. [MBB03].

• *A new Algorithm for Constrained Finite Time Optimal Control of Hybrid Systems with a Linear Performance Index*; M. Baotić, F.J. Christophersen and M. Morari; European Control Conference 2003, Cambridge, UK. [BCM03a].

• *Stability Analysis of Hybrid Systems with a Linear Performance Index*; F.J. Christophersen, M. Baotić and M. Morari; Tech-Report AUT03-14, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [CBM03].

• *Active Vibration Suppression using switched PZTs*; D. Niederberger, M. Baotić and M. Morari; Tech-Report AUT03-06, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [NBM03].

• *Multi-Parametric Toolbox (MPT)*; M. Kvasnica, P. Grieder, M. Baotić and M. Morari; Tech-Report AUT03-17, 2003; Automatic Control Lab, ETH Zürich, Switzerland. [KGBM03].

• *Infinite Time Optimal Control of Hybrid Systems with a Linear Performance Index*; M. Baotić, F.J. Christophersen and M. Morari; Conference on Decision and Control 2003, Maui, Hawaii, USA. [BCM03b].

• Dynamic programming for constrained optimal control of discrete-time linear hybrid systems; F. Borrelli, M. Baotić, A. Bemporad and M. Morari; Paper accepted to Automatica, 2004; Automatic Control Lab, ETH Zürich, Switzerland. [BBBM05b].

• Electronic throttle state estimation and hybrid theory based optimal control; M. Vašak, M. Baotić, I. Petrović and N. Perić; Int. Symposium on Industrial Electronics ISIE 2004, Ajaccio, France. [VBPP04].

• Low Complexity Control of Piecewise Affine Systems with Stability Guarantee; P. Grieder, M. Kvasnica, M. Baotić and M. Morari; American Control Conference 2004, Boston, USA. [GKBM04].

• Low Complexity Control of Piecewise Affine Systems with Stability Guarantee; P. Grieder, M. Kvasnica, M. Baotić and M. Morari; accepted to Automatica, 2005. [GKBMO5].

• Stability Analysis of Hybrid Systems with a Linear Performance Index; F. J. Christophersen, M. Baotić, M. Morari; Conference on Decision and Control 2004, Paradise Island, Bahamas. [CBM04].

• Multi-Parametric Toolbox (MPT): User’s Manual; M. Kvasnica, P. Grieder, M. Baotić, F.J. Christophersen; 2004; Automatic Control Lab, ETH Zürich, Switzerland. [KGB04].

• Optimal Control of PWA Systems by Exploiting Problem Structure; M. Barić, M. Baotić, P. Grieder and M. Morari; accepted, IFAC World Congress, Prague, Czech Republic, 2005. [BBGM05].

• Hybrid Theory Based Time-Optimal Control of an Electronic Throttle; M. Vašak, M. Baotić, I. Petrović and N. Perić; Int. Symposium on Industrial Electronics ISIE 2005, Dubrovnik, Croatia. [VBPP05].

• Efficient On-line Computation of Constrained Optimal Control; M. Baotić, F. Borrelli, A. Bemporad and M. Morari; submitted to journal, 2005. [BBBM05a].
Curriculum Vitae

Mato Baotić
Born on December 23, 1973 in Brčko, Bosnia and Hercegovina

09/2001 – 03/2005 Doctorate at Automatic Control Laboratory, ETH Zürich
(Dr. sc. ETH Zürich)

10/2000 – 07/2001 Visiting researcher at Automatic Control Laboratory, ETH Zürich.

03/1997 – 10/2000 Graduate studies, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (mr. sc.)

10/1992 – 02/1997 Undergraduate studies, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (dipl. ing.)

09/1988 – 04/1992 High School Center Orašje, Bosnia and Hercegovina
### Bibliography


